Faculté des Sciences de Tunis
LIPAH

École doctorale IAEM Lorraine
LORIA

Recommandation Personnalisée Hybride

THÈSE

présentée et soutenue publiquement le 11 novembre 2015

pour l'obtention du

Diplôme de Docteur en Informatique

par

Sonia Ben Ticha

Composition du jury

Président :	Sadok Ben Yahia, Professeur, FST, Université de Tunis El Manar
Rapporteurs :	Mohamed Mohsen Gammoudi, Professeur, Université de la Manouba Yolaine Bourda, Professeure, SUPELEC Paris
Examinateur :	Isabelle Tellier, Professeure, Université Paris 3 Sorbonne Nouvelle
Directeurs :	Anne Boyer, Professeure, Université de Lorraine Khaled Bsaïes, Professeur, FST, Université de Tunis El Manar
Co-encadrant :	Azim Roussanaly, Maitre de conférences, Université de Lorraine

Préparée sous la convention de cotutelle UTM-FST (Tunisie) – UL (France)

Mis en page avec la classe thesul.

Je dédie cette thèse
à Skander,
à mes enfants Myriam, Ali et Youssef,
à mes parents.

Remerciements

Au terme du travail de thèse présenté ici, je voudrais dire un grand Merci à tous ceux qui, de quelle que manière que ce soit, m'ont aidée à le mener à terme.

En premier lieu mes remerciements et ma gratitude vont tout particulièrement à mes directeurs de thèse pour leurs soutiens, leurs encouragements, leurs patiences et leurs précieux conseils durant toutes ces années de thèse. Je remercie Anne Boyer pour sa confiance et sa disponibilité. Son admirable accueil au sein de l'équipe KIWI, et sa constante bonne humeur ont facilité mes séjours au LORIA et m'ont permis de bien avancer dans mes travaux de recherche. Je remercie Khaled Bsaïes pour sa confiance, son aide, son écoute et le soutien qu'il m'a apporté durant les moments les plus difficiles. Je remercie Azim Roussanaly pour son extrême gentillesse, son optimisme constant même durant les périodes les plus critiques de cette thèse. Je serai toujours reconnaissante de leurs bienveillances aussi bien sur le plan scientifique que sur le plan humain.

Mes remerciements s'adressent également aux membres du jury qui m'ont fait l'honneur d'accepter d'évaluer ce travail. Je remercie Sadok Ben Yahia d'avoir bien voulu présider ce jury. Je tiens à remercier tout particulièrement Yolaine Bourda et Mohamed Mohsen Gammoudi d'avoir accepté de rapporter cette thèse. Leurs commentaires et suggestions me furent très précieux. De même, je remercie Isabelle Tellier d'avoir accepté d'être examinatrice de ma thèse.

Je tiens à adresser mes plus vifs remerciements à Narjes Doggaz pour m'avoir aidé à trouver le sujet de la thèse. Elle est également à l'origine de la collaboration entre l'équipe KIWI du LORIA et le laboratoire LIPAH. Qu'elle trouve ici l'expression de ma profonde reconnaissance.

Je remercie mon très cher et magnifique mari Skander pour son constant soutien et ses encouragements tout au long de ces années malgré son travail très prenant. C'est en partie grâce à lui que ce travail a pu être mené à son terme. Je remercie également ma très chère fille Myriam, qui malgré son très jeune âge, a toujours trouvé les mots pour me 'booster' dans les moments de relâchement, pour le courage et le sens de la responsabilité qu'elle a manifestés lors de mes déplacements au LORIA. Je remercie en outre mes parents et mes frères pour leurs aides et leurs soutiens. Je remercie Daloula, qui a pris soin des enfants durant mes absences et qui m'a toujours encouragée et aidée. Je tiens également à remercier Annick Salvan pour son aide lors de mes déplacements en France, et ma cousine Lamia Ben Ticha pour son aide le jour de la soutenance et ses encouragements tout au long de ces années de thèse.

Je remercie les membres de l'équipe KIWI que j'ai eu la chance de côtoyer durant mes séjours au LORIA. J'ai énormément apprécié l'ambiance de travail et les moments agréables que nous avons partagés et qui se sont avérés pour moi riche tant sur le plan scientifique que sur le plan personnel. Je remercie Armel Brun et Sylvain Castagnons pour les conseils avisés qu'ils m'ont prodigués. Je remercie Charif Alchiekh Haydar, Geoffray Bonnin, Nicolas Jones, Ilham Essilimani, Manel Sobra, Sahbi Sidhom, Lina Fahed, Rita Alexandrova, Brahim Batouche, Ngoc-Chan Nguyen, Cédric Bernier pour les discussions enrichissantes et les moments de détentes que nous avons partagés.

Mes remerciements vont également à toutes les personnes que j'ai croisées durant ma thèse : Anja Habacha pour son soutien dans les moments difficiles, les membres du laboratoire LIPAH : Aymen Sellaouti, Mohamed Naouai, Nefissa Annabi, Ines Mouakher, Ines Elouedi et tous les

autres collègues pour les discussions enrichissantes que nous avons échangées ; Faouzia Ahmouda, Antoinette Courrier pour leur grande aptitude à résoudre les problèmes administratifs liés à la thèse ; à toute personne que j'ai oubliée de citer ici et qui a contribué d'une manière ou d'une autre à la réalisation de cette thèse.

Sommaire

Table des figures

Liste des tableaux

Liste des Algorithmes

Liste des Notations

Symbole	Description	Référence
CB	Content Based : filtrage basé sur le contenu avec lequel nous comparons notre approche	section 3.7.2 page 73.
DIL	Diversité Intra-Liste. Plus grande est sa valeur meilleure, est la diversité des recommandations.	formule (1.13) page 30.
D_A	Liste des descripteurs (latents ou pas). Chaque descripteur est décrit dans la dimension des utilisateurs par une colonne de la matrice Q_A.	
i	un item.	
I	Ensemble des items.	
I_u	Ensemble des items notés par u.	
$IBCF$	Item Based Collaborative Filtering : filtrage collaboratif basé sur les items.	section 2.1.1.3 page 37.
$IMDb$	Internet Movie Database	http ://www.imdb.com.
K_A	Dimension du $MSUA$: nombre de colonnes de la matrice Q_A, $K_A = \|D_A\|$.	
F_A	L'ensemble des descripteurs de l'attribut A.	
F_{A_i}	L'ensemble des descripteurs de l'attribut A décrivant l'item i.	
FC	Filtrage Collaboratif	
$NbItems(f_d)$	Nombre d'items décrits par le descripteur f_d.	page 98
$NbMinVC$	Seuil indiquant le nombre minimum de votes que doit avoir un item pour être initialement affecté à un cluster.	section 4.3.2 page 87.
$NbMinIC$	Nombre minimum d'items composant initialement un cluster.	page 98
$NbVotes(i)$	Nombre de votes de l'item i.	
MAE	Mean Absolute Error. Plus petite est sa valeur meilleure, est la précision de la recommandation.	formule (1.2) page 26.
$MSIA_A$	Matrice Sémantique des Items par Attribut pour l'attribut A.	formule (3.9)page 64.
$MSUA$	Modèle Sémantique des Utilisateurs par Attribut modélisé par la matrice Q_A.	section 3.4.3 page 67.
MSU	Modèle Sémantique des Utilisateurs, modélisé par la matrice Q.	section 3.4 page 64 .
Mv	Matrice des Votes des utilisateurs.	section 3.3.1 page 61 .

Symbole	**Description**	**Référence**
L	Liste des attributs pertinents.	
u	un utilisateur.	
U	Ensemble des utilisateurs.	
U_i	Ensemble des utilisateurs ayant noté l'item i.	
$UBCF$	User Based Collaborative Filtering : filtrage collaboratif basé sur les utilisateurs.	section 2.1.1.1 page 36 .
$USCF$	User Semantic Collaborative Filtering : notre approche.	
PUI_i	Profil Usage de l'Item i.	formule (3.3) page 61.
$PUIa_i$	Profil Usage Ajusté de l'item i.	formule (3.5) page 61.
PUU_u	Profil Usage de l'Utilisateur u.	formule (3.2) page 61.
$Précision$	Mesure la précision d'une Top-N liste. Plus grande est sa valeur, meilleure est la précision de la Top-N liste.	formule (1.5) page 27.
PSA_{i_A}	Profil Sémantique par Attribut de l'item i pour l'attribut A.	formule (3.8) page 64.
$Q_{A_{\|U\|,K_A}}$	Matrice modélisant le $MSUA$.	
Q	Matrice modélisant le MSU, les utilisateurs en ligne et les descripteurs (latents ou pas) en colonnes.	
$RMSE$	Root Mean Absolute Error. Plus petite est sa valeur, meilleure est la précision de la recommandation.	formule (1.3) page 26.

Introduction générale

1 Contexte

L'évolution qu'a connue le monde informatique, et plus particulièrement le réseau internet, a considérablement transformé notre mode de vie dans bien des aspects. Nous achetons sur le net, nous recherchons des informations sur le net, nous lisons les news sur le net, nous regardons des films sur le net, nous écoutons de la musique sur le net et la liste est encore longue. Ainsi, nous passons une importante partie de notre journée à bénéficier de services électroniques, e-services [1], disponibles sur la toile. Face à cet engouement, Internet s'est avéré un marché économique très lucratif et un moyen de communication très accessible, attirant ainsi tous les acteurs économiques, politiques, et sociaux. Tout fournisseur d'un e-service voudrait se positionner sur le net et attirer le plus grand nombre d'usagers. Cependant, la surabondance de e-services, de ressources et d'informations sur le net rend cette tâche difficile, surtout que l'utilisateur peut se trouver noyé dans cet océan d'informations faisant de l'accès aux ressources pertinentes une tâche fastidieuse.

Il devient alors impératif d'assister l'utilisateur et de faciliter son accès aux ressources susceptibles de l'intéresser et adaptées à ses besoins personnels. Parmi les solutions envisagées, on trouve l'adaptation des interfaces pour faciliter l'exploration et la recherche au sein d'un service [Gajos *et al.*, 2008], les systèmes à base de navigation sociale [Millen *et al.*, 2006], la recherche personnalisée sur le web [Micarelli *et al.*, 2007], l'aide à la navigation personnalisée [Brusilovsky, 2007], les outils statistiques aidant les utilisateurs à mieux définir leurs recherches par mots-clefs [Smyth *et al.*, 2005], la présentation personnalisée du contenu pour le web [Bunt *et al.*, 2007] etc. Une des principales solutions, très prisée dans le commerce électronique, consiste à intégrer un système de recommandation personnalisé [Adomavicius and Tuzhilin, 2005; Almazro *et al.*, 2010; Ricci *et al.*, 2011a; Lü *et al.*, 2012; Bobadilla *et al.*, 2013] au sein de l'architecture logicielle du e-service de sorte à suggérer, de façon automatique et personnalisée, une liste de ressources pertinentes adaptées aux préférences de chaque utilisateur.

Les systèmes de recommandation [Goldberg *et al.*, 1992] ont émergé comme un domaine de recherche à part entière au milieu des années 1990 à la suite de la publication des premiers articles sur le filtrage collaboratif [Resnick *et al.*, 1994; Shardanand and Maes, 1995; Hill *et al.*, 1995]. Bien que le domaine de recherche des systèmes de recommandation soit plus récent et moins répandu que le filtrage d'information et la recherche documentaire [Salton, 1989; Belkin and Croft, 1992] et en particulier les moteurs de recherche, son intérêt dans l'industrie des e-services n'en est pas moins important. Les systèmes de recommandation ont été utilisés dans de

1. [Boyer *et al.*, 2008] ont défini un service comme étant un intermédiaire établissant un lien entre une ressource et un utilisateur.

nombreux domaines tels que le e-commerce (Amazon [2] [Linden *et al.*, 2003],CastroSchez2011), la vidéo à la demande (MovieLens [3] [Miller *et al.*, 2003], Netflix [4], IMDB [5],Winoto2010), la musique (Ringo [Shardanand and Maes, 1995], LastFM [6],Shulong2011), le ciblage publicitaire (Google AdSense [7]), la télévision ([Smyth and Cotter, 2001],Barragans2010), les documents [Porcel *et al.*, 2009; Serrano-Guerrero *et al.*, 2011] , les livres ([Mooney and Roy, 2000; Ziegler *et al.*, 2005; Núñez-Valdéz *et al.*, 2012], Goodreads [8]), les news [Konstan *et al.*, 1997; Billsus *et al.*, 2002], les pages web [Billsus and Pazzani, 1998; Mobasher *et al.*, 2002; McNally *et al.*, 2011], le e-learning [Zaiane, 2002; Bobadilla *et al.*, 2009] et bien d'autres . Bien que très différents, tous ces domaines proposent aux utilisateurs un choix très important de produits dont le nombre peut atteindre plusieurs millions. L'intégration d'un système de recommandation filtrant, de manière personnalisée, les produits en ne proposant que les plus pertinents peut s'avérer très intéressante aussi bien pour l'utilisateur que pour le fournisseur du e-service. En effet, le système de recommandation facilite la tâche à l'utilisateur en lui proposant des produits diversifiés répondant à ses attentes, ce qui permet à moyen terme d'accroître sa satisfaction, et de garantir sa fidélisation. En garantissant la fidélisation de ses clients, le fournisseur du e-service augmentera ses entrées en augmentant soit le nombre de produits vendus (s'il s'agit d'un site commercial), soit le nombre de ressources consultées (pour les sites non commerciaux).

FIGURE 1 – Recommandation de livres par Amazon (www.amazon.com)

Item est le terme utilisé pour désigner ce que le système recommande à l'utilisateur. Chaque item est considéré comme une unité élémentaire et insécable. Il peut s'agir de la musique à écouter, du livre à acheter, du film à visualiser, de l'article en ligne à lire etc. Un système de

2. www.amazon.com
3. wwww.movielens.org
4. www.netflix.com
5. www.imdb.com
6. www.lasrfm.fm
7. www.google.com/adsense
8. www.goodreads.com

recommandation traite généralement un seul type d'item (CDs, films, livres, news) [Ricci *et al.*, 2011a]. Les algorithmes et techniques de recommandation sont alors appliqués pour inférer des recommandations pertinentes relatives à ce type d'item. La figure 1 illustre le système de recommandation de livres de Amazon. On y voit une liste de recommandations pour l'utilisateur courant, effectuées en partie, à partir des livres déjà en sa possession. Dans ce qui suit, nous utiliserons le terme *item* pour désigner toute ressource pouvant être recommandée et *utilisateur courant* pour désigner l'utilisateur connecté au e-service et recevant des recommandations.

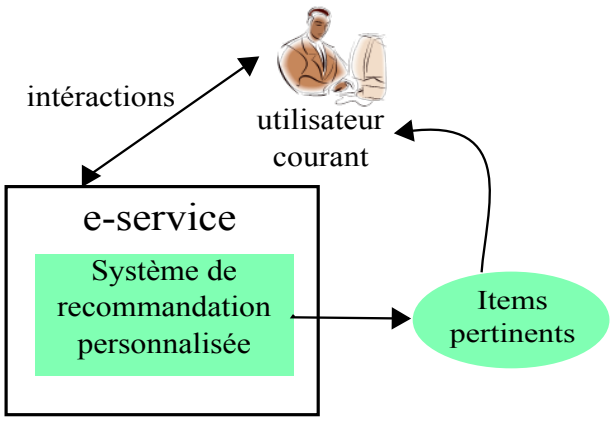


FIGURE 2 – Système de recommandation personnalisée

Un système de recommandation personnalisée (voir figure 2) peut alors être défini comme étant un ensemble d'outils et de techniques intégrés dans un e-service proposant à l'utilisateur courant, de manière personnalisée, le sous ensemble d'items qui l'intéresse et qu'il n'aurait pas consulté spontanément. Afin de remplir ce rôle, un système de recommandation personnalisé doit connaître les goûts de l'utilisateur courant, c'est-à-dire, modéliser ses préférences à partir de l'observation de son comportement et des éventuelles appréciations qu'il dépose lors de ses différents accès au e-service. L'approche basée sur l'observation du comportement de l'utilisateur à partir de ses différentes interactions avec un e-service donné est appelée *analyse des usages*. Les préférences d'un utilisateur donné sont représentées par les évaluations qu'il apporte sur les items qu'il a consultés. L'analyse des usages a pour objectif d'inférer les appréciations de chaque utilisateur à partir de ses différentes interactions avec le e-service. Une appréciation indique l'intérêt ou le désintérêt que porte un utilisateur pour un item qu'il a consulté. Elle peut être représentée par une note (appelée aussi *vote*) attribuée explicitement par l'utilisateur ou déduite à partir des différentes actions qu'il effectue sur le e-service. Un exemple d'appréciation est donné dans la figure 1 illustrant le système de recommandation de livres de Amazon, on y voit que l'utilisateur a le choix entre attribuer une note en cliquant sur le nombre d'étoiles de 1 à 5, ou indiquer qu'il n'est pas intéressé par le livre proposé ou encore préciser qu'il le possède déjà. Le profil personnalisé, d'un utilisateur donné, issu de l'analyse des usages est alors représenté par la liste de ses évaluations sur les items qu'il a consultés. Une fois le profil de l'utilisateur courant défini, les algorithmes de recommandation permettent de prédire parmi l'ensemble des items existants, le sous ensemble d'items correspondant à ses préférences. Il peut s'agir alors de lui présenter une liste d'items triés selon un ordre de pertinence qui lui soit adapté (voir figure 1).

2 Techniques de recommandation

Différentes approches sont envisageables, pour la recommandation personnalisée à partir des usages, suivant que l'on considère le contenu des ressources ou au contraire une dimension plus subjective qui est l'intérêt qu'elle présente pour un utilisateur donné. Lorsque l'on considère le contenu de la ressource, on se place dans le cas des *systèmes de recommandation basée sur le contenu* [Pazzani and Billsus, 2007; Lops *et al.*, 2011], et les items recommandés sont ceux proches (similaires) des items appréciés par le passé par l'utilisateur courant. Cette technique exige la connaissance de la structure de l'item, elle est généralement utilisée pour des ressources de type texte en exploitant des solutions proposées dans le domaine du filtrage de l'information et de la recherche documentaire [Salton, 1989; Baeza-Yates and Ribeiro-Neto, 1999].

En plus du fait que cette technique est limitée par les informations disponibles sur le contenu des ressources recommandées, elle souffre également du problème de sur-spécialisation (Over-specialization) [Shardanand and Maes, 1995; Adomavicius and Tuzhilin, 2005]. En effet, le système ne recommande à l'utilisateur courant que les ressources similaires à celles qu'il a appréciées par le passé. Ainsi, un utilisateur qui n'a aucune connaissance du cinéma maghrébin ne recevra jamais de recommandation de films maghrébins, même les plus primés. Or la diversité des recommandations est très souvent un critère d'évaluation de la qualité d'un système de recommandation [Shani and Gunawardana, 2011; Bobadilla *et al.*, 2013].

L'autre technique très utilisée est *le filtrage collaboratif* [Schafer *et al.*, 2007; Desrosiers and Karypis, 2011; Koren and Bell, 2011; Ekstrand *et al.*, 2011]. Un système de recommandation collaboratif recommande les ressources ayant été appréciées par les utilisateurs partageant les mêmes goûts que l'utilisateur courant. L'hypothèse fondamentale du filtrage collaboratif est que si deux utilisateurs u et v notent n items de façons similaires, ou ont des comportements similaires (par exemple, achètent les mêmes produits, regardent les mêmes films, écoutent la même musique), alors il est très probable qu'ils aient des appréciations ou des comportements similaires vis à vis d'autres items. Le point fort de cette approche est qu'elle est indépendante du type de la ressource (image, texte, audio, vidéo) puisque seules les données issues de l'analyse des usages sont utilisées pour faire de la recommandation. Cependant, elle ne fonctionne que pour des ressources qui ont déjà été observées par un certain nombre d'utilisateurs. Le filtrage collaboratif doit faire face à plusieurs défis [Sen *et al.*, 2009], parmi lesquels le problème de données manquantes (sparsity). En effet, chaque utilisateur n'évalue qu'un nombre très limité de ressources comparé au nombres d'items pouvant être recommandés. Or, la pertinence des recommandations pour un utilisateur donné dépend du nombre d'évaluations effectuées. Si son profil est vide ou contient très peu de votes, il sera difficile aux algorithmes de filtrage collaboratif de l'associer à un groupe d'utilisateurs et la recommandation sera impossible ou peu satisfaisante.

En plus des défis propres à chaque technique, le filtrage collaboratif et le filtrage basé sur le contenu doivent faire face au problème de passage à l'échelle (scalability) pour de gigantesques bases de données avec un nombre croissant d'utilisateurs et de ressources [Adomavicius and Tuzhilin, 2005; Castagnos and Boyer, 2007; Koren *et al.*, 2009; Lops *et al.*, 2011; Luo *et al.*, 2013]. Par ailleurs, les systèmes de recommandation basée sur l'usage ne peuvent produire des recommandations fiables à un utilisateur que s'il a consulté (évalué) un certain nombres de ressources au préalable. Ceci est nécessaire afin que le système arrive à identifier ses préférences. Par conséquent, un nouvel utilisateur avec très peu de ressources consultées (évaluées) ne serait pas en mesure d'obtenir des recommandations satisfaisantes. Ce problème est connu sous le nom du problème de démarrage à froid pour les utilisateurs (user cold start

problem)[Park and Chang, 2009].

Pour remédier aux insuffisances de chaque technique et profiter de leurs points forts, des systèmes de recommandation hybrides combinant le filtrage sur le contenu et le filtrage collaboratif ont fait l'objet de plusieurs travaux de recherche. Plusieurs méthodes d'hybridation ont été ainsi développées [Balabanović and Shoham, 1997; Burke, 2002; Burke, 2007; de Campos *et al.*, 2010; Porcel *et al.*, 2012; Dooms, 2013; Moreno *et al.*, 2014]. La plupart de ces systèmes sont orientés processus, il s'agit d'exécuter du filtrage collaboratif sur les résultats issus d'un filtrage sur le contenu ou vice versa, soit pour résoudre le problème des données manquantes[Pazzani and Billsus, 1997; Ren *et al.*, 2008; Shinde and Kulkarni, 2012; Choi *et al.*, 2012] soit pour résoudre le problème du démarrage à froid [Mobasher *et al.*, 2004; Maneeroj and Takasu, 2009]. Toutefois, très peu de travaux se sont intéressés à définir le lien entre le contenu sémantique des ressources et les évaluations des utilisateurs[Symeonidis *et al.*, 2007; Manzato, 2012]. Ce lien pourrait expliquer l'intérêt porté par un utilisateur pour une ressource donnée.

3 Problématique

L'objectif de cette thèse est de développer une nouvelle technique d'hybridation intégrant l'aspect sémantique des ressources dans un processus collaboratif. Nous nous plaçons dans un contexte à faible risque, c'est-à-dire, les applications pour lesquelles la perte encourue pour un utilisateur par une mauvaise recommandation n'est pas importante, et la description des items à recommander n'est pas complexe. C'est le cas par exemple de la location de DVDs, de l'achat de livres, de la visualisation de film ou du choix d'un restaurant. Ce n'est pas le cas par exemple, des applications dédiées à la vente d'automobiles, aux transactions financières, à la vente immobilières, à l'achat d'ordinateurs, à l'achat d'appareils photo ou aux voyages. Par ailleurs, les seules informations dont nous disposons sont le contenu sémantique pouvant être extrait (ou disponible) à partir des ressources. Aucune information n'est disponible sur l'utilisateur à part un numéro permettant de le distinguer des autres utilisateurs et les différentes évaluations sur les items qu'il a consultés. Plus formellement, il s'agit de traiter un problème de recommandation défini comme suit :

Étant donné un contexte à faible risque, un ensemble d'items définis par leur profil sémantique, un ensemble d'utilisateurs définis par leur profil usage, le problème consiste à construire un nouveau système de recommandation hybride intégrant l'aspect sémantique des items dans une approche collaborative.

Le profil sémantique des items est défini à partir de la description textuelle des items. Pazzani et al. [Pazzani and Billsus, 2007] ont défini trois types de représentations pour décrire le contenu textuel d'un item : *structurée*, *semi-structurée* ou *non structurée*. Dans une représentation structurée, tous les items sont décrits par un petit nombre d'attributs (aussi appelés champs, colonnes, variables selon les domaines de recherche) et les valeurs prises par chaque attribut sont connues. Pour une représentation non structurée, l'item est décrit par un texte libre (la description d'un restaurant, le résumé d'un article de journal,...). Pour ce type de représentation, une étape de pré-traitement est nécessaire pour extraire les informations pertinentes et les structurer. Le contenu des items est analysé et les principales caractéristiques (mots clé, concepts, ...) sont extraites en appliquant des techniques de traitement du langage naturel issues de la recherche et du filtrage de l'information [Baeza-Yates and Ribeiro-Neto, 1999;

Belkin and Croft, 1992]. L'étape de pré-traitement d'un contenu non structuré ne rentre pas dans le cadre de ce travail. Nous supposons disposer d'un ensemble de termes ou mots clés représentant les items non structurés. Dans une représentation semi-structurée, un item est à la fois décrit par des attributs et par du texte libre, par exemple un film peut être décrit par son genre, ses acteurs, son réalisateur et son synopsis.

Le système que nous proposons est constitué d'un seul système de recommandation intégrant les données sémantiques et les données issues de l'analyse des usages dans un algorithme de recommandation personnalisée hybride. L'idée est de définir un nouveau profil utilisateur construit à partir des données sémantiques sur les items et des données d'usage. Ce profil que nous appelons *Modèle Sémantique des Utilisateurs* définit les préférences des utilisateurs pour le contenu des items à partir de leurs appréciations pour les items.

Le modèle sémantique des utilisateurs définissant un nouveau profil pour les utilisateurs est utilisé pour déterminer des communautés d'utilisateurs en se basant sur leurs appréciations pour les descripteurs des items.

Parmi les approches collaboratives, les algorithmes basés sur les plus proches voisins [Desrosiers and Karypis, 2011] (appelés aussi basés sur la mémoire [Breese *et al.*, 1998] ou basés sur des heuristiques [Adomavicius and Tuzhilin, 2005]) jouissent d'une grande popularité en raison de leur simplicité, de leur efficacité, et de leur capacité à produire des recommandations personnalisées pertinentes, Amazon en est le plus grand exemple [Linden *et al.*, 2003; Almazro *et al.*, 2010]. Le principe de base de ces algorithmes est de recommander à l'utilisateur courant les items appréciés par les utilisateurs de sa communauté. La communauté d'un utilisateur donné est constituée de ses plus proches voisins avec lesquels il partage des préférences similaires. La similarité entre deux utilisateurs est déterminée en mesurant la corrélation entre les évaluations des items qu'ils ont notés en commun.

L'idée est de profiter de l'efficacité et de la simplicité de ces algorithmes pour faire de la recommandation en utilisant le modèle sémantique des utilisateurs pour déterminer les plus proches voisins de l'utilisateur courant.

Toutefois, parmi les défis auxquels sont confrontés les algorithmes de filtrage collaboratif basés sur les voisins, on trouve essentiellement la sensibilité aux données manquantes[Sarwar *et al.*, 2000b; Bobadilla and Serradilla, 2009; Desrosiers and Karypis, 2011] et le problème du passage à l'échelle [Sarwar *et al.*, 2001; Luo *et al.*, 2012].

Sensibilité aux données manquantes : les données manquantes sont un problème commun à la majorité des systèmes de recommandation [Papagelis *et al.*, 2005; Kim *et al.*, 2010], dû au fait que les utilisateurs notent très peu d'items, ainsi, même les items les plus populaires se trouvent notés par un petit nombre parmi la totalité des utilisateurs. Les données manquantes sont à l'origine de nombreux problèmes auxquels doivent faire face les algorithmes de filtrage collaboratif basés sur les voisins vu que la similarité entre deux utilisateurs est basée sur les items qu'ils ont co-évalués.

Le taux de couverture : un utilisateur ne peut recevoir des recommandations que si le système arrive à identifier le groupe d'utilisateurs avec lequel il partage les mêmes préférences.

Un utilisateur ayant évalué des items non communs ne peut pas recevoir de recommandations, ce problème est connu sous le nom de *mouton gris* (*Gray Sheep*). Par ailleurs, le calcul de similarité exige une paire d'utilisateurs ayant évalué un certain nombre d'items. En effet, si deux utilisateurs n'ont que 3 items en commun, l'algorithme les qualifiera comme totalement similaires, s'ils les ont notés de la même manière. Les auteurs [Schickel and Faltings, 2006] ont démontré qu'il faut un minimum de 20 votes par utilisateur et un minimum de 7 votes en commun pour que la mesure de similarité ne soit pas biaisée et que les recommandations soient pertinentes.

Transitivité des voisins : les mesures de similarité ne permettent pas de détecter les utilisateurs de goûts similaires n'ayant aucun items en commun comme étant des voisins proches. En effet, si l'utilisateur u est le voisin proche de l'utilisateur w et w est le voisin proche de l'utilisateur z, si z et u ne partagent aucun item alors la similarité entre u et z ne pourra pas être calculée et le système sera dans l'incapacité de détecter qu'ils sont également voisins proches.

Passage à l'échelle : le problème de la complexité des calculs est essentiellement dû au coût élevé imputé à la détermination des voisins surtout pour des applications de e-commerce avec un nombre très élevé d'utilisateurs et d'items[Takács *et al.*, 2009]. Or, dans ce type d'application, le système ne doit pas dépasser une fraction de seconde pour fournir des recommandations pertinentes à l'utilisateur courant.

Par ailleurs, l'utilisation du contenu pour faire de la recommandation suppose la nécessité de disposer d'une description variée des items à recommander, ce qui n'est pas toujours le cas [Adomavicius and Tuzhilin, 2005]. En effet, la présence et la variété des données sémantiques est un facteur important dans la pertinence des recommandations fournies par un système de recommandation basé sur le contenu [Pazzani and Billsus, 2007; Lops *et al.*, 2011; Bobadilla *et al.*, 2013]. Pour ne pas être tributaire de la quantité des données sémantiques disponibles, notre système de recommandation doit permettre de recommander des items pertinents même lorsque très peu de données décrivent des items.

4 Contributions

Le principe de notre système de recommandation hybride, que nous avons intitulé *User Semantic Collaborative Filtering (USCF)*, consiste à intégrer l'aspect sémantique des items dans un processus collaboratif basé sur les plus proches voisins. Il est constitué de deux composants : *l'apprentissage du modèle sémantique des utilisateurs* et *la recommandation* des items pertinents à l'utilisateur courant. Le premier, se charge de l'apprentissage du profil personnalisé des utilisateurs, *le Modèle Sémantique des Utilisateurs*, en inférant les préférences des utilisateurs pour le contenu des items à partir de leurs appréciations pour les items. Le second, calcule la prédiction des votes des items non observés par l'utilisateur courant et lui recommande les plus pertinents d'entre eux.

Apprentissage du profil sémantique des utilisateurs

L'originalité de notre approche se situe dans l'apprentissage du profil sémantique des utilisateurs. Les différentes sources d'information (structurées, semi structurées ou non structurées) sont ramenées à une structure du type attribut-valeurs. Lorsque les items sont structurés, ils sont décrits par un petit nombre d'attributs. Par exemple, la liste des attributs décrivant un film peut

être le genre du film, le réalisateur, les acteurs et la date de parution. Nous nous sommes intéressés dans ce travail à modéliser l'intérêt que porte l'utilisateur aux attributs décrivant les items et leurs influences sur ses préférences en construisant un nouveau profil personnalisé : *le modèle sémantique des utilisateurs*. La plupart des travaux [Pazzani and Billsus, 1997; Melville *et al.*, 2002; Symeonidis *et al.*, 2007] définissant un tel modèle, ont traité l'information décrivant les items sans aucune sélection ni distinction. Dans notre approche, nous commençons par faire une sélection sur les attributs en ne conservant que ceux susceptibles d'influencer le choix des utilisateurs, ainsi, seuls les attributs pertinents sont retenus. Un *profil sémantique des utilisateurs par attribut* est construit pour chaque attribut pertinent. Le profil sémantique de l'utilisateur est alors le résultat de la fusion des différents profils sémantiques des utilisateurs par attribut. Afin de tenir compte des caractéristiques de chaque attribut, nous avons défini une typologie des attributs *les attributs indépendants* et *les attributs dépendants*. Les attributs sont dits indépendants, si le nombre de leurs valeurs est stable et ne varie pas avec le nombre d'items. Les attributs sont dits dépendants, si le nombre de leurs valeurs est très variable et augmente avec le nombre d'items. Pour chaque catégorie d'attributs, nous proposons une approche appropriée et différents algorithmes pour l'apprentissage du modèle sémantique des utilisateurs par attribut. Le découpage de l'information sémantique décrivant les items par attribut a permis, d'une part de paralléliser le processus d'apprentissage du profil sémantique, et d'autre part d'appliquer un algorithme spécifique à chaque attribut tenant compte de ses caractéristiques dans la construction du profil sémantique des utilisateurs par attribut.

Clustering flou pour l'apprentissage du profil sémantique des utilisateurs par attribut

Nous avons proposé un modèle basé sur le clustering flou en appliquant la méthode Fuzzy C-Mean [Bezdek, 1981] pour construire le profil sémantique des utilisateurs par attribut. Ce modèle est appliqué aux attributs indépendants multi-valués, c'est-à-dire pouvant avoir plusieurs valeurs pour un même item. Ce travail a fait l'objet des deux publications [Ben Ticha *et al.*, 2011b; Ben Ticha *et al.*, 2011a].

Clustering pour l'apprentissage du profil sémantique des utilisateurs par attribut

Nous avons proposé un modèle pour construire le profil sémantique des utilisateurs par attribut à partir d'un clustering en utilisant la méthode K-Means [MacQueen, 1967]. Ce modèle s'applique aux attributs indépendants mono-valués. Un algorithme d'initialisation utilisant une ontologie de domaine pour déterminer le nombre initial de clusters ainsi que le centre initial de chaque cluster a également été proposé. Ce travail a fait l'objet de la publication [Ben Ticha *et al.*, 2012].

Fréquence des votes pour l'apprentissage du profil sémantique des utilisateurs

Nous proposons un algorithme permettant de construire le profil sémantique des utilisateurs à partir de la fréquence des votes des utilisateurs et en appliquant la mesure du TF-IDF (Term Frequency Inverse Document Frequency) très utilisée dans le domaine de la recherche et du filtrage d'information [Salton, 1989]. Cet algorithme est destiné aux attributs dépendants dont le nombre de valeurs peut être très élevé et dépasser dans certains cas le nombre d'items. Afin de réduire la dimension du modèle sémantique des utilisateurs, deux algorithmes de réduction de la dimension ont été proposés. Le premier se base sur un filtrage des descripteurs de l'attribut

en fonction des votes des utilisateurs. Le second applique une analyse sémantique latente (LSA) [Dumais, 2004]. Une partie de cette contribution a fait l'objet de la publication [Ben Ticha *et al.*, 2013].

Rocchio pour l'apprentissage du profil sémantique des utilisateurs par attribut

Nous proposons un algorithme construisant le profil sémantique des utilisateurs en appliquant l'algorithme de Rocchio [Buckley and Salton, 1995a]. Cet algorithme s'adresse essentiellement aux attributs dépendants. Étant donné que le nombre de valeurs d'un attribut dépendant peut être élevé, nous avons proposé trois méthodes de réduction de la dimension. La première applique un filtrage sur les descripteurs des attributs. Les deux suivantes appliquent une analyse sémantique latente (LSA) comme technique de factorisation de matrice. Une partie de ce travail a fait l'objet de la publication [Ben Ticha *et al.*, 2014] qui a été nominée au *best student paper*. Il a également fait l'objet d'une version étendue dans la collection LNBIP [Ben Ticha *et al.*, 2015].

Expérimentations

Pour l'évaluation de nos modèles, nous avons adopté une démarche empirique consistant à effectuer une série d'expérimentations sur des données réelles issues du jeu de données MovieLens Dataset [9] largement exploité par la communauté scientifique travaillant sur les systèmes de recommandation. MovieLens [MovieLens, 2014] est un système de recommandation de films, le jeu de données mis à la disposition des chercheurs fournit essentiellement les évaluations des utilisateurs pour les films qu'ils ont observés. Nous avons couplé ces données avec les données issues de la base de données des films IMDB [10] et du jeu de données HetRec2011 [11] pour extraire les informations sémantiques sur les items. Nous avons comparé notre approche à une approche purement collaborative basée sur les voisins, et une approche basée sur le contenu. Pour ce faire, nous avons utilisé deux critères d'évaluation, la précision et la diversité des recommandations.

5 Liste des publications

1. Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. *User-feature model for hybrid recommender system.* In 4th International Conference on Information Systems and Economic Intelligence-SIIE'2011, Marrakech, Maroc, 2011.
2. Sonia Ben Ticha, Azim Roussanaly, and Anne Boyer. User semantic model for hybrid recommender systems. In The 1st Int. Conf. on Social Eco-Informatics - SOTICS, pages 95–101, Barcelona, Espagne, October 2011. IARIA.
3. Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. *User semantic preferences for collaborative recommendations.* In Christian Huemer and Pas- quale Lops, editors, E-Commerce and Web Technologies (EC-WEB), volume 123 of Lecture Notes in Business Information Processing, pages 203–211. Springer Berlin Heidelberg, 2012.
4. Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. *Feature frequency inverse user frequency for dependant attribute to enhance recommendations.* In The Third Int. Conf. on Social Eco-Informatics - SOTICS, Lisbon, Portugal, November 2013. IARIA.

9. http ://grouplens.org/datasets/movielens/
10. www.imdb.com
11. http ://grouplens.org/datasets/hetrec-2011/

5. Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. *User semantic model for dependent attributes to enhance collaborative filtering.* In Proceedings of the 10th International Conference on Web Information Systems and Technologies (WEBIST), pages 205–212, Barcelona, Spain, April 2014. Nominé au *best student paper*.
6. Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. *Rocchio algorithm to enhance semantically collaborative filtering.* In Valérie Monfort and Karl-Heinz Krempels, editors, Web Information Systems and Technologies, Lecture Notes in Business Information Processing (LNBIP), chapter 19. Springer International Publishing Switzerland, 2015.

6 Organisation du manuscrit

Le présent manuscrit est composé de six chapitres, en plus de celui-ci, dont voici les centres d'intérêts :

Chapitre 1 Les systèmes de recommandation personnalisée : est consacré à la présentation des concepts de base d'un système de recommandation personnalisée. Nous nous intéressons dans ce chapitre aux données collectées et comment contribuent-elles à la définition du modèle personnalisé de l'utilisateur. Nous présentons également les principales méthodes et critères utilisés dans la littérature pour évaluer un système de recommandation personnalisée.

Chapitre 2 Les techniques de recommandations : nous présentons dans ce chapitre les deux principales techniques de recommandation personnalisée à savoir le filtrage collaboratif et le filtrage basé sur le contenu. Pour chaque technique, nous présentons les principales approches utilisées. Un intérêt particulier est donné aux algorithmes de filtrage collaboratif basés sur les voisins vu que nous les utilisons dans notre approche. Les différentes techniques d'hybridation combinant le filtrage collaboratif et le filtrage basé sur le contenu sont également présentées.

Chapitre 3 User Semantic Collaborative Filtering : présente l'architecture générale de notre approche. Le chapitre commence par formaliser les données en entrée, puis présente notre approche pour l'apprentissage du modèle sémantique des utilisateurs et l'algorithme de recommandation. Une description détaillée des jeux de données utilisés, des critères d'évaluation et des algorithmes par rapport auxquels nous avons évalué notre approche est également présentée.

Chapitre 4 Clustering pour l'apprentissage du modèle sémantique des utilisateurs : présente les deux contributions appliquant un clustering pour l'apprentissage du modèle sémantique des utilisateurs par attribut. La première en utilisant un clustering flou et la seconde en utilisant la méthode K-Means. Chacune des deux contributions est suivie d'une étude expérimentale évaluant les algorithmes que nous proposons.

Chapitre 5 Apprentissage du profil sémantique des utilisateurs pour les attributs dépendants : présente les deux contributions destinées à la construction du modèle sémantique des utilisateurs pour les attributs dépendants. La première, basée sur la fréquence des votes des utilisateurs, et la seconde en appliquant la méthode de Rocchio. Pour chacune des deux approches, différentes méthodes de réduction de la dimension ont été appliquées. Chaque contribution est suivie d'une étude expérimentale évaluant les résultats obtenus.

Chapitre 6 Synthèse des résultats, conclusion et perspectives : nous clôturons ce manuscrit par un dernier chapitre dans lequel nous présentons une synthèse des résultats que nous avons obtenus, un résumé sur nos différentes contributions et leurs extensions possibles.

Chapitre 1

Les systèmes de recommandation personnalisée

Un système de recommandation personnalisée a pour objectif d'apprendre les goûts et les préférences d'un utilisateur donné pour lui recommander par la suite les items susceptibles de l'intéresser. La recommandation consiste en fait à choisir parmi les items disponibles, ceux répondant au mieux aux centres d'intérêt de l'utilisateur courant et de les afficher selon un ordre de préférence qui lui soit adapté.

Dans ce chapitre, nous présentons les principes de base des systèmes de recommandation personnalisée. La section 1.1 décrit les données collectées et comment elles contribuent à la définition du modèle utilisateur, primordial dans le processus de recommandation. La section 1.2 formalise le problème de recommandation et présente les principales fonctions d'un système de recommandation personnalisée. Comme tout système, un système de recommandation doit être soumis à une évaluation, la section 1.3 résume les principales méthodes utilisées dans la littérature pour évaluer un système de recommandation.

1.1 Personnalisation

Un système de recommandation est en fait un système de traitement de l'information [Ricci *et al.*, 2011a] qui exploite différents types de données pour établir des recommandations. Ces données peuvent être scindées en trois catégories : les données relatives aux **items** à recommander, les données sur les **utilisateurs** recevant les recommandations et les données issues de l'**analyse des usages**, c'est-à-dire les données collectées à partir de l'observation du comportement de l'utilisateur vis-à-vis du e-service ; ces dernières représentent les relations entre les utilisateurs et les items. Toutes ces données peuvent être utilisées dans la modélisation personnalisée des utilisateurs. Cependant, leur exploitation est étroitement liée à la technique de recommandation utilisée [Adomavicius and Tuzhilin, 2005].

Dans cette section, nous présentons les données relatives aux items (section 1.1.1), aux utilisateurs (section 1.1.2) et celles issues de l'analyse des usages (section 1.1.3). La section 1.1.4 décrit brièvement les techniques de modélisation de l'utilisateur. Enfin l'adaptation du modèle utilisateur à l'évolution de ses centres d'intérêt est discutée à la section 1.1.5.

ID	Titre	Genre	Réalisateur	Origine	Année
625	Asfour Stah	Drama	Férid Boughedir	Tunisia	1990
718	Les visiteurs	Comedy	Jean-Marie Poiré	France	1993
1721	Titanic	Romance	James Cameron	USA	1997

TABLE 1.1 – Base de données de films

1.1.1 Typologie des données sur les items

Représentation : suivant la technique de recommandation utilisée, un item peut être représenté soit par un simple numéro permettant de l'identifier de manière unique, c'est le cas par exemple des systèmes de recommandation collaboratifs [Schafer *et al.*, 2007; Lousame and Sánchez, 2009; Ekstrand *et al.*, 2011], soit par une représentation sémantique comme c'est le cas des systèmes de recommandation basés sur le contenu [Lops *et al.*, 2011]. Dans le cas d'une représentation sémantique de l'item, ce dernier peut être décrit selon trois représentations [Pazzani and Billsus, 2007] : **structurée**, **non structurée** ou **semi-structurée**. La table 1.1 illustre le cas d'une représentation structurée, on y voit une table d'une base de données décrivant 3 films (un film par ligne). Les noms des colonnes représentent les *attributs* (aussi appelées *champs*, *variables* en fonction du domaine de recherche) de chaque film. Chaque ligne contient une valeur pour chaque attribut. Un identifiant unique "ID" dans le table 1.1 permet de distinguer entre les items de même nom, et est utilisé comme une clé d'accès aux autres attributs de l'item. Ainsi, une représentation **structurée** est définie par un petit nombre d'attributs, dans laquelle tous les items sont décrits par le même ensemble d'attributs, et l'ensemble des valeurs pouvant être prises par chaque attribut est connu [Pazzani and Billsus, 2007]. Lorsqu'un item est décrit par un texte libre (par exemple les articles de journaux électroniques), sa représentation est dite **non structurée**. Si un item est décrit à la fois par des attributs et du texte libre, il s'agit d'une représentation **semi-structurée** (chaque film peut également être décrit par son synopsis). Dans le cas de texte libre, une étape d'analyse de contenu [Lops *et al.*, 2011] est obligatoire pour convertir le texte libre en une représentation structurée. Cette étape utilise en général des techniques d'indexation d'information issues des domaines de filtrage d'information et de la recherche documentaire [Salton, 1989]. Chaque texte est alors représenté par un ensemble restreint de termes représentant des mots-clés ou des concepts. Par exemple, le système de recommandation Fab System [Balabanović and Shoham, 1997] qui recommande des pages web aux utilisateurs, représente le contenu d'une page web par les 100 mots les plus importants.

Ainsi, dans sa forme la plus simple, un item peut être représenté par un modèle vectoriel construit à partir de sa représentation sémantique [Pazzani and Billsus, 2007]. Dans un modèle vectoriel (VSM) [12], un item est représenté par un vecteur de poids, où chaque poids est associé à un terme. Un **terme** peut être, soit la valeur d'un attribut (après avoir appliqué une discrétisation des attributs à valeur continue), soit un mot-clé (ou un concept) extrait à partir d'un texte si l'item est non structuré. Un poids est soit une valeur booléenne indiquant la présence du terme dans l'item, soit une valeur numérique indiquant, la fréquence, l'importance ou la probabilité du terme dans l'item, calculée en utilisant des techniques d'indexation de l'information [Salton, 1989]. Dans [Mobasher *et al.*, 2004], les auteurs ont utilisé la fréquence comme indicateur du poids d'un terme dans un item. Parmi les techniques d'indexation, le TF-IDF (Terme Frequency-Inverse Document Frequency) est la mesure la plus fréquemment utilisée [Salton, 1989], elle définit un poids compris entre 0 et 1 indiquant l'importance d'un terme dans un item en réduisant l'effet

12. Vector Space Model

des termes les plus populaires. Il est possible dans certains systèmes d'utiliser une représentation plus complexe des items en utilisant pas exemple les ontologies [Aciar *et al.*, 2007; Middleton *et al.*, 2004a; Lops *et al.*, 2011].

Utilité et complexité : un item est l'objet à recommander. Il peut être caractérisé par sa complexité et son utilité [Ricci *et al.*, 2011a]. L'utilité d'un item est positive si l'item est pertinent pour l'utilisateur, négative s'il n'est pas adapté à ses besoins et il a pris une mauvaise décision en le sélectionnant. La complexité se mesure par les propriétés de l'item à prendre en considération lors de la conception d'un système de recommandation. Par exemple, dans la recommandation d'article de presse en ligne, les concepteurs doivent tenir compte de la complexité d'un article (sa structure, sa représentation textuelle, mais aussi le facteur temps qui définit la dépendance entre une nouvelle et la date de son apparition). Par ailleurs, l'acquisition d'un item par un utilisateur est toujours associée à un coût qui inclut le coût cognitif relatif à la recherche de l'item et éventuellement le coût matériel relatif au paiement effectué pour l'achat de l'item. Si l'item est pertinent pour l'utilisateur, le coût est dominé par la satisfaction et la sensation d'avoir effectué le bon choix. Au contraire, si l'item n'est pas pertinent, l'utilité de l'item et de la recommandation est négative et la perte endurée est étroitement liée au coût associé. Dans des domaines tels que la vente d'automobiles, ou l'investissement financier, la perte encourue par une mauvaise recommandation peut être très élevée. Dans ce cas, on parle d'un contexte à *haut risque* dans lequel l'item est caractérisé par une grande complexité et une utilité élevée, il s'agit par exemple des caméras digitales, des PCs, des voyages, de l'emploi et de l'investissement financier [Montaner *et al.*, 2003]. Au contraire, dans un contexte à *faible risque*, les items se caractérisent par une faible complexité et une faible utilité. Il s'agit alors de la location de CDs, de l'achat de livres, de la lecture d'article en ligne ou de la visualisation de films. Dans ce travail, nous nous intéressons uniquement aux applications définies dans un contexte à faible risque.

1.1.2 Typologie des données sur les utilisateurs

Un utilisateur peut être décrit par un simple numéro permettant de l'identifier comme dans le cas des systèmes de recommandation collaboratifs ou basés sur le contenu. Il peut être également décrit par un ensemble d'informations démographiques. L'utilisateur est alors invité à remplir un formulaire pour renseigner des données sur certaines caractéristiques (âge, sexe, niveau d'étude, emploi), son emplacement géographique (ville, pays, origine), sur sa psychologie (style de vie, tempérament), ses loisirs. De telles données sont généralement utilisées dans les systèmes de recommandation démographiques comme par exemple la méthode implémentée par Krulwich dans LifeStyle Finder [Krulwich, 1997] exploitant une base de données démographique commerciale qui contient les préférences de la population d'un pays. Le principale problème est que les utilisateurs sont généralement très réticents à fournir volontairement des informations personnelles [ChoiceStream, 2006; Jacquet and Drouot, 2007], et même s'ils y sont contraints, il est probable qu'ils fournissent de fausses informations. Ainsi, il peut y avoir un doute quant à la fiabilité des données fournies par l'utilisateur, surtout dans un contexte à faible risque. Certaines informations peuvent cependant, être extraites automatiquement comme par exemple le pays (grâce à l'adresse IP).

Par ailleurs, certains systèmes peuvent demander à l'utilisateur de fournir des informations quant à ses goûts, ses préférences et ses centres d'intérêt en sélectionnant des catégories parmi une liste ou en répondant à des questionnaires. Le problème dans ce mode de collecte est qu'il exige un investissement de la part de l'utilisateur qui n'est pas toujours prêt à fournir, surtout dans

le cas d'un contexte à faible risque. Un autre inconvénient est que l'utilisateur est parfois dans l'incapacité de définir ce qui l'intéresse parce que ses goûts ne sont pas encore fixés [Montaner *et al.*, 2003].

1.1.3 Les données collectées par analyse des usages

La modélisation personnalisée des préférences nécessite la connaissance des goûts de l'utilisateur courant afin de lui recommander les items pertinents adaptés à ses préférences. Dans le cas des systèmes de recommandation, la collecte de ces données peut se faire, soit en les lui demandant explicitement (voir section 1.1.2), soit en analysant les traces laissées à la suite de ses interactions avec le e-service. Les interactions de l'utilisateur avec le système sont collectées moyennant des programmes spécifiques et stockées dans des fichiers log sur des serveurs. Chaque entrée du fichier log décrit une transaction (requête) entre l'utilisateur et le système. Une transaction contient une référence à l'item sélectionné par l'utilisateur (représenté par l'url de l'item), une référence à l'utilisateur, une description du contexte (adresse IP, date et heure d'accès), et d'autres informations qui varient en fonction de la nature du e-service. Dans le cas des systèmes de recommandation personnalisée, l'**analyse des usages** consiste à analyser les données disponibles dans les fichiers log afin de modéliser les préférences de l'utilisateur. L'analyse des usages doit, par conséquent, aboutir à inférer les évaluations apportées par l'utilisateur sur les items qu'il a consultés. Une évaluation indique l'utilité (l'intérêt ou le désintérêt) que porte l'utilisateur pour un item. Elle peut être soit sous forme d'une note (également appelée **vote**) qu'il attribue explicitement ou implicitement, soit sous forme d'annotations (tag) qu'il associe à l'item correspondant. Les évaluations des utilisateurs sont également considérées comme un **retour d'expérience** sur les recommandations effectuées par le système, elles permettent à leur tour d'évaluer le système de recommandation en mesurant le taux de recommandations pertinentes.

FIGURE 1.1 – Recommandation de livres par Amazon (www.amazon.com)

Les votes : dans tout ce qui suit, nous utiliserons le terme vote (traduction de rating) pour désigner la note attribuée à un item. Selon Sharfer et al. [Schafer *et al.*, 2007], un vote peut se présenter selon l'une des formes suivantes :

- *vote numérique* : défini sur une échelle de valeurs discrètes, comme par exemple 1-5 étoiles du système de recommandation de livre de Amazon illustré par "*Evaluer cet article*" dans la figure 1.1. Un vote élevé (proche de la limite supérieure) indique que l'utilisateur apprécie l'item, un vote faible exprime au contraire qu'il n'y trouve aucun intérêt.
- *vote ordinal* : exemple "très bien, bien, moyen, mauvais, très mauvais", l'utilisateur est invité à choisir parmi une liste de termes ordonnés et figés, celui décrivant le mieux son opinion. A chaque terme est en faite associé une valeur numérique.
- *vote binaire* : l'utilisateur doit seulement indiquer s'il aime ou n'aime pas l'item.
- *vote unaire* : indique que l'utilisateur a consulté, a acheté ou a noté positivement l'item. L'absence de vote signifie que le système ne dispose d'aucune information quant au lien entre l'utilisateur et l'item. La figure 1.1 illustre un exemple de vote unaire, l'utilisateur est sollicité pour indiquer s'il possède le livre recommandé en cochant la case "*vous l'avez déjà*", le fait de s'abstenir de répondre ne signifie pas qu'il ne le possède pas ou qu'il ignore son contenu.

Les votes peuvent être collectés de manière explicite ou implicite ou les deux à la fois. Dans sa forme explicite, l'utilisateur est invité à fournir explicitement son opinion sur un item. La figure 1.1 illustre trois exemples de votes explicites ("*vous l'avez déjà*", "*Vous n'êtes pas intéressé*", "*Évaluer cet article*"). Dans le cas de vote implicite, l'utilisateur ne fournit aucune appréciation sur l'item. C'est le système qui doit estimer l'opinion de l'utilisateur en analysant l'historique de ses interactions avec le système. Chan [Chan, 2000] a défini deux formules pour estimer le vote de l'utilisateur sur un item à partir des traces laissées dans les fichiers logs. La première formule estime la valeur du vote comme étant la fréquence de consultation de l'item, c'est-à-dire le nombre de fois où l'utilisateur a visité (ou cliqué sur) la page correspondant à l'item, ce critère est considéré comme le plus important dans la détermination de l'intérêt que porte l'utilisateur pour l'item. La seconde formule, plus sophistiquée, introduit en plus de la fréquence de consultation, la durée de consultation de la page, la récence de la consultation, l'affectation ou pas de la page de l'item à la liste des favoris, le pourcentage de liens visités (par rapport à la totalité des liens se trouvant dans la page de l'item). Castagnos [Castagnos, 2008] a généralisé cette formule de sorte à pouvoir y intégrer n'importe quel critère tel que l'impression, l'enregistrement, l'envoi par mail ou le partage de la page correspondant à l'item. Chaque critère est par ailleurs pondéré par un poids mesurant son importance. La formule définie par Castagnos, permet également d'obtenir un vote à valeur entière sur une échelle de valeurs spécifiques.

L'utilisation de votes implicites [Terveen *et al.*, 1997; Konstan *et al.*, 1997; Mobasher *et al.*, 2004] comme retour d'expérience peut s'avérer un excellent moyen pour l'apprentissage des préférences des utilisateurs lorsque le système ne dispose pas de votes explicites, cependant leur efficacité reste à prouver. L'utilisateur n'étant pas généreux en votes explicites, il peut s'avérer intéressant dans certains cas de combiner les votes implicites et les votes explicites pour combler ce manque. Plusieurs systèmes, tels que Amazon [Linden *et al.*, 2003], exploitent aussi bien les votes implicites (opération d'achat) que les votes explicites pour faire de la recommandation. Dans ce qui suit nous utiliserons le terme *vote* pour désigner aussi bien un vote explicite ou un vote implicite à valeurs discrètes définies sur une échelle de valeurs spécifiques.

Les votes sont les données d'usage les plus fréquemment utilisées dans les systèmes de recommandation, ils sont fréquemment représentés par une matrice appelée *matrice des votes*. Chaque

ligne de la matrice des votes représente un *utilisateur* et chaque colonne un *item* et la valeur se trouvant au croisement d'une ligne et d'une colonne indique le vote associé au couple *(utilisateur, item)* correspondant. L'absence de vote au croisement d'une ligne et d'une colonne indique que l'utilisateur n'a pas encore évalué l'item correspondant, on parle alors de *valeur manquante*. La table 1.2 illustre un exemple de matrice de votes du système de recommandation MovieLens [13], les valeurs manquantes sont désignées par un ?.

	Asfour Stah	Les visiteurs	Titanic	The Matrix
user 1	4	1	2	?
user 2	?	5	?	5
user 4	?	?	5	?
user 5	1	?	5	?

TABLE 1.2 – Matrice des votes de MovieLens, votes explicites sur une échelle de 1 à 5

Les annotations (les tags) : une autre forme d'évaluation permettant à l'utilisateur d'exprimer explicitement son opinion sur un item est l'association d'annotations. Une annotation est un mot-clé (une chaîne de caractères) exprimé par l'utilisateur sous une forme assez libre et qui lui est propre. Cette technique connue sous le nom d'indexation personnelle ou folksonomie (traduction du mot anglais folksonomy), permet ainsi de refléter le point de vue de l'utilisateur sur plusieurs critères tels que le prix ou la qualité du produit s'il s'agit d'un achat, une liste de mots résumant un film, un livre ou un document. Elle permet de classer collaborativement les items selon une nomenclature plus fine que celle des catégories figées définies par les experts du domaine. Cette approche est le noyau des systèmes à base de navigation sociale [Millen *et al.*, 2006]. Bien qu'elle ait l'avantage de produire des annotations à grande échelle dont la qualité s'améliore avec l'augmentation du nombre de taggeurs [Marinho *et al.*, 2011], elle souffre des problèmes tels que le problème de données manquantes (les utilisateurs ne fournissent pas assez d'annotations), le problème de polysémie (une même annotation peut avoir plusieurs interprétations), le problème d'individualisation (l'utilisateur associe une annotation pour son organisation personnelle comme par exemple en ajoutant "à lire"). Il est, par ailleurs, possible de proposer à l'utilisateur un lexique préalablement défini dans lequel choisir les annotations. Le site MovieLens (voir figure 1.2) recommandant des films, offre à l'utilisateur la possibilité d'ajouter ses propres mots-clés ou de choisir parmi une liste d'annotations représentant les tags les plus populaires. L'exploitation des annotations pour faire de la recommandation a fait l'objet de plusieurs travaux de recherche [Marinho *et al.*, 2011], au point ou dans certains travaux [Sen *et al.*, 2009], le terme "*Tagommender*" a été utilisé pour désigner les systèmes de recommandation basés sur les annotations des utilisateurs.

1.1.4 Modélisation des utilisateurs

La personnalisation joue un rôle central dans le processus de recommandation et la pertinence des recommandations y est étroitement liée. Il ne peut y avoir personnalisation sans la définition d'un modèle utilisateur, dans le cas contraire, il s'agit de recommandations non personnalisées comme pour recommander les items les plus populaires (la sélection des top-10 par exemple). Le modèle utilisateur a pour objectif de profiler l'utilisateur en modélisant ses goûts et ses

13. Extraite à partir du jeu de données MovieLens [JeuMovieLens, 2014]

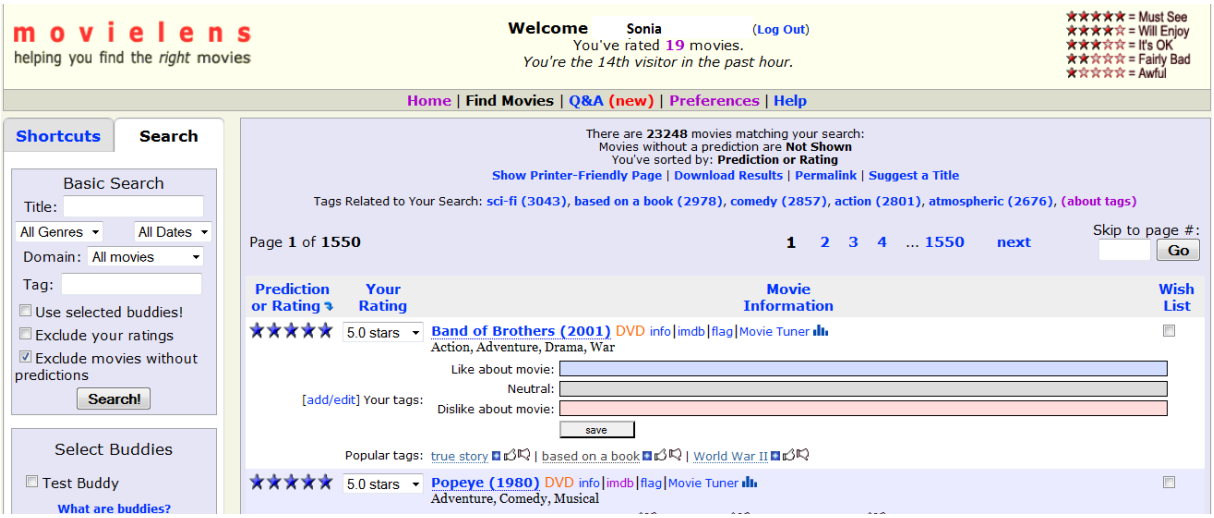


FIGURE 1.2 – Exemple de site (www.movielens.org) proposant aux clients d'attribuer des votes et/ou des annotations.

préférences. Dans le cas des systèmes de recommandations, le profil utilisateur est construit à partir des interactions antérieures entre le e-service et l'utilisateur courant. Plusieurs approches ont été définies pour représenter le profil de l'utilisateur, le mode de représentation est étroitement lié à la nature des données collectées par le e-service.

Données démographiques : les systèmes à filtrage démographique (systèmes de recommandation démographique) [Krulwich, 1997; Pazzani, 1999; Kobsa *et al.*, 2001] définissent le profil utilisateur à partir de stéréotypes. Ainsi, le profil utilisateur est constitué d'une liste de données démographiques telles que l'âge, le sexe, la profession, le niveau d'étude, le pays etc, décrivant le type de l'utilisateur. Les systèmes à filtrage démographique ne fournissent pas de recommandations individualisées, en effet les utilisateurs de même type auront les mêmes recommandations. Pour cette raison, les systèmes de recommandation démographiques ne sont pas considérés comme des systèmes de recommandation personnalisée mais des systèmes de recommandation de groupe [Masthoff, 2011].

Données sur le contenu des items : le contenu sémantique des items est essentiellement exploité par les systèmes de recommandation basée sur le contenu. Plusieurs approches ont été utilisées pour représenter le profil utilisateur à partir du contenu de l'item [Montaner *et al.*, 2003; Lops *et al.*, 2011]. Parmi lesquelles on peut citer le modèle vectoriel (VSM) dans lequel l'utilisateur est représenté par un vecteur de poids défini dans le même espace que celui représentant les items (voir section 1.1.1), chaque poids mesurant l'importance du terme correspondant pour l'utilisateur. Une description plus détaillée est donnée dans la section décrivant les techniques de recommandation basée sur le contenu (section 2.2.1 page 46).

Données sur les usages : la modélisation de l'utilisateur par les données issues de l'analyse des usages, et particulièrement les votes, est essentiellement utilisée par les systèmes de recommandation collaboratifs [Lousame and Sánchez, 2009]. Les utilisateurs sont modélisés par la matrice des votes contenant l'historique de leurs votes sur la totalité des items à recommander.

Ainsi, et comme nous l'avons déjà mentionné à la section 1.1.3, chaque élément de la matrice représente soit un vote soit une valeur manquante si l'évaluation correspondante n'existe pas. Chaque utilisateur est alors modélisé par un vecteur de la matrice, ainsi dans la table 1.2, l'utilisateur $user_1$ est modélisé par le vecteur $(4, 1, 2, ?)$ avec ? indiquant une valeur manquante.

1.1.5 Adaptation et mise à jour du profil utilisateur

Les préférences de l'utilisateur évoluent et changent avec le temps. Par exemple, après une naissance, une mère peut être intéressée par des produits dédiés aux nourrissons, mais son intérêt va progressivement changer vers d'autres produits au cours du temps. Ce qui signifie, que les goûts ne sont pas statiques et que les observations (représentées sous forme de retour d'expérience) les plus récentes, représentant les intérêts actuels de l'utilisateur, sont plus pertinentes que les anciennes observations. Par conséquent, il s'avère nécessaire, dans certains e-services, de disposer de techniques permettant la mise à jour des intérêts de l'utilisateur en conservant les préférences les plus récentes et en éliminant les plus anciennes. Plusieurs approches ont été définies parmi lesquelles on trouve :

Adaptation manuelle : l'utilisateur est invité à mettre à jour son propre profil si ses centres d'intérêts ont changé. En plus du fait qu'elle nécessite une implication directe de la part de l'utilisateur, cette solution est inadaptée lorsque les goûts de l'utilisateur changent fréquemment.

Adaptation par ajout d'information : c'est la méthode la plus utilisée dans les systèmes de recommandation [Montaner *et al.*, 2003]. La mise à jour du profil utilisateur est assurée par les retours d'expérience explicites, ou implicites recueillis par le système par analyse des usages. Comme nous l'avons déjà mentionné à la section 1.1.3, le retour d'expérience peut se présenter soit sous forme de vote explicite ou implicite ou sous forme d'annotations. Les systèmes tels que Amazon [Linden *et al.*, 2003], MovieLens [Miller *et al.*, 2003] et Tapestry [Goldberg *et al.*, 1992] utilisent les retours d'expérience comme moyen d'adaptation de leurs systèmes de recommandation. Le problème posé par cette méthode est qu'elle ne permet pas d'éliminer les anciennes préférences.

Fonction d'oubli progressif (Gradual forgetting function) : concept introduit par Webb et Kuzmyez [Webb and Kuzmycz, 1995] en 1995 inspiré du phénomène de l'oubli naturel défini comme étant un processus progressif. L'idée est d'attribuer un poids à chaque préférence en fonction de son ancienneté. Un poids initial égal à 1 est ainsi attribué à chaque observation (évaluation) qui voit sa valeur diminuer au cours du temps et à chaque introduction d'une nouvelle évaluation. De ce fait, les préférences les plus récentes auront plus d'*importance* que les anciennes qui finiront par disparaître au fil du temps. Koychev [Koychev, 2000] propose une fonction linéaire comme approximation de la fonction d'oubli progressif, mais elle peut être approximée par n'importe quelle autre fonction comme par exemple une fonction logarithmique ou exponentielle [Montaner *et al.*, 2003].

1.2 Recommandation

Dans cette section, nous résumons les principales fonctionnalités d'un système de recommandation (section 1.2.1), puis nous donnons une présentation formelle du problème de recomman-

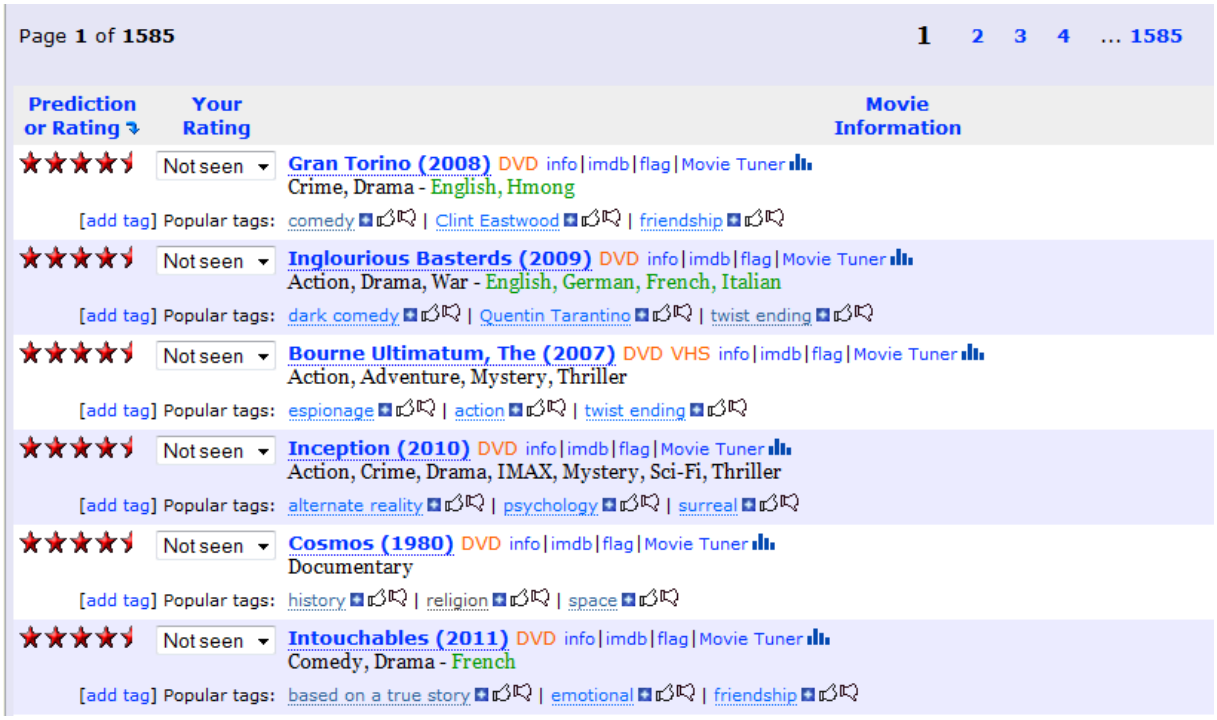


FIGURE 1.3 – MovieLens : système de recommandation collaborative avec prédiction des votes

dation (section 1.2.2). La description des trois techniques de recommandation fera l'objet du chapitre 2.

1.2.1 Les fonctionnalités d'un système de recommandation

Les fonctionnalités d'un système de recommandation sont très diversifiées et dépendent de la nature du domaine d'application. Nous présentons ci-dessous les grandes familles de fonctionnalités définies d'après [Schafer *et al.*, 2007] :

Prédiction : étant donné un item non évalué par l'utilisateur courant, le système prédit la valeur du vote que lui aurait attribué cet utilisateur. La prédiction est souvent utilisée dans les systèmes de recommandation collaborative. Ainsi, comme le montre la figure 1.3, le système de recommandation collaborative MovieLens [MovieLens, 2014] affiche à l'utilisateur courant les films avec leur prédiction. Les films ayant les plus grandes valeurs de prédiction sont affichés en tête de liste, ceux ayant les plus faibles prédictions sont en fin de liste. Le système affiche tous les films dont la prédiction a pu être calculée (on y voit 1585 pages avec 15 films par page soit un total de 23775 films).

Recommandation : affiche une liste d'items ordonnés selon les préférences de l'utilisateur courant. Si la recommandation est précédée par une étape de prédiction alors un Top-N des items ayant une plus grande valeur lui seront recommandés. Cependant, plusieurs systèmes de recommandation font de la recommandation sans passer par une étape de prédiction. C'est le cas de la majorité des systèmes de recommandation basés sur le contenu mais également de certains systèmes collaboratifs. Par exemple, le système de recommandation collaboratif de

Amazon [Linden *et al.*, 2003] calcule une agrégation sur les items similaires aux items achetés et évalués par l'utilisateur courant et affiche le Top-N des items similaires. Au lieu du vote prédit de l'item, Amazon [Amazon, 2014]affiche la moyenne des votes des utilisateurs pour chaque item recommandé (voir figure 1.1, page 16).

Recommandation avec contrainte : l'utilisateur courant définit un ensemble de contraintes sur les items candidats à la recommandation, limitant ainsi les items à l'ensemble des items vérifiant ces contraintes. Par exemple, supposons que Marie désire emmener sa nièce âgée de 10 ans voir un film au cinéma. Elle veut que le film ne contienne pas de scènes de violence, pour des enfants de moins de 12 ans et qu'il soit joué dans un cinéma proche de chez elle. Elle souhaite également choisir un film qui soit adapté à ses goûts. Le système doit lui recommander, parmi les films répondant à ses contraintes, ceux susceptibles de lui plaire. Les auteurs de [Schafer *et al.*, 2002] ont proposé un "*méta- système de recommandation*" qui génère des recommandations à partir d'un ensemble de sources de recommandations vérifiant les contraintes et préférences de l'utilisateur. Ce type de recommandation est appelé recommandation basée sur la connaissance Case-based [Smyth, 2007] ou Knowledge-based [Burke, 2002] selon les sources et est considéré comme faisant partie des systèmes de recommandation basés sur le contenu [Smyth, 2007]. Dans ce manuscrit, nous ne détaillerons pas ce type de système puisque nous nous situons dans un contexte à faible risque (voir section 1.1.1). En effet, les systèmes de recommandation Case-based supposent que l'utilisateur doit définir un ensemble de contraintes sur les items à recommander tels que les caractéristiques d'un appareil photo, ou les propriétés d'un ordinateur portable etc. Ceci suppose un investissement de la part de l'utilisateur et une complexité élevée quant à la description des items.

1.2.2 Formalisation du problème de la recommandation

Nous donnons dans cette section une définition formelle du problème de recommandation personnalisée. Pour ce faire, nous introduisons un ensemble de notations que nous utiliserons tout au long de ce manuscrit. Nous notons par U l'ensemble des utilisateurs du e-service, par I l'ensemble de tous les items pouvant être recommandés, par V l'ensemble des valeurs possibles pour un vote ($[1, 10]$, {aime, n'aime pas}) et par R un ensemble totalement ordonné (entiers ou réels appartenant à un intervalle fini). Par ailleurs, on suppose qu'un utilisateur u de U attribue au plus un seul vote $v_{ui} \in V$ à un item i de I. On note également par U_i le sous ensemble d'utilisateurs de U ayant évalué i et par I_u le sous ensemble d'items de I notés par l'utilisateur u. La fonction d'utilité déterminant l'utilité de l'item i pour l'utilisateur u est définie par $ut : (u, i) \in U \times I \longmapsto ut(u, i) \in R$, qui mesure l'utilité de l'item i pour l'utilisateur u. Les auteurs de [Adomavicius and Tuzhilin, 2005] définissent le problème de recommandation comme suit :

Pour tout utilisateur u de U, on recherche un item $i' \in I \setminus I_u$ inconnu de u maximisant sa fonction d'utilité. Plus formellement :

$$\forall u \in U, \quad i'_u = \underset{i \in I \setminus I_u}{arg\,max} \left(ut(u, i)\right) \tag{1.1}$$

Plusieurs approches ont été définies pour calculer l'utilité des items non évalués par l'utilisateur courant [Montaner *et al.*, 2003], soit en utilisant différentes heuristiques [Desrosiers and Karypis, 2011], soit en utilisant des méthodes issues de l'apprentissage automatique [Breese *et*

al., 1998; Salakhutdinov *et al.*, 2007] ou de la théorie d'approximation [Koren and Bell, 2011].

Lorsque le système dispose de votes, l'utilité $ut(u, i)$ est généralement égale au vote v_{ui}. Or, comme nous l'avons déjà dit, le taux de valeurs manquantes est très élevé (pouvant dépasser 95% dans certains cas). Ainsi, la fonction d'utilité de l'utilisateur u n'est pas définie pour la totalité de l'ensemble I, mais seulement pour les items de I_u. Le problème de la recommandation consiste alors à extrapoler la fonction d'utilité de u à tout l'ensemble I. Ce qui revient alors à définir une fonction de prédiction $pred : (u, i) \in U \times I \setminus I_u \longmapsto pred(u, i) \in R$ qui a pour objectif de prédire le vote $pred(u, i)$ de l'utilisateur $u \in U$ pour un item $i \in I \setminus I_u$. Une fois la fonction de prédiction définie, la formule 1.1 consiste alors à recommander l'item i' ayant la plus grande valeur de prédiction. La prédiction des votes est très utilisée dans les systèmes de recommandation collaborative. L'évaluation d'un tel système de recommandation est effectuée en mesurant la pertinence des prédictions en calculant soit *l'erreur absolue moyenne (Mean Absolute Error, MAE)* (voir formule (1.2)), soit *la racine de l'erreur absolue moyenne (Root Mean Absolute Error RMSE)* (voir formule (1.3)). Une description détaillée de ces mesures est donnée à la section 1.3.3.1.

Lorsque la fonction d'utilité ne prédit pas la valeur d'un vote, c'est le cas par exemple de la majorité des systèmes de recommandation basée sur le contenu, ou les systèmes ne disposant pas de votes (par exemple si seule la liste des produits achetés par l'utilisateur est disponible, ou la fréquence de consultation des pages web [Mobasher *et al.*, 2004]), le problème de recommandation consiste à recommander à l'utilisateur u une liste $L(u)$ contenant les N items les plus adaptés à ses goûts, on parle alors de recommandation Top-N [Deshpande and Karypis, 2004]. La performance d'un tel système est généralement mesurée en utilisant des mesures utilisées dans le domaine de la recherche et du filtrage d'information (voir section 1.3.3.2).

1.3 Évaluation

Lorsqu'un système de recommandation est développé, il est important d'être en mesure d'évaluer son fonctionnement et sa capacité à répondre aux objectifs qui lui ont été fixés. Un algorithme de recommandation a pour objectif d'améliorer l'utilité d'un e-service vis-à-vis de ses utilisateurs en augmentant leur satisfaction. Ainsi, mesurer la satisfaction des utilisateurs en terme de recommandation représente un critère d'évaluation important pour tout algorithme de recommandation. Cependant, pour effectuer de telles mesures, il est indispensable de tester le système en situation réelle en présence de vrais utilisateurs, ce type d'évaluation est appelée *évaluation en ligne* [Ekstrand *et al.*, 2011]. L'évaluation en ligne peut se faire également en situation virtuelle en faisant appel à un groupe d'utilisateurs pour tester le système et récupérer leurs impressions. L'évaluation en ligne peut être très coûteuse et parfois même impossible, une alternative plus simple, moins coûteuse et plus répandue dans l'évaluation des systèmes de recommandation est *l'évaluation offline*. Elle consiste à tester le système sur des données offline à la place de données en ligne. Plusieurs algorithmes d'évaluation offline ont été proposés dans la littérature [Herlocker *et al.*, 2004]. En plus du fait qu'elle soit plus simple à mettre en œuvre, l'évaluation offline permet également de comparer les performances de plusieurs algorithmes de recommandation de façon objective. Ainsi, il peut être intéressant de tester différentes méthodes de recommandation en offline, puis d'évaluer en ligne la meilleure d'entre elles. Nous nous limitons dans cette section à présenter les évaluations offline puisqu'elles sont les plus utilisées pour la comparaison entre les algorithmes de recommandation dans la littérature. Une étude détaillée sur les différents modes

d'évaluations ainsi que les algorithmes utilisés dans chaque mode est donnée dans [Shani and Gunawardana, 2011].

Dans ce qui suit nous présentons les principaux jeux de données utilisés pour l'évaluation offline des systèmes de recommandation (section 1.3.1), puis le principe de fonctionnement de ce mode d'évaluation (section 1.3.2), suivi de la description des principales mesures pour évaluer la précision des recommandations offline (section 1.3.3) et enfin la description des principales mesures de performance définies dans la littérature (section 1.3.4).

1.3.1 Les principaux jeux de données

Pour évaluer un algorithme de recommandation, il est nécessaire de disposer d'un jeu de données contenant les votes des utilisateurs. Les données peuvent être issues d'un système de recommandation existant ou synthétisés à partir d'algorithmes de simulation. Cependant, évaluer à partir de données simulées peut être risqué puisque les données peuvent être trop adaptées à un algorithme plus qu'aux autres algorithmes [Aggarwal *et al.*, 1999; Herlocker *et al.*, 2004]. A cet effet, plusieurs jeux de données issus de systèmes de recommandation existant ont été rendus publics et mis à la disposition des chercheurs pour comparer les performances de leurs algorithmes de recommandation.

MovieLens data set : issu du système de recommandation de films MovieLens [14], et mis à la disposition de la communauté des chercheurs par le groupe de recherche GroupLens [15]. Il est constitué de trois jeux de données [JeuMovieLens, 2014], Movielens100k, MovieLens1M et MovieLens10M. Movielens100k contient 100k de votes, il est différent des deux jeux MovieLens1M et MovieLens10M. MovieLens1M contient 1M de votes et est inclus dans le jeu MovieLens10M qui en contient 10M et tous les votes sont horodatés. Certaines informations sur les films sont données telles que le genre du film. Un lien vers une description détaillée de chaque film dans la base de données des films IMDB [16] est également donné, ainsi, il est possible d'extraire tous les descripteurs (acteurs, réalisateurs, synopsis, mots clés, date de réalisation, etc) à partir de la base IMDB [17]. Les votes sont explicites sur une échelle de 1 à 5 avec une granularité de 1 pour les jeux 100k et 1M et une granularité de 0.5 pour le jeu 10M.

Jester data set : est un jeu de données contenant les votes des utilisateurs collectés à partir du système de recommandation de blagues [18]. Il est composé de deux jeux de données collectés à des dates différentes. La particularité de ce jeu de données est que le nombres d'utilisateurs, de l'ordre de 70000, est très supérieur au nombre de blagues qui est égal à 100.

BookCrossing data set : est un jeu de données de livres collectés à partir du système de recommandation de livres BookCrossing.com [19]. En plus des votes, il contient des données démographiques sur les utilisateurs et quelques données sur la description des livres. Les votes sont explicites, définis sur une échelle de 1 à 10, et implicites.

14. http ://www.movielens.org
15. http ://www.grouplens.org/
16. http ://www.imdb.com
17. description des films à partir de IMDB :http ://www.imdb.com/interfaces
18. Le jeu de données de Jester est disponible : http ://eigentaste.berkeley.edu/dataset/.
19. jeu de données de BookCrossing disponible sur http ://www2.informatik.uni-freiburg.de/ cziegler/BX/

Netflix data set : issu du système de recommandation de films de Netflix [20], a été utilisé en 2006 pour le challenge lancé par Netflix [Bennett and Lanning, 2007]. Il est constitué de plus de 100M de votes horodatés de 17k de films effectués par 480k d'utilisateurs. Le jeu de test a été retiré en 2009 après un problème lié à la protection de la vie privée des utilisateurs [Narayanan and Shmatikov, 2008].

HetRec2011 data set : construit [21] à partir du jeu de données MovieLens 10M en y ajoutant des descriptions sur les films issues de la base de données de films IMDB et la base de critiques des films RottenTomatoes [22] [HetRec2011, 2011].

1.3.2 Évaluation en offline

Le principe de fonctionnement de l'évaluation offline est inspiré de l'approche utilisée dans le domaine de la fouille de données. Le jeu de données initial est partagé en deux parties distinctes, l'une dédiée à l'apprentissage qu'on appellera *jeu apprentissage* et l'autre à l'évaluation qu'on appellera *jeu test*. Dans le cas des systèmes de recommandation, le jeu de données est essentiellement constitué de l'historique des votes des utilisateurs. Ainsi, le *jeu apprentissage* est constitué d'un pourcentage des votes de chaque utilisateur et le *jeu test* du reste des votes. Dans le jeu de données de MovieLens 100k, le partage entre apprentissage et test est fait selon la proportion 80/20, c'est-à-dire que 80% du jeu de données est utilisé pour l'apprentissage et 20% pour le test. Ainsi, pour chaque utilisateur, 80% de ses votes font partie du jeu apprentissage et 20% de ses votes font partie du jeu test. Une fois le modèle de recommandation construit à partir du jeu apprentissage, le modèle est sollicité pour prédire pour chaque utilisateur, l'utilité (vote ou pertinence) des items présents dans le *jeu test*. L'évaluation consiste alors, à comparer les recommandations proposées par l'algorithme aux votes réels se trouvant dans le jeu de test.

Pour s'assurer que le résultat de l'évaluation est indépendant du partage apprentissage-test, il est possible de répéter le processus n fois, le résultat final est la moyenne des n résultats, ce mode est appelé *validation croisée* (cross validation). Pour un rapport 80/20 le processus est répété 5 fois, ainsi, le jeu initial est trié aléatoirement et divisé en 5 parties représentant chacune 20% du jeu initial, à chaque itération on choisit une partie comme jeu de test et le reste est utilisé pour le jeu apprentissage. Par ailleurs, si on dispose de votes horodatés, il est possible de procéder autrement. Pour chaque utilisateur, on trie ses votes selon leur ordre chronologique, les 80% premiers votes feront partie du jeu apprentissage et le reste ajouté au jeu test. Le jeu de données MovieLens 100k fournit les deux possibilités, 5 paires de jeux (80/20) issus du jeu initial trié aléatoirement pour une validation croisée et un jeu 80/20 trié par ordre chronologique.

1.3.3 La précision

La mesure de la précision d'un système de recommandation reste le critère dominant lors de la comparaison des performances de plusieurs systèmes de recommandation [Herlocker *et al.*, 2004; Shani and Gunawardana, 2011]. Le challenge de *Linked Open Data-Enabled Recommender System Challenge* [ESWC, 2014] lancé en décembre 2014 en est la meilleure preuve puisque les performances des algorithmes ont été évaluées essentiellement par rapport à leur précision.

20. www.netflix.com
21. jeu de donnée HetRec2011 est disponible sur http ://grouplens.org/datasets/hetrec-2011/
22. http ://www.rottentomatoes.com/

La précision mesure la capacité d'un système de recommandation à prédire des recommandations pertinentes pour ses utilisateurs. Il peut s'agir de mesurer le potentiel du système à prédire la valeur d'un vote, ou à prédire une liste de recommandations pertinentes. Plusieurs mesures ont été utilisées dans la littérature pour évaluer la précision [Breese *et al.*, 1998; Mobasher *et al.*, 2004; Deshpande and Karypis, 2004; Shani and Gunawardana, 2011]. Il n'existe pas à ce jour une standardisation des mesures à utiliser. Cependant, avant de faire un choix, il faut définir quelle précision on veut mesurer. Est-ce la précision de la prédiction des votes (section1.3.3.1) ? ou est-ce la précision de la prédiction d'une Top-N liste de recommandations (section 1.3.3.2) ? Dans cette section, nous présentons les principales mesures utilisées pour évaluer les systèmes de recommandation, une présentation plus complète se trouve dans [Herlocker *et al.*, 2004; Shani and Gunawardana, 2011].

1.3.3.1 Mesurer la précision de la prédiction des votes

La précision de la prédiction des votes ne peut être calculée que pour les algorithmes de recommandation qui prédisent la valeur du vote d'un item. On note par R l'ensemble des votes r_{ui} de tous les utilisateurs u de U sur les items de i de I. Pour évaluer un système de recommandation, on divise l'ensemble R en deux sous ensembles $R_{apprentissag}$ et R_{test} distincts, le premier est utilisé pour l'apprentissage de la fonction de prédiction $pred(u, i)$, le second pour évaluer la précision de la prédiction. On définit par T l'ensemble des couples (u, i) de R_{test} pour lesquels le système de recommandation a prédit la valeur du vote.

Erreur Absolue Moyenne (Mean Absolute Error MAE) mesure la moyenne des écarts entre un vote prédit et son correspondant dans R_{test}. L'erreur absolue moyenne (formule 1.2) a été utilisée pour évaluer plusieurs systèmes de recommandation [Sarwar *et al.*, 2001; Mobasher *et al.*, 2004; Resnick *et al.*, 1994] et est encore populaire de nos jours.

$$MAE = \frac{\sum_{(u,i)\in T} |pred(u,i) - v_{ui}|}{|T|} \tag{1.2}$$

Racine de l'Erreur Quadratique Moyenne : Root Mean Squared Error (RMSE) (formule 1.3) est devenue la mesure de référence surtout après son utilisation pour mesurer la précision dans le challenge Netflix [Bennett and Lanning, 2007] pour la recommandation de films. Comparée à la MAE, la RMSE pénalise les écarts importants, ainsi un écart de 2 points aura pour effet d'augmenter la somme des erreurs par 2 au niveau de la MAE, alors qu'il augmentera la somme par 4 au niveau de la RMSE.

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in T} (pred(u,i) - v_{ui})^2}{|T|}} \tag{1.3}$$

Plus la MAE ou la RMSE est faible, meilleure est la précision du système. En plus de ces deux mesures, on trouve la MAE normalisée (NMAE) et la RMSE normalisée (NRMSE) deux versions normalisées par l'intervalle des valeurs des votes ($v_{max} - v_{min}$) permettant ainsi de comparer des systèmes s'exécutant sur des jeux de données différents.

1.3.3.2 Mesurer la précision d'une Top-N liste

Plusieurs systèmes de recommandation ne calculent pas la prédiction de la valeur d'un vote soit, parce qu'ils ne disposent que de votes binaires *(aime, aime pas)* ou unaires (acheté, préféré, consulté) soit, parce que l'algorithme utilisé ne calcule pas de prédictions de votes. De tels systèmes recommandent plutôt à l'utilisateur courant une liste $L(u_a)$ contenant N items susceptibles de l'intéresser. On parle alors de recommandation Top-N. La mesure de la précision revient alors à mesurer la capacité du système à décider si un item est pertinent ou pas pour l'utilisateur courant. Si les votes sont numériques, il est primordial de définir un seuil de vote permettant de classer les items de la base de test R_{test} en items pertinents et items non pertinents. Ainsi, pour des votes définis sur une échelle de 1 à 5, on peut considérer que les items ayant un vote ≥ 4 sont pertinents pour l'utilisateur u et ceux ayant un vote < 4 sont non pertinents. La base $R_{apprentissage}$ est utilisée pour l'apprentissage de $L(u)$.

La précision, le rappel et leurs dérivées : la *précision* et le *rappel* sont deux mesures très populaires pour l'évaluation des systèmes de recherche d'information depuis leur proposition par Cleverdon en 1968 [Cleverdon and Kean, 1968]. Ils sont également largement utilisés pour l'évaluation des systèmes de recommandation proposant une Top-N liste [Billsus and Pazzani, 1998; Sarwar *et al.*, 2000b]. On définit par $I_{u_{test}}$ le sous ensemble d'items tests de I_u présents dans la base de test R_{test}. On note par $P(u) \subset I_{u_{test}}$ la liste des items de la base de test pertinents pour l'utilisateur u. Le rappel (resp. la précision) est calculé(e) pour chaque utilisateur u puis agrégé(e) sur tous les utilisateurs de U tel(le) que le définit la formule (1.4 (resp. 1.5).

$$Rappel = \frac{1}{|U|} \sum_{u \in U} \frac{|L(u) \cap P(u)|}{|P(u)|} \tag{1.4}$$

$$Précision = \frac{1}{|U|} \sum_{u \in U} \frac{|L(u) \cap P(u)|}{|L(u)|} \tag{1.5}$$

La précision mesure la proportion de recommandations réellement pertinentes pour l' utilisateur u parmi les items recommandés (dans $L(u)$). Le rappel mesure le taux de recommandations pertinentes parmi l'ensemble des items réellement pertinents pour u (dans $P(u)$). Plus le taux de précision et de rappel sont élevés, meilleure est la performance de l'algorithme de recommandation. Lors de la comparaison de plusieurs algorithmes, le rappel et la précision doivent être utilisés simultanément, en effet [Cleverdon and Kean, 1968] ont prouvé que le rappel et la précision sont inversement proportionnels et dépendent de la valeur de N (la taille de la liste), ainsi quand N augmente, la précision diminue et le rappel augmente et inversement. L'utilisation simultanée des deux mesures en faisant varier N peut compliquer la tâche d'évaluation. Une troisième mesure, la F1-mesure définie par la formule 1.6, combinant le rappel et la précision, est également largement utilisée dans le domaine de la recherche d'information. Elle a été introduite pour l'évaluation des systèmes de recommandation par Sarwar et al. [Sarwar *et al.*, 2000b; Sarwar *et al.*, 2000a] et a été utilisée pour évaluer la Top-N liste dans le challenge *Linked Open Data-Enabled Recommender System* [ESWC, 2014].

$$F1 = \frac{2 \times Précision \times Rappel}{Précision + Rappel} \tag{1.6}$$

Lors de l'utilisation de telles mesures pour évaluer un système de recommandation, on peut se trouver confronté à des difficultés liées au problème des données manquantes. En ef-

fet, ces deux mesures exigent que tous les items présents dans la liste $L(u)$ soient présents dans la base de test $I_{u_{test}}$, c'est-à-dire $L(u) \subset I_{u_{test}}$, afin de pouvoir les classer comme pertinents ($\in P(u)$) ou pas pour u. Or, et en raison des données manquantes, il est assez fréquent de se trouver dans le cas où des items de $L(u)$ ne soient pas présents dans $I_{u_{test}}$. Dans ce cas, le système ne possède aucune information lui permettant de décider s'ils sont pertinents ou pas pour u. Une des solutions proposée dans la littérature [Herlocker *et al.*, 2004; Shani and Gunawardana, 2011] est d'ignorer les items recommandés (dans $L(u)$) non présents dans la base de test $I_{u_{test}}$. Ainsi, la liste $L(u)$ sera composée des N items recommandés par le système et présents dans $I_{u_{test}}$.

Par ailleurs, si le nombre d'items à recommander à l'utilisateur est fixe (par exemple 5 recommandations), c'est-à-dire que le système ne recommande pas tous les items potentiellement pertinents pour l'utilisateur mais seulement un nombre N fixe, alors la mesure de la précision suffit pour évaluer le système [Herlocker *et al.*, 2004; Shani and Gunawardana, 2011]. Si au contraire, l'objectif est de trouver tous les items pertinents alors le rappel devient également important.

Taux de réussite : utilisé par Mobasher et al. [Mobasher *et al.*, 2004] pour évaluer une Top-N liste, pour chaque utilisateur u de U, un item pertinent $i_{u_{test}}$ de I_u est sélectionné aléatoirement. Le jeu de test est alors constitué de l'ensemble de ces items, le reste est utilisé pour l'apprentissage de $L(u)$. L'évaluation consiste alors à calculer le taux de réussite (Hit Ration) de la façon suivante : pour tout utilisateur u, si son item $i_{u_{test}}$ est présent dans la liste des recommandations $L(u)$ alors c'est une réussite. Le taux de réussite est alors défini par la somme des réussites divisée par le nombre d'utilisateurs. Plus le taux est élevé, meilleure est la performance du système.

Évaluer l'ordre des items dans une liste : le rappel, la précision et le taux de réussite ne mesurent pas la capacité d'un algorithme de recommandation à ordonner une liste de recommandations. De telles mesures traitent tous les items de la même façon indépendamment de leur position dans la liste $L(u)$. En 1998, Breese et al. [Breese *et al.*, 1998] propose la *Half-life Utility Metric (HUM)* qui évalue plus fortement le haut d'une liste ordonnée, le reste de la liste n'étant pas important pour l'utilisateur.

$$HUM(u) = \sum_{j \in [1,|L(u)|]} \frac{max(v_{uj} - d, 0)}{2^{(j-1)/(\alpha-1)}} \, avec \, \alpha \in [1, |L(u)|] \tag{1.7}$$

$$HUM = 100 \frac{\sum_{u \in U} HUM(u)}{\sum_{uinU} max(HUM(u))} \tag{1.8}$$

Avec j étant le rang du $j^{ème}$ item de la liste ordonnée $L(u)$, d est la note seuil au dessus de laquelle l'item est considéré comme pertinent, v_{uj} est la notée observée pour le $j^{ème}$ item de la liste et α la valeur divisant la liste en haut et bas de la liste. Si v_{uj} est une note manquante on lui attribue la valeur d. La fonction $\frac{1}{2^{(j-1)/(\alpha-1)}}$ est décroissante en fonction du rang j de l'item dans la liste, les items se trouvant après la $\alpha^{ième}$ position sont quasiment ignorés dans le mesure HUM.

Une autre mesure introduite par [Deshpande and Karypis, 2004], l'*Average Reciprocal Hit Rank (ARHR)*, mesure le taux de réussite en fonction du rang de l'item dans la liste. Comme pour la mesure du taux de réussite présentée ci-dessus, pour chaque utilisateur, un item pertinent

est sélectionné aléatoirement et placé dans le jeu de test, le reste est utilisé pour l'apprentissage.

$$ARHR = \frac{1}{|U|}\frac{1}{rang(i_u, L(u))} \tag{1.9}$$

Si i_u n'est pas présent dans la liste alors son rang est égal à ∞.

1.3.3.3 Quelle mesure utiliser ?

Plusieurs études [Herlocker *et al.*, 2004; Cremonesi *et al.*, 2010] ont montré qu'il peut y avoir une décorrélation entre les différentes mesures de précision. En effet, une étude menée par [Koren, 2008] sur les algorithmes participant au challenge Netflix en 2007 [Bennett and Lanning, 2007] a montré que des écarts relatifs inférieurs à 10% sur la RMSE peuvent mener à d'énormes écarts en terme d'évaluation Top-N (de l'ordre de 300% pour $N = 1$). Toutefois, la *MAE* et la *RMSE* depuis le challenge Netflix [Bennett and Lanning, 2007] sont les mesures de référence pour évaluer la précision de la prédiction des votes. Pour l'évaluation d'une Top-N liste, la *F1-mesure* ou la *précision* restent les mesures les plus fréquemment utilisées.

1.3.4 Au-delà de la précision

Bien que la majorité des travaux de recherche évaluent leurs algorithmes par rapport à la précision, d'autres critères d'évaluation ont également été définis et explorés pour évaluer un système de recommandation.

La diversité : est généralement définie comme l'opposée de la similarité, elle mesure la capacité du système à offrir une liste de recommandations diversifiées. Elle est essentiellement utilisée pour évaluer les systèmes de recommandation basés sur le contenu. La diversité est appliquée pour évaluer une liste Top-N de recommandations en se basant sur la similarité entre les items la composant, les items étant comparés selon leur contenu. Ainsi, il est possible de calculer la diversité d'une liste à partir de la somme, la moyenne, le maximum ou le minimum des distances entre les paires d'items la composant [Ziegler *et al.*, 2005]. La diversité peut être également utilisée en la couplant avec d'autres mesures telles que la précision [Zhang, 2009]. [Shani and Gunawardana, 2011] propose de représenter la courbe précision-diversité en faisant varier N, le système le plus performant correspond à la courbe dominante. La formule 1.10 représente la *Similarité Intra-Liste SIL* définie par les auteurs de [Ziegler *et al.*, 2005] pour mesurer la diversité des recommandations pour l'utilisateur u, plus la SIL est petite, meilleure est la diversité. Une autre mesure est la *Diversité Intra-Liste (DIL)* décrite par la formule 1.12 a été utilisée pour mesurer la diversité dans *Linked Open Data-Enabled Recommender System Challenge* [ESWC, 2014].

$$SIL(u) = \frac{1}{2}\sum_{i \in L(u)}\sum_{j \in L(u), i \neq j} sim(i,j) \tag{1.10}$$

$$SIL = \frac{1}{|U|}\sum_{u \in U} SIL(u) \tag{1.11}$$

$$DIL(u) = \frac{1}{2}\sum_{i \in L(u)}\sum_{j \in L(u), i \neq j} (1 - sim(i,j)) \tag{1.12}$$

$$DIL = \frac{1}{|U|} \sum_{u \in U} DIL(u) \tag{1.13}$$

La couverture : mesure la proportion d'items que l'algorithme peut recommander. Dans sa forme la plus simple, il s'agit de calculer le pourcentage de tous les items recommandés durant une expérimentation. Il peut être intéressant, pour certaines applications, de distinguer entre les items en les pondérant, par exemple par leur popularité ou par leur utilité. La couverture peut être également définie par rapport aux utilisateurs, en calculant la proportion d'utilisateurs pour lesquels le système peut recommander des items.

La nouveauté : mesure la capacité d'un système à recommander des items totalement inconnus de l'utilisateur courant et assez différents de ses items préférés. Ce critère intéresse essentiellement la recommandation collaborative puisque les algorithmes basés sur le contenu, de part leur principe de fonctionnement, ne peuvent pas recommander de tels items. Cependant, la nouveauté est assez difficile à mesurer puisqu'elle ne peut se faire que pour une évaluation en ligne en présence de vrais utilisateurs pour indiquer si une recommandation leur est surprenante ou pas.

Le démarrage à froid : mesure la capacité du système à recommander des items aux nouveaux utilisateurs (démarrage à froid pour les utilisateurs), ou à recommander des items nouvellement ajoutés à la base (démarrage à froid pour les items). Pour le démarrage à froid pour les utilisateurs, il peut s'agir de mesurer le nombre de votes nécessaires à partir duquel l'utilisateur peut recevoir des recommandations pertinentes.

Le passage à l'échelle Les algorithmes peuvent être évalués par rapport à leur complexité en temps de calcul ou en occupation mémoire.

1.4 Conclusion

Un système de recommandation a pour objectif de proposer à l'utilisateur des items susceptibles de l'intéresser parmi un large choix de possibilités. Plusieurs techniques de recommandation existent dans la littérature [Ricci *et al.*, 2011b]. Parmi ces techniques, on trouve celles proposant des recommandations personnalisées en définissant un profil personnalisé pour chaque utilisateur. Dans ce travail, nous nous intéressons aux systèmes de recommandation personnalisée dans lesquels le profil utilisateur est construit à partir de l'analyse des usages. Les systèmes de recommandation basés sur le contenu, les systèmes de recommandation collaborative et les systèmes hybrides combinant les deux techniques comptent parmi les techniques les plus utilisées dans la recommandation personnalisée [Adomavicius and Tuzhilin, 2005]. La description de ces trois techniques, leur principe de fonctionnement et leurs principaux algorithmes seront décrits dans le chapitre 2.

Par ailleurs, nous rappelons que nous nous intéressons dans ce travail aux applications évoluant dans un contexte à faible risque, c'est-à-dire à faible utilité pour l'utilisateur et dont les items ne nécessitent pas de description complexe. C'est le cas par exemple de la location de DVDs, de l'achat de livres, de la visualisation de film ou du choix d'un restaurant. Ce n'est pas le cas par exemple, des applications dédiées à la vente d'automobiles, aux transactions financières, à

l'achat d'ordinateur, à l'achat d'appareil photo ou à l'achat de voyages. Dans ce contexte, aucune information n'est disponible sur l'utilisateur à part un numéro permettant de l'identifier et une liste de votes attribués explicitement ou implicitement aux items qu'il a consultés. En ce qui concerne les items, on suppose disposer d'information textuelles décrivant le contenu des items. Cette information peut être structurée ou non structurée.

Chapitre 2

Les techniques de recommandations

Les systèmes de recommandation personnalisée ont été classés en trois catégories [Balabanović and Shoham, 1997; Adomavicius and Tuzhilin, 2005] en fonctions de la nature des données utilisées et de la manière dont est effectué le filtrage des items :

- **Les systèmes de recommandation collaborative** : le filtrage des recommandations est réalisé de manière collaborative. Ainsi, le système recommandera à l'utilisateur courant les items appréciés par les utilisateurs avec lesquels il partage des goûts et des préférences similaires. Dans ce type de filtrage, seules des données issues de l'analyse des usages sont utilisées.
- **Les systèmes de recommandation basée sur le contenu** : le filtrage est basé sur le contenu des items. Le système recommandera à l'utilisateur courant les items similaires à ceux qu'il a appréciés par le passé. En plus des données issues de l'analyse des usages, le système utilise la description textuelle des items pour faire de la recommandation.
- **Les systèmes de recommandation hybride** : combinent les méthodes collaboratives et les méthodes basées sur le contenu pour faire de la recommandation.

Ce chapitre est consacré à la présentation des trois techniques de recommandation. L' origine et les principaux algorithmes de filtrage collaboratif feront l'objet de la section 2.1. La section 2.2 sera consacrée à la présentation des principaux algorithmes de filtrage basé sur le contenu. Enfin, la section 2.3 résumera les différentes techniques d'hybridation entre le filtrage collaboratif et le filtrage basé sur le contenu.

2.1 Le filtrage collaboratif

Le filtrage collaboratif est une technique basée sur le partage d'opinions entre les utilisateurs. Bien que le terme n'ait été introduit que depuis moins de deux décennies, il implémente le principe du *"bouche à oreille"* pratiqué depuis toujours par les humains pour se construire une opinion sur un produit ou un service [Schafer *et al.*, 2007]. Considérons par exemple que les collègues de Marie trouvent que le dernier film sorti en salle est un succès, elle peut juger intéressant d'aller le voir. Si au contraire, la majorité de ses collègues estiment que c'est un échec, alors il se peut qu'elle décide de s'abstenir d'y aller. Mieux encore, si Marie estime qu'elle a toujours apprécié les films recommandés par Sami, que les films recommandés par Sara l'ont toujours déçus et que Alice recommande tous les films sans aucune distinction, au fil du temps, elle va finir par apprendre quelles opinions considérer lors de la détermination de son choix.

En exploitant le web et la puissance des ordinateurs, il devient possible pour une personne de partager ses opinions avec des milliers voire des millions d'utilisateurs au lieu de se contenter de l'avis d'une dizaine de personnes. C'est ce que fait le filtrage collaboratif en recommandant à l'utilisateur courant les items appréciés par les utilisateurs du e-service avec lesquels il partage les mêmes goûts.

Les débuts du filtrage collaboratif

Le système Tapestry [Goldberg *et al.*, 1992] développé par Xerox PARC est le premier système à utiliser l'opinion de ses utilisateurs pour le filtrage de messages électroniques. En plus du contenu du message, son auteur et ses lecteurs, Tapestry stocke les avis des utilisateurs sous forme d'annotations (exemple "intéressant", "excellent"). L'utilisateur a la possibilité de définir son propre filtre en combinant des mots clés associés au contenu (par exemple "système de recommandation") avec des mots clés issus des annotations (annoté "excellent" par "Jean"). Le terme *collaboratif*, alors introduit pour la première fois, signifie que les utilisateurs collaborent entre eux pour s'entraider à filtrer leurs messages. Ce modèle de filtrage est appelé *"pull-active-collaborative filtering"* [Schafer *et al.*, 2007] parce que c'est à la charge de l'utilisateur de tirer vers lui les recommandations. Après l'émergence du système Tapestry, les chercheurs se sont intéressés au potentiel que peut apporter le partage des opinions des utilisateurs dans un système de filtrage d'information au sein d'une organisation. Maltz et Ehrlich [Maltz and Ehrlich, 1995] ont développé un *"push-active collaborative filtering"* permettant à un employé de *"pousser"* un document vers les collègues intéressés par son contenu. Ce type de recommandation s'est largement popularisé par la suite, essentiellement sur les réseaux sociaux, Facebook [23] en est le plus grand exemple. Chaque utilisateur a la possibilité d'envoyer (partager selon le langage de Facebook) vers un groupe d'amis les documents susceptibles de les intéresser. Dans ces deux exemples, il s'agit d'un *système de filtrage collaboratif actif* qui exige pour son fonctionnement que les utilisateurs du système se connaissent pour partager leurs opinions. En effet, dans le système "*pull active*" l'utilisateur doit savoir à quelles opinions faire confiance, dans le système "*push active*", l'utilisateur doit connaître les utilisateurs susceptibles d'apprécier l'item à partager. Dans les systèmes de filtrage collaboratif automatisé, cette contrainte est supprimée et les groupes d'utilisateurs partageant les mêmes goûts sont formés automatiquement à partir d'une base de données contenant l'historique des préférences de tous les utilisateurs du système.

Parmi les premiers systèmes de filtrage collaboratif automatisé, on trouve GroupLens [Resnick *et al.*, 1994; Konstan *et al.*, 1997] dans le domaine des forums de discussions d'articles Usenet, Ringo [Shardanand and Maes, 1995] pour la recommandation d'albums de musique et d'artistes, et Bellcore's Video Recommender [Hill *et al.*, 1995] pour la recommandation de films. Dans ce manuscrit, l'expression *filtrage collaboratif* est utilisée pour désigner le filtrage collaboratif automatisé, le filtrage collaboratif actif ne fait pas l'objet de ce travail et ne sera pas détaillé au niveau de ce manuscrit.

Principaux algorithmes du filtrage collaboratif

Les auteurs de [Breese *et al.*, 1998; Su and Khoshgoftaar, 2009] ont regroupé les méthodes de filtrage collaboratif en deux classes : *les algorithmes basés sur la mémoire* et *les algorithmes basés sur un modèle*. Les algorithmes de filtrage collaboratif basés sur la mémoire, appelés

23. www.facebook.com

également *basés sur des heuristiques* [Adomavicius and Tuzhilin, 2005] ou basés sur les voisins [Desrosiers and Karypis, 2011] utilisent les votes des utilisateurs stockés en mémoire pour faire de la prédiction. Les algorithmes basés sur un modèle construisent en offline une image réduite de la matrice des votes dans un objectif de réduire la complexité des calculs et/ou de traiter le problème des données manquantes. Le modèle passe par une étape d'apprentissage, puis, il est utilisé pour faire de la recommandation. Plusieurs approches ont été utilisées dans les algorithmes de recommandation basés sur un modèle parmi lesquelles on peut citer les méthodes de réduction de la dimension basées sur l'algèbre linéaire appelées également SVD-ModelBased [Koren and Bell, 2011], les approches probabilistes [Breese *et al.*, 1998; Hofmann, 2003], les approches basées sur le clustering [Sorensen and McElligott, 1998], et les approches basées sur les règles d'association [Heckerman *et al.*, 2001].

La section 2.1.1 présente le principe de fonctionnement des algorithmes de filtrage collaboratif basés sur les voisins. Les points forts et les défis de ces algorithmes sont respectivement présentés aux sections 2.1.2 et 2.1.3. Un aperçu sur les méthodes de réduction de la dimension provenant de la théorie d'algèbre linéaire est décrit dans la section 2.1.4. La section 2.1.5 résume les approches probabilistes.

2.1.1 Filtrage collaboratif basé sur les voisins

Les algorithmes de filtrage collaboratif basés sur les voisins [Nakamura, 1998; Konstan *et al.*, 1997; Hill *et al.*, 1995; Delgado and Ishii, 1999] utilisent la totalité ou une partie de la matrice des votes des utilisateurs pour faire de la recommandation. Ils sont scindés en deux groupes : *basés sur les utilisateurs (user-based)* ou *basés sur les items (item-based)*. Pour les algorithmes basés sur les utilisateurs tels que GroupLens [Resnick *et al.*, 1994] ou Ringo [Shardanand and Maes, 1995], l'intérêt potentiel d'un utilisateur u pour un item i est prédit en utilisant les votes de ses *voisins* (utilisateurs avec lesquels il partage les mêmes goûts). Les algorithmes basés sur les items (item-based) [Sarwar *et al.*, 2001; Linden *et al.*, 2003; Deshpande and Karypis, 2004] déterminent l'intérêt d'un utilisateur u pour un item candidat à la recommandation i à partir des votes de l'utilisateur u pour les items voisins de i. Le point commun entre ces deux approches est la détermination des voisins de l'utilisateur u (dans le cas des algorithmes basés sur les utilisateurs) ou la détermination des voisins de l'item i (pour les algorithmes basés sur les items). La détermination des voisins, d'un item ou d'un utilisateur, se fait en mesurant la similarité de ce dernier avec les candidats potentiels.

Dans ce qui suit, nous présentons chacune des deux approches : basée sur les utilisateurs (sections 2.1.1.1 et 2.1.1.2) et basée sur les items (sections 2.1.1.3 et 2.1.1.4). Les principales méthodes utilisées par la recommandation collaborative pour le calcul de la similarité entre deux objets (items ou utilisateurs) fait l'objet de la section 2.1.1.5. Enfin, nous présentons les points forts (section 2.1.2) et les défis des algorithmes de filtrage collaboratif basé sur les voisins (section 2.1.3).

En plus des notations définies dans la section 1.2.2, on note par $\overline{v_u}$ la moyenne des votes de l'utilisateur u sur les items qu'il a notés (formule 2.1) et par $\overline{v_i}$ la moyenne des votes de l'item i (formule 2.2).

$$\overline{v_u} = \frac{\sum_{i \in I_u} v_{ui}}{|I_u|}. \tag{2.1}$$

.

$$\overline{v_i} = \frac{\sum_{u \in U_i} v_{ui}}{|U_i|}. \tag{2.2}$$

.

On note également par $sim(u, w)$ la fonction mesurant la similarité entre les deux utilisateurs u et v (resp. $sim(i, j)$ celle mesurant la similarité entre les deux items i et j). On définit $I_{uw} = I_u \cap I_w$ l'ensemble des items notés à la fois par les utilisateurs u et w, et de façon équivalente $U_{ij} = U_i \cap U_j$ l'ensemble des utilisateurs ayant notés à la fois les items i et j. Enfin, on note par $voisins(u)$ l'ensemble des utilisateurs $w \in U$ définis comme les voisins de l'utilisateur u.

2.1.1.1 Prédiction basée sur les utilisateurs

La prédiction basée sur les utilisateurs (user-based) a été introduite pour la première fois dans le système GroupLens [Resnick *et al.*, 1994], son principe de fonctionnement se résume ainsi :

1. Déterminer les voisins de l'utilisateur courant en calculant sa similarité avec les autres utilisateurs de la base.
2. Calculer la prédiction du vote de l'utilisateur courant u_a pour un item $i \in I \setminus I_{u_a}$ candidat à la recommandation, en analysant les votes de ses voisins sur ce même item.

Calcul de la similarité : la similarité entre deux utilisateurs u et w peut être mesurée en utilisant, soit le coefficient de corrélation de Pearson (formule (2.11)), soit le Cosinus (formule (2.9)). Il a été cependant démontré que le coefficient de Pearson reste le plus utilisé et le plus performant en terme de pertinence de prédiction [Schafer *et al.*, 2007].

Calcul de la prédiction : pour le calcul de la prédiction, le plus naïf est de calculer la moyenne des votes de tous les voisins de l'utilisateur courant u_a comme l'illustre l'équation 2.3.

$$pred(u_a, i) = \frac{\sum_{u \in voisins(u_a) \cap U_i} v_{ui}}{|voisins(u_a) \cap U_i|} \tag{2.3}$$

La formule donnée en 2.3 est dite naïve parce qu'elle considère tous les voisins au même pied d'égalité et ne tient pas compte du fait que certains voisins soient plus proches que d'autres de l'utilisateur courant u_a [Schafer *et al.*, 2007]. Afin de tenir compte de cette diversité, dans la somme de la formule 2.3, on pondère le vote de chaque voisin par la valeur de sa similarité avec l'utilisateur courant. Ainsi, les votes des plus proches voisins auront un poids plus important que celui des voisins les moins proches. Vu que la somme des poids de tous les voisins n'est pas égale à 1, et afin de normaliser la valeur de la prédiction, cette somme est divisée par la somme des similarités de l'utilisateur courant avec ses voisins. L'équation 2.4 donne la formule correspondante pour le calcul de la prédiction.

$$pred(u_a, i) = \frac{\sum_{u \in voisins(u_a) \cap U_i} sim(u_a, u) v_{ui}}{\sum_{u \in voisins(u_a) \cap U_i} |sim(u_a, u)|} \tag{2.4}$$

Par ailleurs, tous les utilisateurs sont différents dans leurs façons d'évaluer un item. En effet, il existe des utilisateurs qui notent large en affectant la valeur de 5 sur une échelle de 1 à 5 pour un item qu'ils jugent satisfaisant alors que d'autres, qui ont tendance à noter de façon plus stricte, attribueront la valeur 3 à un item qu'ils jugent satisfaisant. Pour compenser la variation

dans le jugement des utilisateurs, le vote de chaque utilisateur u est ajusté par la moyenne de ses votes $\overline{v_u}$. L'équation 2.5 donne la formule adoptée par les auteurs de [Resnick *et al.*, 1994] pour le calcul de la prédiction.

$$pred(u_a, i) = \overline{v_{u_a}} + \frac{\sum_{u \in voisins(u_a) \cap U_i} sim(u_a, u)(v_{ui} - \overline{v_u})}{\sum_{u \in voisins(u_a) \cap U_i} |sim(u_a, u)|} \tag{2.5}$$

Dans l'algorithme initial tel que implanté dans le GroupLens système [Resnick *et al.*, 1994], tous les voisins sont pris en compte dans le calcul de la prédiction. Il a été démontré par la suite que la prise en compte des k plus proches voisins améliore, non seulement la pertinence des prédictions, mais également l'efficacité de l'algorithme [Herlocker *et al.*, 1999].

2.1.1.2 Recommandation Top-N basée sur les utilisateurs

Les algorithmes de recommandation Top-N sont généralement utilisés lorsque le système ne dispose pas de votes numériques définis sur une échelle de valeur mais de votes binaires (*aime*, *n'aime pas*) ou unaire (*a acheté*, *a consulté*). Comme pour la prédiction basée sur les utilisateurs, une première étape consiste à définir les plus proches voisins de l'utilisateur courant en utilisant soit le coefficient de Pearson soit le Cosinus. Une fois les plus proches voisins de l'utilisateur courant u_a identifiés, une deuxième étape consiste à déterminer pour chaque voisin w, la liste L_w des items pertinents (items achetés). Les items sont par la suite triés selon la fréquence de leur présence dans les listes de tous les voisins. Le Top-N des items les plus fréquents seront alors recommandés à u_a [Sarwar *et al.*, 2000a]. Les algorithmes de recommandation Top-N basés sur les utilisateurs souffrent du problème de passage à l'échelle et du problème de performance pour les applications en temps réel [Karypis, 2001].

2.1.1.3 Prédiction basée sur les items

Introduite par Sarwar et al. [Sarwar *et al.*, 2001], la prédiction du vote de l'utilisateur u pour un item candidat $i \in I \setminus I_u$ est calculée à partir de ses évaluations pour les items voisins de i. Son principe de fonctionnement est le suivant :

1. Pour tout item i de I, déterminer les voisins les plus proches en calculant sa similarité avec les autres items.
2. Calculer la prédiction du vote de l'utilisateur courant u_a pour l' item i en analysant les votes de ce dernier sur les voisins de i.

Détermination des voisins : la similarité entre deux items i et j peut être calculée en utilisant soit le *Cosinus* (voir formule (2.10)), soit *le coefficient de Pearson* (voir formule (2.12)), soit le *cosinus ajusté* (voir formule (2.13)). Cependant, une étude expérimentale menée par les auteurs de [Sarwar *et al.*, 2001], comparant les trois mesures, a montré que le *Cosinus ajusté* reste le plus performant en terme de pertinence de prédiction.

Calcul de la prédiction : la prédiction du vote de l'utilisateur courant u_a pour un item candidat à la recommandation i revient à calculer une moyenne pondérée de ses votes sur l'ensemble des items similaires à i. Chaque vote $v_{u_a,j}$ est pondéré par la similarité de l'item j avec l'item i. Afin d'avoir une prédiction dans le même intervalle de valeurs que les votes, la prédiction est divisée par la somme des similarités. L'ajustement du vote est inutile dans ce cas puisqu'il s'agit

du même utilisateur. L'équation 2.6 donne la formule de calcul de la prédiction telle que donnée dans [Sarwar *et al.*, 2001].

$$pred(u_a, i) = \frac{\sum_{j \in I_{u_a}} sim(i,j) v_{uj}}{\sum_{j \in I_{u_a}} |sim(i,j)|} \tag{2.6}$$

Les auteurs [Sarwar *et al.*, 2001] ont démontré que la pertinence des prédictions est très sensible au nombre de voisins considérés dans la formule 2.6. Ainsi, parmi les items notés par u_a, seuls les k plus proches voisins $voisins_k(i)$ de i, sont pris en compte comme le définit la formule 2.7. Les expérimentations qu'ils ont menées sur les données du système de recommandation MovieLens [JeuMovieLens, 2014] ont montré que le seuil de la prédiction est atteint pour les 30 plus proches voisins.

$$pred(u_a, i) = \frac{\sum_{j \in voisins_k(i) \cap I_{u_a}} sim(i,j) v_{uj}}{\sum_{j \in voisins_k(i) \cap I_{u_a}} |sim(i,j)|} \tag{2.7}$$

Les algorithmes basés sur les items sont moins sensibles aux données manquantes et plus performants en terme d'efficacité (complexité de calcul) que les algorithmes basés sur les utilisateurs. Cependant, et malgré leur lenteur, les expérimentations ont montré que les algorithmes basés sur les utilisateurs sont plus performants en terme de précision [Deshpande and Karypis, 2004].

2.1.1.4 Recommandation Top-N basée sur les items

Le système de recommandation de Amazon utilise un algorithme de recommandation Top-N basé sur les items [Linden *et al.*, 2003]. Après une opération d'achat, le système affiche à l'utilisateur un message de la forme "*les clients ayant acheté ce produit ont également consulté ces produits*" suivi d'une liste de recommandations. Les algorithmes Top-N basés sur les items ont été développés pour contrecarrer les problèmes de performance et de passage à l'échelle dont souffraient les algorithmes Top-N basés sur les utilisateurs [Karypis, 2001] présentés dans la section 2.1.1.2. Le principe de fonctionnement [Deshpande and Karypis, 2004] de tels algorithmes suit deux étapes. Une première étape consiste à calculer les voisins de chaque item de I. Une fois les k plus proches voisins, $voisins_k(j)$ de chaque item j identifiés, une deuxième étape consiste à construire la liste des items à recommander à l'utilisateur courant u_a. Soit I_{u_a} la liste des items achetés par u_a, pour chaque item $j \in I \backslash I_{u_a}$ candidat à la recommandation, on calcule la somme de ses similarités, $x_{u_a}(j) = \sum_{i \in I_{u_a} \cap voisins_k(j)} (sim(j,i))$ avec ses k plus proches voisins ($voisins_k(j)$) parmi les items achetés par u_a. La liste des Top-N items à recommander à l'utilisateur courant est constituée des N items ayant les plus grandes valeurs de $x_{u_a}(j)$.

2.1.1.5 Calcul de la similarité

Le calcul de la similarité a pour objectif de déterminer les voisins d'un utilisateur (ou d'un item). Dans les algorithmes basés sur les voisins tels que définis dans cette section, la similarité entre deux utilisateurs ou items est calculée en mesurant la corrélation existante entre leurs votes. C'est pour cette raison que ces approches sont dites également approches basées sur la corrélation des votes des voisins (neighborhood approches based on rating correlation) [Desrosiers and Karypis, 2011].

Cosinus : une mesure de similarité entre deux objets a et b, très utilisée en recherche et filtrage d'information [Salton, 1989], consiste à représenter les deux objets par deux vecteurs $\vec{x_a}$ et $\vec{x_b}$ et

de mesurer le cosinus de l'angle formé par les deux vecteurs.

$$sim(a,b) = \cos(\vec{x_a}, \vec{x_b}) = \frac{\vec{x_a} \bullet \vec{x_b}}{\|\vec{x_a}\| \, \|\vec{x_b}\|} \tag{2.8}$$

Dans le cas du filtrage collaboratif, chaque utilisateur u est représenté par un vecteur $x_u \in R^{|I|}$, où $x_{ui} = v_{ui}$ si l'utilisateur u a évalué l'item i. Étant donné les valeurs manquantes dans la matrice des votes, le cosinus est calculé sur l'ensemble des items notés par les deux utilisateurs. La similarité entre deux utilisateurs u et w est alors donnée par la formule 2.9.

$$sim(u,w) = \cos(\vec{x_u}, \vec{x_v}) = \frac{\sum_{i \in I_{uw}} v_{ui} \times v_{wi}}{\sqrt{\sum_{i \in I_{uw}} v_{ui}^2} \sqrt{\sum_{i \in I_{uw}} v_{wi}^2}} \tag{2.9}$$

La formule 2.10 applique le cosinus pour calculer la similarité entre deux items. En fait, il suffit de remplacer dans l'équation 2.9 les utilisateurs par leurs équivalents en items.

$$sim(i,j) = \cos(\vec{x_i}, \vec{x_j}) = \frac{\sum_{u \in U_{ij}} v_{ui} \times v_{uj}}{\sqrt{\sum_{u \in U_{ij}} v_{ui}^2} \sqrt{\sum_{u \in U_{ij}} v_{uj}^2}} \tag{2.10}$$

Le Cosinus varie entre 0 et 1. Une valeur égale à 1 indique que les deux utilisateurs ont des préférences identiques, une valeur égale à 0 indique qu'ils n'ont rien en commun. Un inconvénient majeur de l'utilisation du cosinus dans le filtrage collaboratif est qu'il ne tient pas compte de la variation dans le jugement des utilisateurs.

Coefficient de corrélation de Pearson : utilisé par les auteurs du système GroupLens [Resnick *et al.*, 1994] pour calculer la similarité entre deux utilisateurs u et w de U. Le coefficient de corrélation de Pearson mesure la liaison linéaire entre deux variables numériques en calculant le rapport entre leur covariance et le produit non nul de leur écart type. Il permet ainsi de mesurer la similarité en supprimant l'inconvénient de la variation des votes. L'équation 2.11 donne la formule de calcul de la mesure de corrélation entre deux utilisateurs u et w. En raison des données manquantes, seuls les items notés à la fois par u et w sont pris en compte.

$$sim(u,w) = Pearson(u,w) = \frac{\sum_{i \in I_{uw}} (v_{ui} - \overline{v_u})(v_{wi} - \overline{v_w})}{\sqrt{\sum_{i \in I_{uw}} (v_{ui} - \overline{v_u})^2} \sqrt{\sum_{i \in I_{uw}} (v_{wi} - \overline{v_w})^2}} \tag{2.11}$$

Le coefficient de corrélation de Pearson varie entre -1 et 1. Une valeur égale à 1 indique que les utilisateurs partagent exactement les mêmes goûts, une valeur de -1 indique qu'ils ont des goûts totalement opposés. Le coefficient de Pearson est généralement reconnu comme étant la meilleure mesure pour calculer la similarité entre les utilisateurs [Herlocker *et al.*, 1999]. Il peut être également utilisé pour mesurer la corrélation entre deux items. L'équation 2.12 donne le coefficient de Pearson entre deux items i et j.

$$sim(i,j) = Pearson(i,j) = \frac{\sum_{u \in U_{ij}} (v_{ui} - \overline{v_i})(v_{uj} - \overline{v_j})}{\sqrt{\sum_{u \in U_{ij}} (v_{ui} - \overline{v_i})^2} \sqrt{\sum_{u \in U_{ij}} (v_{uj} - \overline{v_j})^2}} \tag{2.12}$$

Le cosinus ajusté (Adjusted cosine) : lorsqu'on applique le coefficient de Pearson pour mesurer la similarité entre deux items (voir formule (2.12)), les votes d'un même utilisateur sont centrés par rapport à la moyenne de ses votes. Or, la variation de notation pour un même

utilisateur n'est pas aussi importante que la variation entre les différents utilisateurs. C'est pour cette raison qu'il est plus intéressant, lors du calcul de la similarité entre deux items, d'ajuster les votes par rapport à la moyenne des votes des utilisateurs plutôt que par rapport à la moyenne des votes des items. C'est ce que fait le Cosinus ajusté.

Introduit par Sarwar et al. [Sarwar *et al.*, 2001], c'est la plus populaire et la plus pertinente des mesures utilisées dans le calcul de la similarité entre deux items dans les algorithmes de filtrage collaboratifs selon [Schafer *et al.*, 2007]. Le cosinus ajusté est en fait une amélioration du cosinus en ajustant la valeur des votes d'un utilisateur par rapport à la moyenne de ses votes.

$$sim(i,j) = CosinusAjusté(i,j) = \frac{\sum_{u \in U_{ij}} (v_{ui} - \overline{v_u}) \times (v_{uj} - \overline{v_u})}{\sqrt{\sum_{u \in U_{ij}} (v_{ui} - \overline{v_u})^2} \sqrt{\sum_{u \in U_{ij}} (v_{uj} - \overline{v_u})^2}} \tag{2.13}$$

Comme le coefficient de Pearson, le Cosinus ajusté varient entre -1 et 1. Une valeur égale à 1 indique que les deux items sont identiques, une valeur égale à -1 indique qu'ils sont opposés.

2.1.1.6 Sélection des voisins

Une première sélection des voisins d'un utilisateur u consiste à ignorer toutes les similarités négatives. Une forte similarité positive entre deux utilisateurs est un bon indicateur de leur appartenance à un même groupe d'intérêt, alors qu'une similarité négative indique qu'ils appartiennent à des groupes d'intérêts différents. Les expérimentations menées par les auteurs de [Herlocker *et al.*, 1999] ont montré que les corrélations négatives n'apportent aucune amélioration quant à la pertinence de la prédiction. Le même principe est appliqué lors de la détermination des voisins d'un item donné.

Une deuxième sélection se fait lors du calcul de la prédiction, en sélectionnant les k plus proches voisins (k nearest neighbors). La détermination de k se fait de façon empirique. Les expérimentations menées dans la littérature montrent que la pertinence de la prédiction est sensible à la valeur de k [Sarwar *et al.*, 2001; Mobasher *et al.*, 2004; Herlocker *et al.*, 1999] et qu'elle suit une courbe concave. Lorsque le nombre de voisins est faible ($k < 20$), la pertinence est également faible, lorsque k augmente, le nombre de voisins contribuant à la prédiction augmente ce qui augmente la pertinence de la prédiction. Enfin, la pertinence décroit lorsque trop de voisins ($k > 60$) participent à la prédiction, ceci est dû au fait que les relations fortes avec les plus proches voisins se trouvent diluées par les plus faibles.

2.1.2 Les points forts des algorithmes basés sur les voisins

Les algorithmes de filtrage collaboratif basé sur les voisins ont de nombreux avantages parmi lesquels on peut citer :

La simplicité : une telle approche est intuitive et simple à implémenter. Dans sa forme la plus simple, seul le nombre k de voisins à considérer est à définir.

Explication : de récents travaux ont montré la nécessiter d'expliquer les raisons des recommandations proposées aux utilisateurs pour les inciter à ajuster leur profil [Tintarev and Masthoff, 2011]. Tel est le cas par exemple des sites commerciaux qui affichent, à la suite de la consultation d'un produit par l'utilisateur courant, un message de la forme *"les client ayant consulté ce produit ont également consulté ces produits"*. Les approches basées sur les voisins permettent aisément de

fournir des justifications. Ainsi, le système de recommandation de livres de Amazon, qui utilise un algorithme basé sur les items, affiche comme explication *"'Ces recommandations sont basées sur les articles que vous possédez et plus encore"'* (voir figure 1). L'utilisateur peut alors affiner son profil en fournissant une liste plus complète des livres qu'il possède.

Efficacité : contrairement aux approches basées sur un modèle, les algorithmes de filtrage collaboratif basés sur la mémoire ne nécessitent pas de phases d'apprentissage coûteuses à renouveler fréquemment. Toutefois, la recommandation basée sur les voisins est plus coûteuse en raison du calcul des similarités entres les items (ou les utilisateurs). Une solution consiste à pré-calculer les similarités, et à ne conserver que les k plus proches voisins. Le stockage des k plus proches voisins ne nécessitant pas un espace important, ce qui permet à de telles approches de passer l'échelle même pour des applications ayant des millions d'utilisateurs et d'items [Desrosiers and Karypis, 2011].

Stabilité : des observations sur les sites de e-commerce ont montré que ces approches ne sont pas très sensibles à l'ajout constant d'utilisateurs, d'items ou de votes [Desrosiers and Karypis, 2011]. Ainsi, lorsque les similarités entre les utilisateurs sont calculées, il est possible à un algorithme basé sur les utilisateurs de recommander de nouveaux items sans recalculer les similarités. De même, lorsqu'un nouvel utilisateur évalue un certain nombre d'items, seule sa similarité avec les utilisateurs du système est à recalculer.

Effet de surprise : (serendipity) : serendipity en anglais désigne l'effet de surprise que peut ressentir un utilisateur en recevant une recommandation pertinente qu'il n'aurait jamais soupçonné intéressante pour lui. Des études [Good *et al.*, 1999] ont montré que "'l'effet de surprise"' est un facteur important pour mesurer la satisfaction des utilisateurs. Les algorithmes basés sur les utilisateurs permettent de faire des recommandations à *effet de surprise*. En effet, si un utilisateur A est proche d'un utilisateur B du fait qu'il ne regarde que des comédies, et si B apprécie un film d'un autre genre, ce film peut être recommandé à A du fait de sa proximité avec B.

2.1.3 Les défis du filtrage collaboratif basé sur les voisins

Les algorithmes basés sur la corrélation des votes, tels que présentés dans cette section, ont démontré leur efficacité surtout lorsque le nombre de votes est suffisant pour déterminer des groupes d'intérêts. Les auteurs [Schickel and Faltings, 2006] ont démontré que les utilisateurs doivent avoir un minimum de 20 votes pour avoir des prédictions pertinentes. Cependant, ils sont confrontés à deux défis : *la sensibilité aux données manquantes* et *le passage à l'échelle* pour des applications avec un nombre d'utilisateurs et d'items très élevé, comme c'est la cas des applications de e-commerce.

Sensibilité aux données manquantes : les données manquantes sont un problème commun à la majorité des systèmes de recommandation [Papagelis *et al.*, 2005] en raison du fait que les utilisateurs ne notent que très peu d'items. Ainsi, même les items les plus populaires se trouvent notés par un petit nombre parmi la totalité des utilisateurs. De ce fait, la matrice des votes est une matrice creuse avec un taux de valeurs manquantes pouvant atteindre 95% voire 99% du total des valeurs, soit un taux de couverture de l'ordre de 5% à 1%. C'est le cas, par exemple,

du système de recommandation MovieLens [24] [MovieLens, 2014]. Les données manquantes sont à l'origine de nombreux problèmes auxquels doivent faire face les algorithmes de filtrage collaboratif basé sur les voisins puisque la similarité entre deux utilisateurs est basée sur les items qu'ils ont co-votés. Le même problème se pose pour la similarité entre les items.

Le taux de couverture : un utilisateur ne peut recevoir des recommandations que si le système arrive à identifier ses voisins. Un utilisateur ayant évalué des items non communs ne peut pas recevoir des recommandations, ce problème est connu sous le nom de *mouton gris* (*Gray Sheep*). Par ailleurs, la similarité exige une paire d'utilisateurs ayant évalué un certain nombre d'items. En effet, si deux utilisateurs n'ont que 3 items en commun, l'algorithme les qualifiera comme totalement similaires, s'ils les ont notés de la même manière. Les auteurs [Schickel and Faltings, 2006] ont démontré qu'il faut un minimum de 20 par utilisateur et 7 votes en commun pour que la mesure de similarité ne soit pas biaisée et que les recommandations soient pertinentes. Ainsi, les utilisateurs ayant très peu de votes ne peuvent pas recevoir de recommandations pertinentes.

Transitivité des voisins : les mesures de similarité ne permettent pas de détecter les utilisateurs de goûts similaires n'ayant aucun items en commun comme étant des voisins proches. En effet, si l'utilisateur u est le voisin proche de l'utilisateur w et w est le voisin proche de l'utilisateur z, si z et u ne partagent aucun item alors la similarité entre u et z ne pourra pas être calculée et le système sera dans l'incapacité de détecter qu'ils sont également voisins proches. Ce problème est connu dans la littérature sous le nom de *transitivité des voisins* (neighbors transitivity).

Le démarrage à froid (cold start) : concerne les items et les utilisateurs lorsqu'ils sont nouvellement introduits dans le système [Schein *et al.*, 2002]. Un nouvel utilisateur qui n'a noté aucun item ne peut pas recevoir de recommandation puisque le système ne connaît pas ses goûts, ce problème est connu sous le nom du démarrage à froid pour les utilisateurs (user cold start). Une solution à ce problème est de lui demander explicitement de noter un certain nombres d'items. C'est le cas par exemple, du système de recommandation de MovieLens [MovieLens, 2014], qui demande à l'utilisateur de noter 20 films avant de recevoir ses premières recommandations. D'autres solutions consistent à recommander les items les plus populaires ou de choisir aléatoirement une liste d'items parmi les items à recommander. Le même problème se pose lorsqu'un nouvel item est ajouté dans le système, il ne peut être recommandé avant de recevoir un nombre suffisant d'évaluations.

Parmi les solutions proposées au problème de données manquantes on trouve l'utilisation de valeurs par défaut pour calculer la similarité entre les utilisateurs ou les items, telle que l'utilisation d'une valeur neutre [Breese *et al.*, 1998], la moyenne des votes d'un utilisateur, d'un item ou la moyenne des votes d'un groupe d'utilisateurs [Chee *et al.*, 2001]. D'autres approches consistent à exploiter le contenu des items pour réduire l'impact des données manquantes [Degemmis *et al.*, 2007; Good *et al.*, 1999; Melville *et al.*, 2002], il peut s'agir par exemple d'utiliser les similarités basées sur le contenu à la place ou en plus des similarités basées sur les usages [Mobasher *et al.*, 2004] ou de remplir les données manquantes par des prédictions issues d'un système de recommandation basé sur le contenu [Melville *et al.*, 2002]. D'autres approches consistent à faire de la

24. Estimation calculée à partir des benchmarks MovieLens 100k et MovieLens 1M http ://grouplens.org/datasets/movielens/.

factorisation de matrice [Koren, 2008; Takacs *et al.*, 2008]. La factorisation peut être appliquée à la matrice des votes ou à la matrice des similarités des items ou des utilisateurs [Desrosiers and Karypis, 2011].

Passage à l'échelle : un deuxième problème auquel sont confrontés les algorithmes de filtrage collaboratif basés sur les voisins est la complexité des calculs. En effet, le calcul des similarités entres les utilisateurs (ou les items) est assez coûteux, surtout pour des applications de e-commerce avec un nombre très élevé d'utilisateurs et d'items. Par ailleurs, dans ce type d'application, le système ne doit pas dépasser une fraction de seconde pour fournir des recommandations pertinentes à l'utilisateur courant. Parmi les solutions proposées, on trouve le calcul des similarités en offline [Linden *et al.*, 2003], ainsi seul le calcul de la prédiction sera effectué en temps réel. Le problème qui se pose alors est la complexité en terme d'espace mémoire ($O(\|U\|^2)$ pour la similarité entre les utilisateurs et ($O(\|I\|^2)$ pour la similarité entres les items [Desrosiers and Karypis, 2011]. En ne stockant que les k plus proches voisins, la complexité en espace mémoire est largement réduite. Une autre approche consiste à combiner les algorithmes basés sur les voisins avec des méthodes de réduction de la dimension [Billsus and Pazzani, 1998; Goldberg *et al.*, 2001; Bell *et al.*, 2007; Koren, 2008].

2.1.4 Méthode de réduction de la dimension

Les méthodes de réduction de la dimension permettent de projeter les items (et/ou les utilisateurs) dans une dimension réduite définie par des variables latentes afin de traiter le problème de la sensibilité aux données manquantes [Ekstrand *et al.*, 2011] et le problème du passage à l'échelle [Schafer *et al.*, 2007]. Vu que les utilisateurs (ou les items) seront comparés dans cet espace plus dense et plus réduit que l'espace défini par la matrice des votes, de nouvelles relations entre des paires d'utilisateurs (ou les items) pourront être détectées même s'ils n'ont aucun item en commun.

Les méthodes de réduction de dimension ont connu un large succès [Koren and Bell, 2011] surtout après le challenge lancé par Netflix [Bennett and Lanning, 2007]. En effet, des travaux [Takács *et al.*, 2007; Koren *et al.*, 2009] analysant les résultats du challenge ont démontré la supériorité en terme de précision des approches appliquant des techniques de réduction de la dimension par rapport aux algorithmes de filtrage collaboratif basé sur les voisins. Les méthodes de factorisation de matrice [Koren, 2008; Takacs *et al.*, 2008; Canny, 2002] sont utilisées pour l'extraction des variables latentes telles que l'Analyse en Composante Principale (ACP) [Goldberg *et al.*, 2001] ou la Décomposition en Valeur Singulière (SVD en anglais Singular Value Decomposition) [Bell *et al.*, 2007; Koren, 2008]. Elles ont été essentiellement utilisées, soit pour réduire la dimension de la matrice des votes, soit pour réduire la dimension de la matrice de similarité [Desrosiers and Karypis, 2011].

L'Analyse Latente Sémantique (Latent Semantic Analysis (LSA)) appelée également Indexation Sémantique Latente (Latent Semantic Indexing (LSI)) est une méthode de réduction de dimension très répandue dans le domaine du filtrage et de la recherche d'information [Belkin and Croft, 1992]. Elle a également été utilisée pour réduire la dimension de la matrice des votes M dans les algorithmes de filtrage collaboratif [Desrosiers and Karypis, 2011]. LSA approxime la matrice $M_{|U|,|I|}$ de rang r par une matrice $M' = PQ^t$ de rang $k << r$, où $P_{|U|,k}$ est une matrice représentant les utilisateurs dans l'espace réduit LSA, et $Q_{|I|,k}$ est une matrice représentant les items dans l'espace réduit LSA. P et Q étant deux matrices orthonormales. Pour déterminer les

matrices P et Q, une décomposition en valeurs singulières (SVD) est appliquée à la matrice des votes M. La SVD décompose la matrice M en trois matrices D, Σ et T

$$M = D_{|U|,r} * \Sigma_{r,r} * T^t_{r,|I|} \tag{2.14}$$

où D et T sont deux matrices orthonormales, r est le rang de la matrice M et Σ est une matrice diagonale dont les valeurs représentent les valeurs singulières de la matrice M ordonnées par ordre décroissant. LSA applique une SVD tronquée pour approximer la matrice M en ne conservant que les k plus grandes valeurs singulières et leurs vecteurs singuliers correspondants.

$$M \approx M^{'} = D_{|U|,k} * \Sigma_{k,k} * T^t_{k,|I|} \tag{2.15}$$

Les matrices modélisant respectivement les utilisateurs et les items dans l'espace réduit LSA sont alors définies par $P = D_k \Sigma_k^{1/2}$ et $Q = T_k \Sigma_k^{1/2}$. Chaque utilisateur dans $P_{|U|,k}$ est représenté par un ensemble de k variables latentes au lieu de ses votes. Chaque item dans $Q_{|I|,k}$ est représenté par k variables latentes au lieu des votes des utilisateurs.

Une fois les matrices P et Q obtenues, la prédiction du vote de l'utilisateur u pour l'item i est donné par :

$$pred(u,i) = p_u q_i^t \tag{2.16}$$

Toutefois, le problème principal qui se pose lors de l'application d'une SVD à la matrice des votes M est la présence des données manquantes. Une SVD ne peut pas s'appliquer à une matrice comportant des valeurs manquantes. Une solution consiste à attribuer des valeurs par défaut aux valeurs manquantes ce qui peut biaiser les données [Desrosiers and Karypis, 2011]. La solution la plus fréquente est l'apprentissage de P et Q en se limitant aux votes connus [Takács *et al.*, 2007; Takacs *et al.*, 2008; Bell *et al.*, 2007; Koren, 2008]. Une étude intéressante sur les récents travaux appliquant la factorisation de matrice dans les algorithmes de recommandations collaboratives est donnée dans [Koren and Bell, 2011].

2.1.5 Les modèles probabilistes

Le principe de ces approches consiste à modéliser le comportement des utilisateurs par un modèle probabiliste pour pouvoir prédire ses comportements futurs. L'idée sur laquelle repose ces algorithmes consiste à calculer la probabilité $P(v|u,i)$ que l'utilisateur u attribue l'évaluation v à l'item i connaissant ses évaluations antérieures. La prédiction du vote $pred(u,i)$ correspond, soit au vote ayant la plus grande probabilité, soit à l'évaluation espérée telle que définie par la formule (2.17) [Schafer *et al.*, 2007].

$$pred(u,i) = E(v|u,i) = \sum_{v \in V} v.P(v|u,i) \tag{2.17}$$

V étant l'ensemble des valeurs que peut prendre un vote (un vote est défini sur une échelle de valeurs de 1 à 5 ou 1 à 7 par exemple).

Cross Sel [Kitts *et al.*, 2000] utilise un classifieur bayésien naïf pour faire de la recommandation. A partir de l'historique des achats d'un utilisateur, il estime la probabilité qu'il achète un item j sachant qu'il a acheté l'item i. Breese et all. [Breese *et al.*, 1998], qui comptent parmi les premiers travaux appliquant un modèle probabiliste dans les algorithmes de filtrage collaboratif,

proposent un modèle construit à partir des réseaux Bayésiens et utilisant les arbres de décision pour calculer les probabilités.

Plusieurs modèles probabilistes issus des domaines de recherche sur l'intelligence artificielle et sur l'apprentissage automatique ont été appliqués au filtrage collaboratif. Certains ont défini le problème de recommandation comme étant un problème de classification [Breese *et al.*, 1998]. Les réseaux Bayésien [Breese *et al.*, 1998; Chien and George, 1999; Zigoris and Zhang, 2006; Castagnos and Boyer, 2007], les processus décisionnels de Markov [Shani *et al.*, 2005] et les réseaux de neurones [Salakhutdinov *et al.*, 2007] ont été utilisés pour faire de la recommandation.

D'autres approches ont appliqué des méthodes probabilistes pour réduire la dimension ou faire du clustering [Hofmann, 2003; Jin *et al.*, 2004]. Les techniques de réduction de la dimension utilisent des méthodes de factorisation de matrice probabilistes. PLSA Probabilistic latent semantic analysis [Hofmann, 2004] est une technique de factorisation similaire à la décomposition en valeur singulière (SVD) mais provenant de la théorie des probabilités au lieu de la théorie de l'algèbre linéaire [Hofmann, 1999; Ekstrand *et al.*, 2011]. Mobasher et all. [Mobasher *et al.*, 2006] l'ont appliquée pour protéger les algorithmes de filtrage collaboratif contre les manipulations frauduleuses des utilisateurs.

2.1.6 Autres approches

Les règles d'association ont également été utilisées dans le filtrage collaboratif. Le principe de base de ces méthodes est de construire des modèles basés sur les associations récurrentes dans la matrice des votes [Heckerman *et al.*, 2001]. Ainsi, on peut observer que les utilisateurs qui attribuent une note élevée à l'item 1, attribuent également une note élevée à l'item 2. Une règle est représentée par une prémisse (exemple item 1 a une note élevée) et une conclusion (exemple item 2 a une noté élevée). Le support de la règle représente la partie des utilisateurs ayant évalué les items présents à la fois dans la partie prémisse et dans la partie conclusion, et la confiance de la règle constitue la partie des utilisateurs qui vérifient la prémisse et la conclusion de la règle. Pour prédire le vote de l'utilisateur u pour l'item i, on sélectionne les règles ayant l'item i dans la partie conclusion et dont la prémisse contient des items évalués par l'utilisateur u. Des heuristiques sont alors utilisées pour déterminer la prédiction du vote à partir des prémisses et des supports des règles. Une étude intéressante et détaillée sur l'utilisation des règles d'association dans les systèmes de recommandation se trouve dans [Mobasher, 2007; Felfernig *et al.*, 2011].

D'autres travaux ont utilisé le clustering pour traiter le problème du passage à l'échelle en identifiant rapidement les voisins d'un utilisateur [Linden *et al.*, 2003]. Dans ces approches, l'utilisateur courant est comparé à un groupe d'utilisateurs au lieu d'être comparé de façon individuelle à chaque utilisateur. Une fois le groupe d'utilisateurs identifié, les plus proches voisins sont sélectionnés parmi ce groupe (cluster). L'algorithme de clustering non hiérarchique tel que le K-Means [MacQueen, 1967] ou les algorithmes de clustering hiérarchiques [Johnson, 1967] ont été utilisés pour déterminer les groupes d'utilisateurs.

2.2 Le filtrage basé sur le contenu

Un système de recommandation basé sur le contenu [Lang, 1995; Pazzani and Billsus, 1997; Billsus and Pazzani, 2000] recommande à l'utilisateur courant des items ayant une description similaire aux items qu'il a appréciés par le passé. Ce qui revient à identifier les caractéristiques

communes aux items ayant reçu une évaluation favorable de la part de l'utilisateur courant. Ceci suppose qu'une description riche et variée est disponible sur les items à recommander. Dans la plupart des systèmes, la description est donnée sous forme textuelle [Pazzani and Billsus, 2007].

Les approches utilisées dans les systèmes de recommandations basées sur le contenu sont généralement inspirées des domaines de la recherche documentaire [Baeza-Yates and Ribeiro-Neto, 1999; Salton, 1989] et du filtrage d'information [Belkin and Croft, 1992]. Toutefois, un système de recherche d'information personnalisé [Micarelli *et al.*, 2007] est différent d'un système de recommandation basé sur le contenu. La différence se situe essentiellement dans la détermination du profil personnalisé des utilisateurs. Dans un système de recherche personnalisé, le profil utilisateur est construit à partir de l'historique des requêtes introduites par l'utilisateur lors de ses différentes opérations de recherche. Dans un système de recommandation basé sur le contenu, l'utilisateur ne saisit aucune requête, et son profil est construit à partir de l'historique de ses évaluations, plus généralement à partir de l'analyse des usages (voir section 1.1.3, page 16).

Les auteurs de [Lops *et al.*, 2011] découpent l'architecture d'un système de recommandation basé sur le contenu en trois composants : *L'analyseur du contenu*, *l'apprentissage du profil utilisateur* et *le filtrage des items* (ou recommandation).

Analyseur du contenu : lorsque l'information décrivant les items est non structurée (exemple du texte libre), une étape de pré-traitement est nécessaire pour extraire les informations pertinentes et les structurer. Le principal objectif de ce composant est de représenter le contenu non structuré des items (documents, description des produits, pages web, article de journal) sous une forme exploitable par le composant *apprentissage du profil utilisateur*. Ainsi, il analyse le contenu des items et en extrait les principales caractéristiques (mot-clé, concepts, n-grammes, ...) en appliquant des techniques de traitement du langage naturel issues de la recherche et du filtrage de l'information [Baeza-Yates and Ribeiro-Neto, 1999; Belkin and Croft, 1992]. Il est à noter que la structuration d'un contenu non structuré ne rentre pas dans le cadre de ce travail.

L'apprentissage du profil utilisateur : ce composant collecte les données issues de l'analyse des usages pour construire le profil utilisateur. Le profil utilisateur définit le modèle de préférence des utilisateurs pour le contenu des items à partir de leurs appréciations pour les items.

Filtrage : ce module se charge de filtrer les items à recommander à l'utilisateur courant. Le profil utilisateur est comparé à celui des items à recommander en utilisant des mesures de similarités. Le résultat de la comparaison peut être binaire (item pertinent, item non pertinent) ou continu auquel cas une liste triée selon l'ordre de pertinence des items est établie. Une Top-N liste composée des N items susceptibles d'être les plus pertinents sera recommandée à l'utilisateur courant.

2.2.1 Apprentissage du profil utilisateur

Différentes méthodes et techniques ont été utilisées dans la littérature pour l'apprentissage du profil utilisateur dans un système de recommandation basé sur le contenu. Nous présentons ici, quelques-unes d'entre elles. Une présentation plus détaillée se trouve dans [Montaner *et al.*, 2003; Pazzani and Billsus, 2007; Lops *et al.*, 2011].

Modèle vectoriel (Vector Space Model ,VSM) : les items sont représentés par un même ensemble de termes. Un terme est soit un mot soit un concept (pour les items non structurés) soit la valeur d'un attribut (pour les items structurés). Le modèle vectoriel consiste à représenter un item par un vecteur de valeurs, chaque valeur est associée à un terme indiquant l'importance du terme dans l'item. La valeur peut être un booléen indiquant la présence du terme dans l'item, un réel indiquant la fréquence ou l'importance du terme dans l'item. TF-IDF [Salton, 1989](TF : Term Frequency, IDF : Inverse Document Frequency) représente la mesure la plus populaire pour déterminer l'importance d'un terme dans un document ou dans un item faisant partie d'une collection ou d'un corpus [Pazzani and Billsus, 2007]. Le modèle vectoriel a été largement utilisé dans les systèmes de recommandation dans différents domaines tels que la recommandation de pages web (Letizia [Lieberman, 1995], Personal WebWatcher [Mladenic, 1999] et Syskill & Webert [Pazzani *et al.*, 1996; Pazzani and Billsus, 1997]), la recommandations de news (NewsDude [Billsus and Pazzani, 2000], INFOrmer [Sorensen *et al.*, 1997] et YourNews [Ahn *et al.*, 2007]) et la recommandation de livres LIBRA [Mooney and Roy, 2000].

Le profil utilisateur est représenté par un modèle vectoriel (VSM) défini dans le même espace que celui modélisant les items. Différentes techniques ont été utilisées dans la littérature pour l'apprentissage du profil utilisateur [Pazzani and Billsus, 2007]. Nous citons à titre d'exemples les systèmes de recommandation NewsWeeder [Lang, 1995], NewT [Sheth and Maes, 1993] et YourNews [Ahn *et al.*, 2007] qui ont appliqué la méthode Rocchio [Rocchio, 1971] qui détermine le vecteur profil de l'utilisateur à partir de la moyenne des profils des items pertinents et non pertinents.

Réseau sémantique pondéré (weighted semantic networks) : les réseaux sémantiques permettent de stocker la signification des mots permettant ainsi un apprentissage plus précis du profil utilisateur [Montaner *et al.*, 2003]. La principale motivation de telles approches est de fournir des systèmes de recommandation intégrant l'aspect culturel et linguistique permettant d'interpréter le langage naturel et de raisonner sur son contenu [Lops *et al.*, 2011]. Le système SiteIF [Stefani and Strapparava, 1998] est un système de recommandation d'articles (news) multilingues intégrant des connaissances linguistiques pour faire de la recommandation. Le profil utilisateur est représenté par un réseau sémantique où chaque nœud représente un mot lu par l'utilisateur, l'arc entre deux nœuds modélise la relation entre deux mots, un poids est associé à chaque nœud et à chaque arc représentant les différents niveaux d'intérêt de l'utilisateur. Quickstep [Middleton *et al.*, 2004b] recommande des articles de recherche, le système utilise une ontologie décrivant la catégorie des articles de recherches pour modéliser les items. Une présentation plus détaillée sur l'utilisation des ontologies est donnée dans [Lops *et al.*, 2011].

Autres modèles : d'autres méthodes ont été utilisées pour modéliser le profil utilisateur dans les systèmes de recommandation basés sur le contenu. On cite à titre d'exemple l'usage des n-grammes pondérés pour modéliser les items dans le système PSUN [Sorensen and McElligott, 1995]. Les auteurs de [Riordan and Sorensen, 1995] ont utilisé les réseaux associatifs pondérés (weighted associative networks) pour modéliser le profil utilisateur. Un réseau associatif pondéré représente les termes, concepts ou mots qui suscitent l'intérêt de l'utilisateur, un ensemble de relations pondérées établit l'organisation de ces termes en phrases pertinentes. En plus de ces modèles, plusieurs systèmes ont modélisé le profil utilisateur par un modèle de classification tel que les arbres de décision, les réseaux de neurones, les règles d'induction ou les réseaux bayésiens [Montaner *et al.*, 2003; Pazzani and Billsus, 2007].

2.2.2 Recommandation

Dans une méthode de recommandation basée sur le contenu, l'utilité d'un item i pour un utilisateur u, définie par la fonction $ut(u,i)$, est estimée à partir des utilités $ut(u,i')$ (évaluations) attribuées par l'utilisateur u aux items $i' \in I$ similaires à i [Adomavicius and Tuzhilin, 2005]. Soit $contenu(i)$ le profil modélisant la description de l'item i. Soit $profilcontenu(u)$ le profil de l'utilisateur u représentant les préférences de cet utilisateur pour le contenu des items. Ce profil est obtenu en analysant le contenu des items précédemment notés par l'utilisateur, et est généralement construit à partir des techniques d'analyse de contenu issues de la recherche d'information [Salton, 1989]. Les auteurs de [Adomavicius and Tuzhilin, 2005] définissent l'utilité $ut(u,i)$ comme suit :

$$ut(u,i) = score(ProfilContenu(u), Contenu(i)) \tag{2.18}$$

Dans ce qui suit nous présentons brièvement les principales méthodes de recommandation basée sur le contenu. Une étude plus détaillée se trouve dans [Montaner *et al.*, 2003; Pazzani and Billsus, 2007; Lops *et al.*, 2011].

2.2.2.1 Cosinus comme mesure de similarité

Le profil utilisateur et le profil modélisant l'item sont tous les deux représentés par un modèle vectoriel (VSM). Ainsi, $contenu(i) = \vec{w_i}$ et $profilcontenu(u) = \vec{w_u}$ représentent deux vecteurs de poids, représentant par exemple le TF-IDF de chaque terme (ou mot) respectivement pour l'item i et pour l'utilisateur u. L'utilité $ut(u,i)$ est alors obtenue en mesurant le cosinus de l'angle formé par les deux vecteurs. Le cosinus comme mesure de similarité provient du domaine de recherche et de filtrage d'information [Belkin and Croft, 1992]. Il a été utilisé initialement dans SMART [Salton and McGill, 1983] pour mesurer la similarité entre un document et une requête tous deux représentés par deux vecteurs définis dans le même espace dimensionnel.

$$ut(u,i) = cosinus(\vec{profilcontenu}(u), \vec{contenu}(i)) = \frac{\vec{profilcontenu}(u) \bullet \vec{contenu}(i)}{\left\|\vec{profilcontenu}(u)\right\| \left\|\vec{contenu}(i)\right\|} \tag{2.19}$$

2.2.2.2 Plus proches voisins

L'algorithme des plus proches voisins classifie un item non observé par l'utilisateur courant en le comparant aux items qu'il a observés en utilisant une fonction de similarité [Lops *et al.*, 2011]. La classe de l'item non observé est déterminée à partir des classes d'appartenance de ses N plus proches voisins. La fonction utilisée pour le calcul de la similarité dépend de la représentation des items. Pour une représentation structurée, la distance euclidienne est utilisée, pour une représentation en VSM, le Cosinus est le plus souvent utilisé [Pazzani and Billsus, 2007]. LaboUr [Kamba *et al.*, 1995], NewsDude [Billsus and Pazzani, 1999] WebSell [Cunningham *et al.*, 2001] et Quickstep [Middleton *et al.*, 2004a] sont différents exemples de systèmes de recommandation appliquant l'algorithme des plus proches voisins.

2.2.2.3 Classification

La tâche de recommandation peut être ramenée à une tache de classification dans un système de recommandation basé sur le contenu. A partir de la description des items, le système infère un modèle pour chaque utilisateur permettant de classifier un item non observé. La classification

consiste à définir deux classes (*interessant*, *non intéressant*) [Billsus and Pazzani, 1999], mais l'algorithme peut également classifier les items dans plusieurs classes [Montaner *et al.*, 2003]. Syskill & Webert [Pazzani *et al.*, 1996; Pazzani and Billsus, 1997], NewsDude [Billsus and Pazzani, 1999], Daily Learner [Billsus and Pazzani, 2000], LIBRA [Mooney and Roy, 2000] et ITR [Degemmis *et al.*, 2007; Semeraro *et al.*, 2009] ont appliqué une classification naïve bayésienne pour construire le profil utilisateur et estimer l'intérêt qu'il a pour un item qui lui est inconnu.

D'autres systèmes ont utilisé des modèles de classification plus complexes tels que les arbres de décision, les réseaux de neurones ou les règles d'induction. A titre d'exemples, Re :Agent [Boone, 1998] ont implémenté un réseau de neurones pour filtrer des e-mails en deux catégories *travail (work)* et *autre (other)*. Syskill & Webert [Pazzani *et al.*, 1996] ont utilisé un arbre de décision pour classifier des pages web en *intéressant* ou *non intéressant* pour l'utilisateur courant. Les auteurs de [Basu *et al.*, 1998] ont utilisé des règles d'induction pour classifier des films. Une présentation détaillée de différentes méthodes de classification est donnée dans [Pazzani and Billsus, 2007].

2.2.3 Les points forts du filtrage basé sur le contenu

Le filtrage basé sur le contenu présente plusieurs avantages lorsqu'on le compare au filtrage collaboratif :

Indépendance de l'utilisateur : le filtrage basé sur le contenu traite chaque utilisateur de façon indépendante. Ainsi, seules les évaluations de l'utilisateur courant sont prises en compte pour construire son profil utilisateur et faire de la recommandation. Ce qui n'est pas le cas pour les approches collaboratives qui nécessitent les votes des autres utilisateurs pour déterminer les plus proches voisins de l'utilisateur courant. Ainsi, seuls les items appréciés par les plus proches voisins seront recommandés.

Nouvel item : le filtrage basé sur le contenu peut recommander des items nouvellement introduits dans la base avant même qu'ils reçoivent aucune évaluation de la part des utilisateurs. Ce qui n'est pas le cas des approches collaboratives qui ne peuvent recommander un item que s'il a été préalablement évalué par quelques utilisateurs. C'est le problème du démarrage à froid (cold start) pour les items.

2.2.4 Les limites du filtrage basé sur le contenu

Le filtrage basé sur le contenu doit faire face à de nombreux défis dont les plus importants :

Limite de l'analyse du contenu : une limite naturelle du filtrage basé sur le contenu est la nécessité de disposer d'une représentation variée et riche du contenu des items, ce qui n'est pas toujours le cas. La précision des recommandations est liée à la quantité d'informations dont dispose le système pour discriminer les items appréciés de ceux non appréciés par l'utilisateur [Lops *et al.*, 2011]. Contrairement au filtrage collaboratif qui peut traiter tout type d'items, le filtrage basé sur le contenu ne peut traiter que les items disposant d'un contenu pouvant être analysé. Un autre problème inhérent à l'analyse du contenu est l'incapacité du système à faire la distinction entre les bons et mauvais items puisque la sélection se fait uniquement sur la description du contenu. Par exemple, dans un système de recommandation d'articles, si deux articles de presse traitent du même sujet, ils seront forcément décrits par les mêmes termes et

auront par conséquent le même profil, le système n'aura aucun moyen de distinguer entre l'article de presse bien rédigé de celui mal rédigé.

Sur-spécialisation (Over-specialization) : ce problème est la conséquence directe du principe même du filtrage basé sur le contenu. En effet, puisque le système ne peut recommander que les items similaires au profil utilisateur, l'utilisateur courant se trouve limité à recevoir des recommandations proches des items qu'il a observés par le passé. Le problème de sur-spécialisation ne se limite pas au fait que le système ne peut pas recommander à l'utilisateur des items différents de ceux qu'il a déjà observés, mais également qu'il ne doit pas lui recommander des items trop proches de ceux qu'il a appréciés par le passé. C'est pourquoi certains systèmes de recommandation, tels que Daily Learner [Billsus and Pazzani, 2000], éliminent non seulement les items différents du profil utilisateur, mais également ceux très proches des items déjà observés par l'utilisateur courant. La diversité des recommandations est un critère d'évaluation de la qualité des recommandations [Shani and Gunawardana, 2011]. Idéalement, l'utilisateur doit recevoir des recommandations pertinentes diversifiées et non homogènes [Adomavicius and Tuzhilin, 2005]. Ainsi, il n'est pas judicieux de recommander tous les films de Woody Allen à un utilisateur qui a apprécié l'un de ses films.

Nouvel utilisateur : un utilisateur doit évaluer un certain nombre d'items avant que le système ne puisse interpréter ses préférences et lui fournir des recommandations pertinentes. Ce problème est connu dans la littérature sous le nom du problème de démarrage à froid pour les utilisateurs (user clod start).

2.3 Les systèmes de recommandation hybride

Un système de recommandation hybride est un système qui combine deux ou plusieurs techniques de recommandations différentes. Burke [Burke, 2002], a identifié 4 classes de techniques de recommandation : *la recommandation collaborative*, *la recommandation basée sur le contenu*, *la recommandation basée sur les données démographiques* [Pazzani, 1999; Kobsa *et al.*, 2001], et *la recommandation basée sur la connaissance* [Schmitt and Bergmann, 1999; Smyth, 2007]. Comme nous l'avons déjà mentionné, les systèmes de recommandation démographique ne font pas partie de la recommandation personnalisée, et la recommandation basée sur la connaissance (voir section 1.2.1, page 22) suppose un échange d'information sur les centres d'intérêt de l'utilisateur, ces deux techniques ne rentrent pas dans le cadre de ce travail. C'est pour cette raison que nous nous intéressons à l'hybridation entre la recommandation collaborative et la recommandation basée sur le contenu.

2.3.1 Pourquoi l'hybridation ?

La recommandation basée sur le contenu et la recommandation collaborative ont souvent été considérées comme complémentaires [Adomavicius and Tuzhilin, 2005]. Le filtrage sur le contenu permet de recommander les nouveaux items non encore évalués par aucun utilisateur, alors que le filtrage collaboratif ne peut recommander un item que s'il a été au-préalable évalué par un certain nombre d'utilisateurs. Le filtrage sur le contenu nécessite la disposition de données sémantiques sur les items, en plus d'une étape d'analyse pour pouvoir les extraire et les représenter. Dans plusieurs domaines, le contenu sur les items est soit insuffisant (les livres sans résumés disponibles), soit difficile à extraire ou à représenter (musique, film). Le filtrage collaboratif ne requiert pas

de contenu pour faire de la recommandation. La complexité d'un système de recommandation basée sur le contenu est liée à la complexité avec laquelle sont extraites et analysées les données sémantiques sur les items. A titre d'exemple, si le système ne dispose que du genre des films comme information sur le contenu dans un système de recommandation de films, alors le modèle ne pourra intégrer que cette dimension. De plus, s'il s'avère difficile d'extraire automatiquement certaines propriétés sur les items, le filtrage sur le contenu est contraint à les ignorer dans ses algorithmes de recommandation. Par exemple, la qualité des données multimédia (image, vidéo ou audio) d'une page web peut représenter une information importante pour certains utilisateurs, cette information est difficile à extraire automatiquement [Balabanović and Shoham, 1997; Schafer *et al.*, 2007]. Le filtrage collaboratif permet l'évaluation d'une telle caractéristique puisqu' il se base sur les évaluations des utilisateurs.

Par ailleurs, les recommandations produites par le filtrage sur le contenu pour un utilisateur donné souffrent d'un manque de diversité lié au problème de sur-spécialisation (over-specialization) (voir section 2.2.4). Le filtrage collaboratif est considéré par les chercheurs comme étant plus diversifié, produisant même des recommandations à effet de surprise, c'est-à-dire des recommandations pertinentes inattendues par l'utilisateur [Herlocker *et al.*, 2004].

L'hybridation de ces deux techniques, afin de traiter les insuffisances de chaque technique et profiter de leurs points forts, a fait l'objet de plusieurs travaux de recherche [Basu *et al.*, 1998; Claypool *et al.*, 1999; Mobasher *et al.*, 2004; Basilico and Hofmann, 2004; Liu *et al.*, 2010; Porcel *et al.*, 2012; Dooms, 2013]. Par exemple, il peut s'agir de déterminer les items les plus proches des items appréciés par l'utilisateur en appliquant un filtrage sur le contenu, puis d'appliquer un filtrage collaboratif en se basant sur la qualité des items à partir des évaluations des utilisateurs. Le système Fab [Balabanović and Shoham, 1997], compte parmi les premiers systèmes de recommandation hybride, il combine le filtrage collaboratif et le filtrage basé sur le contenu afin de traiter le problème de sur-spécialisation et le problème du démarrage à froid pour les items. Dans ce système, pour qu'un item soit recommandé à l'utilisateur courant deux critères doivent être satisfaits : (a) son profil contenu doit être similaire au profil contenu de l'utilisateur, et (b) l'item doit être apprécié par les voisins les plus proches de l'utilisateur courant.

Il existe plusieurs manières de faire de l'hybridation et aucun consensus n'a été défini par la communauté des chercheurs. Toutefois, Burke [Burke, 2002; Burke, 2007] a identifié sept différentes manières de faire de l'hybridation :

— Pondéré (weighted) : le score ou le vote obtenu par chacune des deux techniques est combiné en un seul résultat.
— Sélection (Switching) : le système bascule entre les deux techniques de recommandation en fonction de la situation.
— Mixte (Mixed) : les recommandations des deux techniques sont proposées simultanément.
— Combiner les propriétés (Feature combination) : les données issues des deux techniques sont combinées et transmises à un seul algorithme de recommandation.
— Augmentation de propriétés (Feature augmentation) : le résultat d'une technique est utilisé comme entrée de l'autre technique.
— Cascade : un système affine les recommandations données par l'autre système.
— Meta-level : une première technique construit un modèle qui sera utilisé comme entrée par la seconde technique.

La section suivante donne un aperçu sur ces différentes techniques.

2.3.2 Techniques d'hybridation

Pondérée

L'item à recommander est obtenu en combinant les résultats de chacune des deux techniques. Les auteurs de [Claypool *et al.*, 1999] fusionnent les prédictions des votes obtenus par chaque technique en une seule prédiction en appliquant une combinaison linéaire. La technique d'hybridation de Pazzani [Pazzani, 1999] combine les votes obtenus par chacune des trois techniques de recommandation (démographique, filtrage basé sur le contenu et filtrage collaboratif) en appliquant un schéma de consensus. Robin Burke [Burke, 2007] a démontré empiriquement que la pertinence des recommandations d'une telle technique d'hybridation est rarement supérieure à celle obtenue par chaque technique séparément, principalement s'il s'agit du filtrage collaboratif. Cependant, Mobasher et all. [Mobasher *et al.*, 2004] ont présenté une autre technique de combinaison afin de booster sémantiquement le filtrage collaboratif basé sur les items. Ils calculent la similarité entre deux items en appliquant une combinaison linéaire entre la similarité issue de leurs profils usage et la similarité issue de leurs profils sémantique, puis appliquent un algorithme de filtrage collaboratif basé sur les items pour calculer la prédiction des votes. Le profil sémantique est modélisé par un vecteur VSM construit en alignant tous les attributs sémantiques représentant l'item. Un attribut sémantique (sementic attribut), selon leur définition, correspond à la valeur prise par un attribut pour un item donné (par exemple *action* pour l'attribut *genre* d'un film donné). Une LSA [25] est également appliquée pour réduire la dimension du vecteur VSM ainsi obtenu. Tout en proposant une solution pour le problème du démarrage à froid pour les items, l'approche présentée a affiché une augmentation de la pertinence des recommandations par rapport à celles enregistrées par un pur algorithme de filtrage collaboratif basé sur les items. Par ailleurs, la meilleure combinaison est généralement déterminée de façon empirique. A titre d'exemple, Mobasher et all. montrent que les meilleures performances sont obtenues par la combinaison 60/40 (sémantique/collaboratif).

Switching

En fonction de la situation, le système choisit les recommandations fournies par la meilleure technique. Le choix entre l'une ou l'autre technique se fait en évaluant la qualité des recommandations [Mobasher and Nakagawa, 2003; Lekakos and Caravelas, 2008]. NewsDude [Billsus and Pazzani, 2000] recommande des reportages en hybridant trois systèmes de recommandation : un algorithme des plus proches voisins basé sur le contenu, un système collaboratif puis un autre système basé sur le contenu utilisant un classifieur bayésien. Les systèmes sont ordonnés, l'algorithme des plus proches voisins est appliqué en premier, s'il ne produit pas des recommandations avec un degré de confiance satisfaisant, alors le filtrage collaboratif est appliqué, puis le classifieur bayésien est appliqué si les résultats du second système ne sont pas satisfaisants. Ce type d'hybridation ajoute une complexité supplémentaire au processus de recommandation puisqu'il exige l'évaluation du critère de basculement entre les différentes techniques [Burke, 2007].

Mixte

Technique d'hybridation applicable lorsqu'il est possible de fournir un nombre important de recommandations. L'hybridation mixte consiste à fournir les recommandations issues de diffé-

25. Latent Semantic Analysis

rents systèmes de recommandations simultanément. Le système PTV [Smyth and Cotter, 2001] applique cette technique d'hybridation pour recommander des programmes TV. Il utilise un filtrage sur le contenu basé sur la description textuelle des programmes TV et un filtrage collaboratif basé sur les préférences des utilisateurs. Les recommandations proposées par les deux techniques sont combinées en une seule liste de suggestions.

Combiner les propriétés

L'idée est d'injecter les propriétés d'une technique de recommandation (par exemple le filtrage collaboratif) dans un algorithme appliquant une autre technique (par exemple filtrage basé sur le contenu). Les auteurs de [Billsus and Pazzani, 1998] utilisent des règles d'induction pour l'apprentissage de règles basées sur le contenu pour définir les préférences des utilisateurs. Ce type d'hybridation est différent des autres techniques d'hybridation puisqu'il ne s'agit pas de combiner différents systèmes de recommandations, mais de combiner des données utilisées par des techniques de recommandation différentes et construire un seul système de recommandation.

Augmentation de propriétés

Une première technique est utilisée pour produire le vote ou la classification d'un item, puis cette information est incorporée dans le processus de recommandation du second système. Par exemple, le système de recommandation de livres Libra [Mooney and Roy, 2000] utilise une technique de recommandation basée sur le contenu à partir des données stockées dans amazon.com en appliquant un classifieur bayésien naïf. Les données textuelles contiennent des informations que Amazon génère en appliquant son algorithme de filtrage collaboratif. Le système hybride de [Melville *et al.*, 2002] construit un modèle basé sur le contenu à partir des données d'apprentissage, puis utilise ce modèle pour générer les prédictions des votes.

Cascade

Dans ce type d'hybridation, une technique de recommandation est utilisée pour produire un premier classement des candidats et une deuxième technique, de plus faible priorité, affine la recommandation parmi l'ensemble des candidats. L'hybridation par cascade évite au système d'employer la technique de recommandation de faible priorité sur les items qui ont été bien différenciés par la première technique, ou ceux qui ont été mal classés au point de ne pas pouvoir être recommandés. Par ailleurs, puisque la seconde technique n'est appliquée qu'aux items nécessitant une classification supplémentaire, l'hybridation en cascade est plus performante que la technique d'hybridation pondérée qui applique les deux techniques à tous les items [Burke, 2007]. Les auteurs de [Ghazanfar and Prugel-Bennett, 2010] appliquent l'hybridation en cascade en exploitant les données démographiques et les votes en plus des données sur le contenu des items pour sélectionner parmi les voisins d'un item, les plus proches d'entre eux, la pertinence des recommandations est nettement supérieure à celle obtenue par un algorithme de filtrage collaboratif basé sur les items.

Meta-level

Un autre moyen de combiner deux techniques de recommandation est d'utiliser le modèle généré par la première technique comme entrée de la seconde technique. Cette technique est différente de la technique d'augmentation de propriétés (features augmentation) dont le modèle est utilisé pour générer des données qui seront utilisées comme entrée du second algorithme.

Le système Fab [Balabanović and Shoham, 1997; Balabanović, 1998] compte parmi les premiers systèmes appliquant cette technique d'hybridation. Un autre système de recommandation de restaurants [Pazzani and Billsus, 1997] utilise une technique bayésienne naïve pour construire un modèle des préférences des utilisateurs basé sur le contenu. Une étape collaborative est appliquée pour identifier les paires d'utilisateurs. Dans [Symeonidis *et al.*, 2007] une méthode heuristique est appliquée pour définir le profil utilisateur à partir du contenu des items. Le modèle ainsi généré est par la suite utilisé dans un algorithme de filtrage collaboratif Top-N basé sur les utilisateurs. Les auteurs de [Manzato, 2012] définissent le profil de l'utilisateur à partir de l'attribut genre des films du système de recommandation MovieLens en appliquant des heuristiques et une décomposition en valeurs singulières. Le modèle des utilisateurs est utilisé dans une approche collaborative.

Très peu de travaux se sont intéressés à faire une étude comparative entre les différentes hybridations. La performance d'un système de recommandation est très liée à la nature des données qu'il traite [Herlocker *et al.*, 2004; Shani and Gunawardana, 2011], ainsi un même algorithme de recommandation peut présenter des performances en terme de précision très différentes sur deux jeux de données distincts. Les auteurs de [Ghazanfar and Prugel-Bennett, 2010] ont évalué leur proposition sur le jeu MovieLens[JeuMovieLens, 2014] et un autre jeu propriétaire représentant également des films en mesurant la MAE (voir 1.3.3 page 26) ; la MAE obtenue par le jeu MovieLens varie entre 0.73 et 0.79 alors que celle du second jeu varie entre 1.30 et 1.44, et la différence est très importante. Ainsi, la comparaison entre différents algorithmes de recommandation ne peut être exploitable que s'ils sont exécutés sur le même jeu de données. Parmi les études les plus récentes, on trouve celle effectuée par Robin Burke [Burke, 2007], dans laquelle il présente une étude comparative en expérimentant les différentes techniques d'hybridation présentées ci-dessus sur le même jeu de données issu du système de recommandation *Entrée restaurant* [Burke, 1999; Burke, 2000]. Il justifie ce choix par le fait qu'il contient, en plus des données sur l'usage et sur le contenu des items, des données démographiques et une base de connaissance. Dans cette étude, les expérimentations ont été menées sur les quatre techniques de recommandation : recommandation collaborative (deux algorithmes de filtrage collaboratif basé sur les utilisateurs, l'un utilisant la mesure de corrélation de Pearson, l'autre utilisant une autre heuristique pour calculer la similarité entre les utilisateurs), un algorithme de recommandation basée sur le contenu, un algorithme de recommandation démographique, et un algorithme de recommandation basée sur la connaissance. L'hybridation a consisté à combiner deux algorithmes issus de techniques de recommandation distinctes. Les performances de chaque technique d'hybridation ont été comparées aux algorithmes non hybrides la composant. Le premier constat à faire en analysant les résultats concerne les techniques d'hybridation *pondérée*, *switching*, *combinaison des propriétés* et *meta-level*, dont la précision des recommandations s'est vue détériorée par rapport aux algorithmes non hybrides les composant et ce pour la plupart des expérimentations menées. L'auteur justifie ce défaut de performance par le fait que ces techniques d'hybridation ne sont pas conçues pour ce type de jeu de données. Ce qui vient corroborer différentes études [Shani and Gunawardana, 2011; Bruno PRADEL, 2013] qui disent que l'efficacité d'un algorithme de recommandation est liée en partie aux données dont il dispose. Comme second constat, on remarque que dans le top 2 des meilleures performances enregistrées par chaque technique d'hybridation, le composant filtrage collaboratif est toujours présent et la combinaison filtrage collaboratif, filtrage sur le contenu est en tête. Enfin, dans le top 8 des meilleures performances, on trouve les techniques d'hybridation *cascade* et *augmentation de propriétés* dont la pertinence des recommandations est nettement supérieure à celle obtenue par l'algorithme de filtrage collaboratif basé sur les utilisateurs les

composant. Le filtrage collaboratif utilisant la corrélation de Pearson est présent dans les 4 meilleures performances. Il est à noter que les deux algorithmes de filtrage collaboratif basé sur les utilisateurs faisant l'objet des expérimentations ont affiché les meilleures performances par rapport aux autres algorithmes non hybrides expérimentés.

En analysant les résultats de cette étude, on peut en conclure que la performance d'un algorithme d'hybridation est liée à la nature des données traitées par le système de recommandation. En effet, [Lekakos and Caravelas, 2008] propose un algorithme hybride combinant un filtrage collaboratif et un filtrage basé sur le contenu en appliquant la technique du switching pour recommander des films, ce qui a permis d'améliorer la pertinence des recommandations par rapport aux algorithmes non hybrides le composant, contrairement aux résultats obtenus dans [Burke, 2007]. De même [Barragáns-Martínez *et al.*, 2010; Choi *et al.*, 2012] proposent des solutions hybrides combinant également un algorithme de filtrage collaboratif et un algorithme de filtrage sur le contenu qui affichent des performances supérieures à celles enregistrées par les algorithmes non hybrides les composant.

La plupart de ces approches sont orientées processus, il s'agit d'exécuter du filtrage collaboratif sur les résultats issus d'un filtrage sur le contenu ou vice versa. D'autres travaux ont proposé un modèle unifié basé sur les méthodes bio inspirées telles que les algorithmes génétiques [Linqi and Li, 2008; Al-Shamri and Bharadwaj, 2008], les méthodes probabilistes telles que les réseaux de neurones [Lee *et al.*, 2002; Christakou and Stafylopatis, 2005; Ren *et al.*, 2008], les réseaux bayésiens [de Campos *et al.*, 2010], le clustering [Liu *et al.*, 2010; Shinde and Kulkarni, 2012; Verma *et al.*, 2013].

Le système de recommandation hybride que nous proposons est également représenté par un modèle unifié, pouvant être considéré, selon la classification de Burke, comme étant une hybridation Meta-level. Il consiste à exploiter les informations pouvant être extraites à partir de la description des items dans un algorithme de filtrage collaboratif. L'intérêt premier est de profiter de la simplicité et de l'efficacité du filtrage collaboratif [Burke, 2007; Schafer *et al.*, 2007], tout en traitant le problème des données manquantes et celui du passage à l'échelle, en particulier, celui relatif à la détermination des communautés d'utilisateurs [Desrosiers and Karypis, 2011]. Les différentes sources d'information (structurées, semi structurées ou non structurées) sont ramenées à une structure du type attribut-valeurs. Nous nous sommes intéressés dans ce travail à modéliser l'intérêt que porte l'utilisateur aux attributs décrivant les items et leurs influences sur ses préférences en construisant un nouveau profil personnalisé : le modèle sémantique des utilisateurs. Certains travaux dont [Pazzani and Billsus, 1997; Melville *et al.*, 2002] ont défini un tel modèle mais en considérant toute l'information décrivant les items sans aucune sélection ni distinction. Dans notre approche, nous avons commencé par faire une sélection sur les attributs en ne conservant que ceux susceptibles d'influencer le choix des utilisateurs (attributs pertinents voir section 3.4 page 67). Les auteurs de [Symeonidis *et al.*, 2007; Manzato, 2012] proposent une solution traitant un attribut à la fois mais en leur appliquant le même algorithme. Afin de tenir compte des caractéristiques de chaque attribut, nous avons défini une typologie des attributs. Pour chaque catégorie d'attributs, nous proposons une approche appropriée et différents algorithmes. Un modèle sémantique des utilisateurs est alors construit pour chaque attribut sélectionné en appliquant un algorithme approprié. Toutefois, comme nous l'avons déjà mentionné, l'un des problèmes inhérent à l'exploitation de l'information décrivant les items est que celle-ci n'est pas toujours disponible en quantité suffisante [Pazzani and Billsus, 2007]. Or, le découpage par attribut permet au système de recommandation que nous proposons

de s'adapter également à la quantité d'information disponible. Ainsi, le système est capable de fournir des recommandations pertinentes même lorsque les items ne sont décrits que par un seul attribut, à condition toutefois, que ce dernier soit pertinent.

2.4 Conclusion

Nous avons présenté dans ce chapitre les techniques de recommandations personnalisées, à savoir, le filtrage collaboratif et le filtrage basé sur le contenu. Pour chaque technique, nous avons présenté les principaux algorithmes avec un intérêt particulier pour les algorithmes de filtrage collaboratif basés sur les voisins. Nous avons également présenté les différentes techniques d'hybridation identifiées dans [Burke, 2002] et une analyse de l'étude comparative présentée par Burke [Burke, 2007]. Les chapitres suivants sont consacrés à la description et à l'évaluation de notre approche que nous avons intitulée User Semantic Collaborative Filtering.

Chapitre 3

User Semantic Collaborative Filtering

Dans ce chapitre nous présentons l'architecture générale du système de recommandation hybride que nous proposons. Un rappel de la problématique est donné à la section 3.1. L'architecture générale du système que nous proposons ainsi que les défis à relever sont présentés à la section 3.2. La section 3.3 décrit et formalise les entrées du système. Tout système de recommandation personnalisé est constitué d'un composant chargé de la personnalisation et d'un composant chargé de la recommandation. L'approche que nous proposons pour la personnalisation est décrite à la section 3.4, celle relative à la recommandation fait l'objet de la section 3.5. Une synthèse résumant l'apport de notre approche est donnée à la section 3.6. Tout système de recommandation doit être soumis à une évaluation, la section 3.7 présente le jeu de données, et les critères d'évaluation que nous avons utilisés. Enfin, une conclusion résume les principales notions présentées et définies dans ce chapitre.

3.1 Problématique

Le rôle d'un système de recommandation est de proposer à l'utilisateur des items pertinents parmi un large choix de possibilités. Plusieurs techniques de recommandation existent dans la littérature parmi lesquelles on trouve celles proposant des recommandations personnalisées en définissant un profil pour chaque utilisateur. Dans ce travail, nous nous intéressons aux systèmes de recommandation personnalisés dont le profil utilisateur est défini à partir de l'analyse des usages (section 1.1.3). Le modèle utilisateur est le plus souvent décrit par la matrice des votes des utilisateurs où les lignes correspondent aux utilisateurs, les colonnes aux items et les valeurs de la matrice aux votes. La matrice des votes est une matrice creuse avec un taux de valeurs manquantes très élevé pouvant dépasser les 95% de la totalité des votes.

Le filtrage collaboratif (CF) et le filtrage basé sur le contenu (CB) sont les techniques les plus utilisées dans les systèmes de recommandation personnalisés. Le principe fondamental sur lequel repose le filtrage collaboratif est que si deux utilisateurs u et v notent n items de la même façon alors ils noteront d'autres items de façon similaire. Il reprend en fait le principe du bouche à oreille largement utilisé dans notre vie courante pour se faire recommander des produits ou des services. Le filtrage basé sur le contenu suppose que les utilisateurs sont indépendants les uns des autres et recommande à l'utilisateur courant les items similaires à ceux qu'il a appréciés par le passé. La différence entre les deux se situe au niveau des données utilisées pour faire de la recommandation. Le filtrage collaboratif n'utilise que les données issues de l'analyse des usages,

alors que le filtrage basé sur le contenu utilise, en plus, les données décrivant le contenu des items. Le contenu est utilisé pour mesurer la similarité entre les items, il est généralement décrit par du texte et dépend du type des items, si par exemple, le système recommande des chansons, alors le nom de l'artiste, le type de la musique et la durée de la chanson peuvent représenter des exemples de contenu d'items. Ceci suppose qu'une description variée des items doit être disponible, ce qui peut ne pas être le cas pour certains systèmes.

L'objectif de ce travail est de proposer une nouvelle approche d'hybridation combinant les techniques de recommandation collaboratives et celles basées sur le contenu. Nous avons vu au chapitre 2 que plusieurs techniques d'hybridation existent, toutefois, la plupart sont orientées processus, il s'agit d'exécuter du filtrage collaboratif sur les résultats issus d'un filtrage sur le contenu ou vice versa sans s'intéresser au lien pouvant exister entre les utilisateurs et le contenu des items.

Nous nous intéressons uniquement aux applications définies dans un contexte à faible risque. Dans ce genre d'application, les items se caractérisent par une faible complexité et une faible utilité. La complexité se mesure par les propriétés de l'item à prendre en considération lors de la conception d'un système de recommandation, c'est le cas par exemple des caméras digitales, des PCs, des voyages. L'utilité d'un item est positive si l'item est pertinent pour l'utilisateur, négative s'il n'est pas adapté à ses besoins et il a pris une mauvaise décision en le sélectionnant. L'utilité est faible si la perte encourue par une mauvaise recommandation n'est pas élevée pour l'utilisateur, élevée dans le cas contraire. Par exemple, dans le domaine de l'immobilier, de l'automobile ou des voyages, une mauvaise recommandation est beaucoup plus coûteuse pour l'utilisateur que dans le cas de la recommandation de films, de DVDs ou de livres. Dans un contexte à faible risque, l'utilisateur n'est pas disposé à investir beaucoup d'énergie pour recevoir une recommandation. C'est pour cette raison, qu'aucune information n'est disponible sur l'utilisateur à part un numéro permettant de l'identifier et les différentes évaluations sur les items qu'il a observés. Nous supposons également disposer de représentations textuelles décrivant les items à recommander. La problématique de notre travail peut être résumée comme suit :

Étant donné un contexte à faible risque, un ensemble d'items I décrits par un contenu textuel, un ensemble d'utilisateurs et une matrice des votes, le problème consiste à construire un nouveau système de recommandation hybride intégrant l'aspect sémantique des items dans une approche collaborative.

3.2 Proposition

3.2.1 Principe général

Le système que nous proposons est constitué d'un système de recommandation unifié intégrant les données sémantiques et les données issues de l'analyse des usages dans un algorithme de recommandation personnalisée hybride. L'idée est de définir un nouveau profil utilisateur construit à partir des données sémantiques sur les items et des données d'usage. Ce profil que nous appellerons *Modèle Sémantique des Utilisateurs (MSU)* définit les préférences des utilisateurs pour les données décrivant les items à partir de leurs appréciations pour ces mêmes items.

Le modèle sémantique des utilisateurs (MSU) définissant un nouveau profil pour les utilisateurs est utilisé pour déterminer des communautés d'utilisateurs en se basant sur leurs apprécia-

tions pour le contenu des items.

Nous avons vu au chapitre 2, que parmi les approches collaboratives existantes, les algorithmes de filtrage collaboratif basés sur les plus proches voisins [Desrosiers and Karypis, 2011] (appelés aussi basés sur la mémoire [Breese *et al.*, 1998] ou basés sur des heuristiques [Adomavicius and Tuzhilin, 2005]) jouissent d'une grande popularité en raison de leur simplicité, de leur efficacité, et de leur capacité à produire des recommandations personnalisées pertinentes.

L'idée est de profiter de l'efficacité et de la simplicité de ces algorithmes pour faire de la recommandation en utilisant le modèle sémantique des utilisateurs (MSU) pour déterminer les plus proches voisins de l'utilisateur courant.

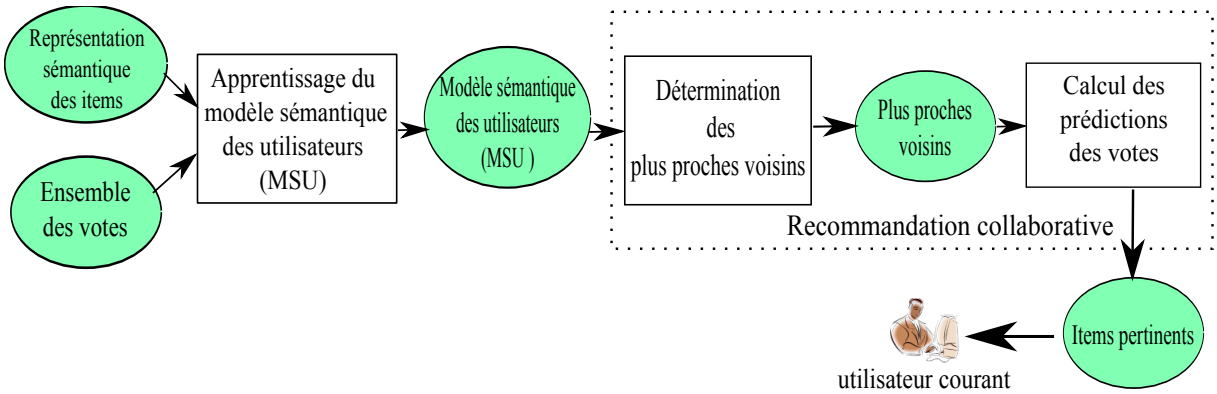


FIGURE 3.1 – Notre approche : User Semantic Collaborative Filtering (USCF).

La figure 3.1 illustre l'architecture générale de notre système, que nous avons appelé User Semantic Collaborative Filtering (USCF). Il est constitué de deux composants : *l'apprentissage du modèle sémantique des utilisateurs* et *la recommandation* des items pertinents à l'utilisateur courant. Le premier a pour objectif l'apprentissage du profil personnalisé des utilisateurs *le Modèle Sémantique des Utilisateurs (MSU)*, en inférant les préférences des utilisateurs pour le contenu des items à partir de leurs appréciations pour les items. Le second se charge de recommander les items les plus pertinents à l'utilisateur courant en calculant la prédiction des votes des items qui lui sont inconnus. La prédiction des votes est calculée à partir de ses plus proches voisins en appliquant un algorithme de filtrage collaboratif basé sur les utilisateurs (voir section 2.1.1.1). La similarité entre les utilisateurs est déterminée à partir du Modèle Sémantique des Utilisateurs (MSU).

3.2.2 Les défis

Parmi les défis auxquels sont confrontés les algorithmes de filtrage collaboratif basé sur les voisins, on trouve essentiellement le problème du passage à l'échelle [Luo *et al.*, 2012] et le problème des données manquantes [Desrosiers and Karypis, 2011]. En plus de ces deux problèmes, les systèmes basés sur le contenu doivent faire face au problème de sur-spécialisation qui se traduit par le manque de diversité au niveau des recommandations proposées et la nécessité de disposer

d'une description variée sur les items à recommander, ce qui n'est pas toujours le cas [Lops *et al.*, 2011; Bobadilla *et al.*, 2013].

Les données manquantes : la sensibilité aux données manquantes est un problème commun à la majorité des systèmes de recommandation [Adomavicius and Tuzhilin, 2005], dû essentiellement au fait que les utilisateurs notent très peu d'items. Pour les algorithmes de filtrage collaboratif basé sur les voisins, un utilisateur ne peut recevoir des recommandations que si le système arrive à identifier la communauté d'utilisateurs avec lesquels il partage les mêmes goûts. Or, le calcul des similarités exige une paire d'utilisateurs ayant évalué un certain nombre d'items en commun. Dans notre approche, les similarités seront calculées à partir du modèle sémantique des utilisateurs (MSU) et non pas à partir de la matrice des votes comme c'est le cas des algorithmes de filtrage collaboratif basés sur les voisins. Le Modèle sémantique des utilisateurs (MSU) doit par conséquent réduire le taux de données manquantes pour être capable de déterminer le voisinage de chaque utilisateur. Cette condition est primordiale pour proposer des recommandations pertinentes à la plupart des utilisateurs.

Le problème du passage à l'échelle : pour les algorithmes de filtrage collaboratif basé sur les voisins, le problème de la complexité des calculs est essentiellement dû au coût élevé imputé à la détermination des voisins surtout pour des applications de e-commerce avec un nombre très élevé d'items [Sarwar *et al.*, 2001]. Plusieurs approches ont été définies pour faire face à ce défi. Parmi les solutions proposées on trouve le calcul des similarités *offline* [Linden *et al.*, 2003] en ne stockant que les k plus proches voisins, la complexité en espace mémoire se trouve alors largement réduite ; la factorisation et la réduction de la dimension [Koren, 2008; Takacs *et al.*, 2008; Canny, 2002] ; l'application de méthodes issues de la fouille des données telles que les modèles probabilistes [Breese *et al.*, 1998] ou la classification non supervisée [Castagnos, 2008].

Description variée sur les items : la présence et la variété des données sémantiques est un facteur important dans la pertinence des recommandations fournies par un système de recommandation basé sur le contenu. Pour ne pas être tributaire de la quantité des données sémantiques disponibles, notre système de recommandation doit permettre de recommander des items pertinents même lorsque très peu de données décrivent des items.

Le problème de la diversité : notre système doit garantir des recommandations diversifiées comme c'est le cas des techniques de recommandations collaboratives basées sur les voisins, particulièrement celles basées sur les utilisateurs.

Les trois sections suivantes décrivent respectivement les entrées du système (section 3.3), le composant chargé de l'apprentissage du modèle sémantique des utilisateurs (section 3.4) et celui chargé de la recommandation (section 3.5). La section 3.6 décrit comment les défis cités ci-dessus sont traités par notre approche.

3.3 Formalisation des données en entrée

Comme le montre la figure 3.1, le système admet en entrée des données issues de l'analyse des usages et des données textuelles décrivant le contenu des items. Dans cette section, nous décrivons et formalisons les données en entrée du système. Les formalismes présentés dans cette section seront utilisés tout au long de ce manuscrit et particulièrement dans les chapitres 4 et 5.

3.3.1 Les données issues de l'analyse des usages

L'analyse des usages a pour objectif d'extraire les préférences des utilisateurs à partir des différentes interactions qu'ils ont avec le e-service. Dans notre cas, on suppose que les préférences des utilisateurs sont représentées par des votes numériques définis sur une échelle de valeur spécifique. Il peuvent être explicites ou implicites (voir section 1.1.3).

Nous reprenons les notations utilisées dans la section 1.2.2. Nous notons par U l'ensemble des utilisateurs du e-service, par I l'ensemble de tous les items pouvant être recommandés, par V l'ensemble des valeurs possibles d'un vote, par U_i l'ensemble des utilisateurs ayant noté l'item i et par I_u l'ensemble des items notés par l'utilisateur u. Un utilisateur donné n'attribue, au plus, qu'un seul vote $v_{ui} \in V$ à un item i de I. On note également par *nulle* la valeur associée à une donnée manquante. La valeur *nulle* associée à un couple (u, i) signifie que l'item i est inconnu de l'utilisateur u. Par ailleurs, on suppose que le résultat de toute opération ayant pour opérande *nulle* est égal à *nulle*. On définit la fonction de vote des utilisateurs δ_u par :

$$\forall u \in U\ \delta_u : i \in I \longmapsto \delta_u(i) \in V \cup \{nulle\} = \begin{cases} v_{ui}\ \ si\ \ i \in I_u. \\ nulle\ \ sinon. \end{cases} \tag{3.1}$$

La matrice des votes des utilisateurs définie par $M_v = (\delta_u(i))_{(u=1..|U|,\, i=1..|I|)}$ représente les données en entrée du système issues de l'analyse des usages.

On définit le profil usage d'un utilisateur u par le vecteur PUU_u (voir formule (3.2)) et par PUI_i (voir formule (3.3)) le profil usage de l'item i. PUU_u n'est autre que la ligne d'indice u de la matrice des votes M_v, et PUI_i est la colonne d'indice i de M_v.

$$PUU_u = (\delta_u(i))_{(i=1...|I|)} \tag{3.2}$$

$$PUI_i = (\delta_i(u))_{(u=1...|U|)} \tag{3.3}$$

où δ_i est la fonction de vote de l'item i définie par la formule (3.4).

$$\delta_i : u \in U \longmapsto \delta_i(u) \in V \cup \{nulle\} = \begin{cases} v_{ui}\ \ si\ \ u \in U_i. \\ nulle\ \ sinon. \end{cases} \tag{3.4}$$

On définit également le profil usage ajusté de l'item i par le vecteur $PUIa_i$:

$$PUIa_i = (\delta_i(u) - \overline{v_u})_{(u=1..|U|)} \tag{3.5}$$

Le profil ajusté donné par la formule (3.5) permet d'éliminer la variation entre les jugements des utilisateurs, tous les utilisateurs ne notent pas les items de la même façon, il y a des utilisateurs qui sont assez stricts et d'autres qui notent de manière plus large. Ainsi, chaque vote de u est centré par rapport à la moyenne de ses votes $\overline{v_u}$. La moyenne des votes d'un utilisateur u est

donné par la formule (2.1).

Pour résumer, un utilisateur u est modélisé par son profil usage PUU_u et un item est modélisé, soit par son profil usage PUI_i, soit par son profil usage ajusté $PUIa_i$.

3.3.2 Les données issues du contenu des items

Le contenu textuel d'un item peut être représenté de trois manières différentes comme les ont définies les auteurs de [Pazzani and Billsus, 2007] : *structurée*, *semi-structurée* ou *non structurée*. Dans une représentation structurée, tous les items sont décrits par le même petit nombre d'attributs et les valeurs prises par chaque attribut sont connues. Pour une représentation non structurée, l'item est décrit par un texte libre (la description d'un restaurant, le résumé d'un article de journal,...), dans une représentation semi-structurée, un item est à la fois décrit par des attributs et par du texte libre. Nous reprenons ici cette définition pour modéliser l'aspect sémantique des items.

3.3.2.1 Items structurés

Dans une représentation structurée, un item est décrit par un ensemble d'attributs. Un attribut est défini dans un domaine de valeurs, il peut être de type numérique ou de type catégoriel. Un attribut de type numérique peut être continu (c'est-à-dire ses valeurs appartiennent à un sous-ensemble infini de l'ensemble $\mathbb{R}$) ou discret (ses valeurs appartiennent à un sous-ensemble fini de l'ensemble $\mathbb{N}$). Le salaire, l'âge ou le poids sont des exemples d'attributs continus, alors que le nombre de produits achetés est un exemple d'attribut discret. Il est possible cependant de passer des données à valeurs continues à des données à valeurs discrètes en effectuant une discrétisation. La discrétisation consiste à effectuer un découpage de l'ensemble des valeurs continues en des tranches afin d'obtenir un nombre fini de valeurs possibles. Le type catégoriel représente toutes les valeurs non numériques. Les attributs de type catégoriel sont des attributs dont l'ensemble des valeurs est fini. Ces valeurs sont alphanumériques, comme par exemple le numéro d'une carte d'identité nationale, le titre d'un film, la couleur des yeux ou l'évaluation d'un client. Si les différentes valeurs peuvent être classées, alors on parle d'attribut catégoriel ordinal comme par exemple pour l'évaluation d'un client (*mauvais*, *satisfaisant*, *bien*, *excellent*), dans le cas contraire on parle d'attribut catégoriel nominal [Kassab, 2009]. Ainsi, ce qui distingue les attributs numériques des autres attributs est qu'il est possible de leur appliquer des calculs mathématiques comme par exemple la moyenne ou la somme.

id	Titre	Genre
1	Les visiteurs	Comédie
2	Titanic	Romantique, Drame

Table 3.1 – Représentation structurée d'items

Un item est représenté par un modèle vectoriel construit à partir de sa représentation sémantique. Dans un modèle vectoriel (VSM) [26], un item est représenté par un vecteur de poids, où chaque poids est associé à un terme, et un poids est une valeur numérique indiquant soit la présence, soit la fréquence, soit l'importance du terme dans l'item. Dans le cas d'une représentation

26. Vector Space Model

structurée, un terme correspond à la valeur d'un attribut (après avoir appliqué une discrétisation des attributs à valeurs continues), le poids est représenté par une valeur booléenne indiquant la présence ou pas de la valeur de l'attribut dans l'item. Prenons l'exemple des films décrits dans le tableau 3.1. Les films sont décrits par deux attributs : *titre* et *genre* , l'*id* est un numéro permettant d'identifier les films. Les valeurs de l'attribut *genre* sont {*comédie*, *romantique* et *drame*} et les valeurs de l'attribut *titre* sont {*Les visiteurs*, *Titanic* }.

Le tableau 3.2 donne la représentation en modèle vectoriel relative aux items du tableau 3.1.

id	Les visiteurs	Titanic	Comédie	Romantique	Drame
1	1	0	1	0	0
2	0	1	0	1	1

TABLE 3.2 – Représentation VSM des items

Dans tout ce qui suit, nous utiliserons le terme *descripteur* pour représenter la valeur d'un attribut, ainsi *comédie* est un descripteur de l'attribut *genre*. On note par F l'ensemble des descripteurs associés aux items de I et par F_i le sous ensemble des descripteurs de F décrivant l'item i.
Le poids d'un descripteur f_l dans un item i peut être représenté par la fonction $présence_i$ définie par la formule (3.6).

$$présence_i : f_l \in F \longmapsto présence_i(f_l) \in \{0,1\} = \begin{cases} 1 \ \ si \ \ f_l \in F_i. \\ 0 \ \ sinon. \end{cases} \tag{3.6}$$

Toutefois, il peut exister des attributs dont le poids représente la fréquence d'un descripteur dans un item comme c'est le cas des annotations. Le poids d'une annotation correspondra dans ce cas au nombre de fois qu'elle a été utilisée pour annoter un item. Dans ce cas, le poids est défini par la fonction $fréquence_i()$ telle que le définit la formule (3.7).

$$fréquence_i : f_l \in F \longmapsto fréquence_i(f_l) \in \mathbb{N} = \begin{cases} n \in \mathbb{N}^* : \ nombre \ d'occurrences \ de \ f_l \ dans \ i. \\ 0 \ \ si f_l \notin F_i. \end{cases} \tag{3.7}$$

En résumé, le poids d'un descripteur f_l dans un item i de I est égal soit à $présence_i(f_l)$, soit à $fréquence_i(f_l)$ s'il l'on dispose du nombre d'occurrences de F_l dans i.

3.3.2.2 Items non structurés

Lorsque les items sont décrits par du texte libre, il faut au préalable les transformer en une représentation structurée. Pour structurer des items, une indexation est appliquée à l'ensemble des items. L'indexation, utilisée dans la recherche et filtrage d'information pour le traitement des documents [Salton, 1989], est l'opération consistant à extraire les mots les plus pertinents qui caractérisent le contenu d'un ensemble de documents. A la suite de l'étape d'indexation, chaque item sera modélisé par une représentation vectorielle (VSM) dans laquelle il sera décrit par un vecteur de poids, où chaque poids sera associé à un mot. Le poids est une valeur numérique représentant la fréquence du mot dans l'item. Par souci de clarté, nous utiliserons le terme *descripteur* pour désigner un mot issu de l'indexation des items. On définit par F l'ensemble des descripteurs issus de l'indexation des items de I, et par F_i le sous ensemble des descripteurs de F décrivant l'item i. Le poids, $poids_i()$ d'un descripteur $f_l \in F$ dans un item i est donné par la

fonction $fréquence_i()$ définie également par la formule (3.7).

Dans toute la suite de ce manuscrit, et pour des raisons de clarté, nous utiliserons le terme attribut pour désigner également les données non structurées décrivant les items. Par exemple, le synopsis d'un film sera considéré comme un attribut et les mots issus de l'indexation de l'ensemble des synopsis seront considérés comme les descripteurs de cet attribut.

3.3.2.3 Profil sémantique des items

On note par A un attribut, par $F_A = \{f_1, ..., f_l, ..., f_{|F_A|}\}$ la liste de ses descripteurs décrivant les items de I et par F_{A_i} le sous ensemble des descripteurs de F_A décrivant l'item i. On définit le profil sémantique par attribut de l'item i pour l'attribut A, PSA_{i_A}, par le vecteur défini par le poids des descripteurs de F_A.

$$PSA_{i_A} = (poids_i(f_l))_{(l=1...|F_A|)} \tag{3.8}$$

On définit par ailleurs, la Matrice Sémantique des Items par Attribut pour l'attribut A, $MSIA_A$, comme étant la transposée de la matrice construite à partir des PSA_{i_A} de tous les items de I.

$$MSIA_A = ((PSA_{i_A})_{(i=1...|I|)})^t. \tag{3.9}$$

Ainsi, le profil sémantique par attribut de l'item i, PSA_{i_A} correspond à la ligne d'indice i de la matrice $MSIA_A$. Le profil sémantique d'un item, PSI est alors composé de l'ensemble de ses profils sémantiques par attribut. Si l'item est non structuré, alors son profil sémantique est constitué du profil sémantique issue de l'indexation de l'ensemble des items. Si l'item est structuré, alors son profil sémantique est constitué de l'ensemble des profils sémantiques par attribut de l'ensemble des attributs décrivant les items. Si l'item est semi-structuré, son profil sémantique est composé des profils issus de la représentation structurée ainsi que celui issu de la représentation non structurée.

3.3.3 Synthèse

En entrée du système, les données issues de l'analyse des usages sont représentées par la matrice des votes des utilisateurs Mv. Les données sémantiques sur les items sont textuelles pouvant être structurées, non structurées ou semi-structurées. Un utilisateur u est défini par son profil usage PUU_u (ligne d'indice u de la matrice des votes). Un item i est défini par son profil usage PUI_i (colonne d'indice i de la matrice des votes) et ses différents profils sémantiques par attribut PSA_{i_A}, A étant un attribut décrivant les items de I. Si les items sont décrits par une représentation structurée alors A représente l'un des attributs décrivant les items. Si les items sont décrits par du texte libre, alors A représente le champ décrit par ce texte (le résumé, le synopsis, le texte d'un article ...).

3.4 Apprentissage du modèle sémantique des utilisateurs

L'idée est d'inférer les préférences des utilisateurs pour les descripteurs des items à partir des données issues de l'analyse des usages et de la représentation sémantique des items. Plusieurs travaux ont déjà traité ce sujet [Symeonidis *et al.*, 2007; Sen *et al.*, 2009; Manzato, 2012]. Dans la plupart de ces travaux, le profil sémantique des utilisateurs est défini dans le même espace que

celui du profil sémantique des items. Aucune distinction n'est faite entre les attributs décrivant les items. Or, tous les attributs décrivant l'ensemble des items à recommander n'ont pas la même importance auprès des utilisateurs. En effet, il peut y avoir des attributs plus pertinents pour les utilisateurs que d'autres. Un attribut est dit pertinent pour les utilisateurs s'il est discriminatif dans la sélection des items. Par exemple, pour un système de recommandation de livres, il peut s'agir de l'auteur et/ou de la catégorie du livre, pour un système de recommandation de films cela peut être le genre et/ou le réalisateur du film. La liste des attributs pertinents est liée au domaine d'activité du système de recommandation.

Nous nous basons sur cette hypothèse pour construire le modèle sémantique des utilisateurs. Ainsi, seuls les attributs pertinents seront considérés. Par ailleurs, pour chaque attribut pertinent nous construisons un *Modèle Sémantique des Utilisateurs par Attribut* (MSUA). Le modèle sémantique des utilisateurs (MSU) est alors déduit par fusion des différents modèles sémantiques des utilisateurs par attribut (MSUA) associés au groupe d'attributs pertinents.

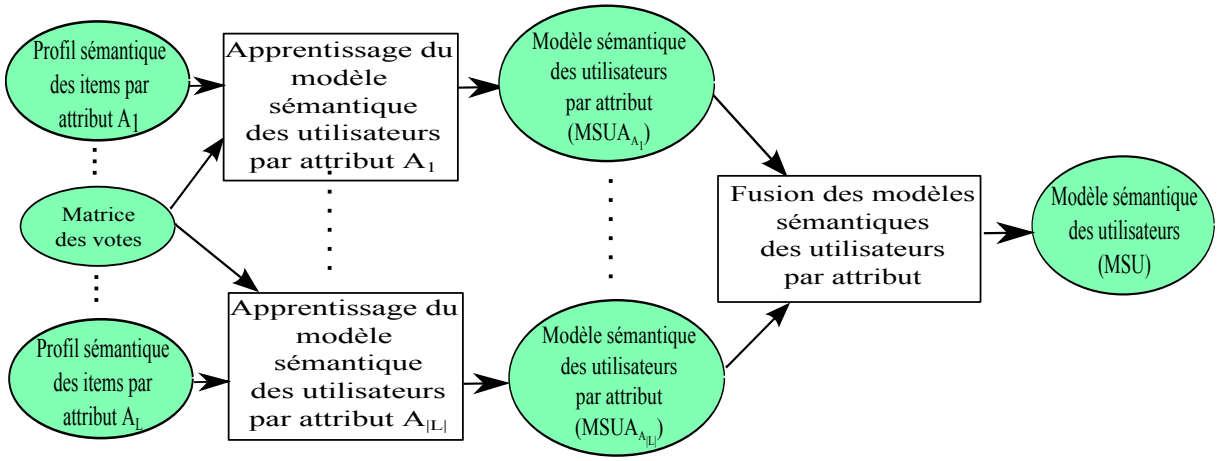


FIGURE 3.2 – Apprentissage du modèle sémantique des utilisateurs

La figure 3.2 illustre les étapes de construction du modèle sémantique des utilisateurs.

1. Définir une liste d'attributs pertinents : la liste L des attributs pertinents est liée au domaine d'application. Nous supposons à ce niveau que la liste est déterminée par des experts du domaine.
2. Construire, pour chaque attribut pertinent, le *Modèle Sémantique des Utilisateurs par Attribut (MSUA)* en inférant les préférences des utilisateurs pour les descripteurs de l'attribut à partir de leurs profils usage et du Modèle Sémantique des Items par Attribut (MSIA).
3. Fusionner les différents Modèles Sémantiques des Utilisateurs par Attribut (MSUA) ainsi obtenus pour générer le Modèle Sémantique des Utilisateurs (MSU).

3.4.1 Algorithme

L'algorithme 3.1 décrit le principe de construction du modèle sémantique des utilisateurs (MSU). Le profil usage des utilisateurs est défini par la matrice des votes Mv, le profil sémantique des items est défini par l'ensemble des MSIA de la liste L des attributs pertinents

représentant l'ensemble des items à recommander. Pour chaque attribut pertinent A de la liste L, on construit le modèle sémantique des utilisateurs par attribut (MSUA) correspondant modélisé par la matrice $Q_{(A_{|U|},|D_A|)}$ (ligne 3 de l'algorithme 3.1), D_A étant la liste des descripteurs de l'attribut A représentés dans la dimension des utilisateurs. Chaque descripteur de D_A est défini par un vecteur colonne de la matrice Q_A. Une fois tous les modèles sémantiques des utilisateurs construits, la matrice Q modélisant le modèle sémantique des utilisateurs est construite par concaténation horizontale des différentes matrices Q_A (ligne 7 de l'algorithme 3.1).

Il est à noter que pour chaque attribut pertinent A, la construction du modèle sémantique des utilisateurs par attribut ($MSUA_A$) associé à A est totalement indépendante des autres attributs. Ce qui signifie que les constructions des différents MSUA sont des processus totalement indépendants et dont l'exécution peut se faire en parallèle (ligne 3 de l'algorithme 3.1).

Algorithme 3.1 : Apprentissage du Modèle Sémantique des Utilisateurs (MSU)

Données :
L : liste des attributs pertinents,
Mv : matrice des votes,
$MSIA_{A \in L}$ les matrices sémantiques des items par attribut pour tous les attributs de L.
Résultat :
Q : *la matrice modélisant MSU*

```
1 début
2 |  Q = matrice vide
3 |  pour chaque (A ∈ L) faire // Cette boucle est totalement parallélisable
4 |  |  Q_A = Construction_MSUA(Mv, MSIA_A)// exécutée en offline
5 |  fin
6 |  pour chaque (A ∈ L) faire
7 |  |  Q = Concaténation_horizontale(Q, Q_A)
8 |  fin
9 fin
```

3.4.2 Classification des attributs

En observant les descripteurs des attributs, nous avons constaté que tous les attributs n'ont pas la même relation vis à vis des items qu'ils décrivent. Pour cela nous avons défini une nouvelle typologie composée de deux classes d'attributs : les *attributs indépendants* et les *attributs dépendants*.

Définition *Attributs indépendants :* caractérisés par un nombre stable de valeurs potentielles indépendamment du nombre d'items. Cette typologie d'attribut est généralement utilisée pour découper les items en groupe d'appartenance. Le *sexe*, l'*état civil*, le *pays de naissance*, le *genre* d'un film ou la *catégorie* d'un livre sont des exemples d'attributs indépendants. On voit bien que le nombre de leurs valeurs (descripteurs) possibles ne varie avec le nombre d'items qu'ils décrivent.

Définition *Attributs dépendants :* caractérisés par un nombre de valeurs potentielles variant selon le nombre d'items qu'ils décrivent. Les *acteurs*, les *réalisateurs* d'un système de recommandation de films, l'*artiste* d'un système de recommandation de musique sont des exemples

d'attributs dépendants. Ainsi, plus une base contiendra de films, plus le nombres d'acteurs sera élevé, de même pour les auteurs d'une base de données de livres.

L'adaptation du profil sémantique des utilisateurs par attribut peut varier en fonction de la classe de l'attribut. En effet, pour les attributs dépendants, l'ajout de nouveaux items peut influer sur le modèle sémantique des utilisateurs par attribut et nécessiter par conséquent des adaptations assez fréquentes. Pour les attributs indépendants, vu que le nombre de leurs descripteurs est stable, le profil sémantique des utilisateurs par attribut est moins sensible à l'arrivée de nouveaux items.

3.4.3 Apprentissage du modèle sémantique des utilisateurs par attribut

Définir le profil sémantique par attribut d'un utilisateur u revient à déterminer l'importance de chaque descripteur f_l de l'attribut A pour l'utilisateur u à partir de ses appréciations pour les items décrits par le descripteur f_l. Formellement, il s'agit de déterminer une agrégation sur tous les items notés par u et décrits par le descripteur f_l de A.

$$\forall u \in U, \;\; q_A(u, f_l) = \underset{i \in I_{f_l} \cap I_u}{aggr} v_{ui} \tag{3.10}$$

I_{f_l} est l'ensemble des items décrits par le descripteur f_l, $q_A(u, f_l)$ définit l'importance du descripteur f_l pour l'utilisateur u. Par exemple, dans un système de recommandation de films, l'importance du descripteur *action* de l'attribut *genre* pour un utilisateur u est égale à :

$$q_{genre}(u, action) = \underset{i \in I_{action} \cap I_u}{aggr} v_{ui}. \tag{3.11}$$

Le problème revient alors à déterminer la fonction d'agrégation. Elle peut être définie soit par une simple fonction telle que la moyenne, soit en utilisant des fonctions mathématiques plus élaborées, soit en utilisant des fonctions spécifiques par utilisateur issues de l'apprentissage automatique et de la théorie d'approximation.

Pour chaque classe d'attributs nous proposons des approches différentes pour construire le modèle sémantique par attribut. Ainsi pour les attributs indépendants, nous proposons des approches issues de l'apprentissage automatique, pour les attributs dépendants nous proposons des approches issues du domaine de la recherche et du filtrage d'information [Salton, 1989; Bezdek, 1981]. Une description détaillée des approches proposées pour la construction des modèles sémantiques des utilisateurs par attribut (MSUA), pour les attributs indépendants est donnée dans le chapitre 4 et pour les attributs dépendants est donnée dans le chapitre 5.

Le modèle sémantique des utilisateurs par attribut ($MSUA$) pour un attribut pertinent A est alors modélisé par une matrice $Q_{A_{|U|,|D_A|}}$, les utilisateurs en ligne et les descripteurs D_A de l'attribut A en colonne défini par l'équation 3.12.

$$Q_A = (q(u, d))_{(u=1..|U|, d=1..|D_A|)} \tag{3.12}$$

$q(u, d)$ représente l'importance du descripteur d pour l'utilisateur u. Pour les attributs dépendants, et vu que le nombre de leurs descripteurs est élevé pouvant même dépasser le nombre d'items, la dimension de l'espace D_A doit être réduite par rapport à celle de F_A.

3.5 Recommandation

Comme nous l'avons déjà dit, le modèle sémantique des utilisateurs (MSU) est utilisé pour déterminer les similarités entre les utilisateurs. Les similarités sont utilisées pour déterminer les N plus proches voisins de l'utilisateur courant dans un algorithme de filtrage collaboratif basé sur les utilisateurs [Resnick *et al.*, 1994].

3.5.1 Détermination des plus proches voisins

Le profil sémantique de l'utilisateur u (PSU_u) est représenté par la ligne d'indice u de la matrice Q modélisant le modèle sémantique des utilisateurs (MSU). Le calcul de la similarité entre deux utilisateurs revient alors à calculer la corrélation entre leurs deux profils sémantiques. Nous avons vu dans la section 2.1.1.5 les mesures les plus répandues pour calculer la similarité entre les utilisateurs dans un algorithme de filtrage collaboratif basé sur les voisins. La mesure de corrélation de Pearson est en tête de liste. Le Cosinus est également utilisé, mais on lui reproche de ne pas faire la distinction entre les jugements des utilisateurs, ce qui n'est pas le cas pour la mesure de Pearson puisque elle permet d'ajuster les différentes évaluations.

Dans notre cas, le profil sémantique de l'utilisateur u (PSU_u) modélise l'importance des descripteurs (éventuellement latents) pour l'utilisateur u. L'ajustement des évaluations est effectué lors de l'apprentissage des modèles sémantiques des utilisateurs par attributs pour les attributs pertinents. Pour cette raison, nous avons opté pour le Cosinus comme mesure de calcul de la corrélation entre deux utilisateurs u et v comme le définit la formule (3.13).

$$sim(u,v) = \cos(\vec{PSU_u}, \vec{PSU_v}) = \frac{\vec{PSU_u} \bullet \vec{PSU_v}}{\left\|\vec{PSU_u}\right\| \left\|\vec{PSU_v}\right\|} \tag{3.13}$$

3.5.2 Calcul des prédictions des votes

Pour calculer la prédiction de la valeur du vote d'un item i non observé par l'utilisateur courant u_a, nous avons appliqué la formule (2.5) en ne conservant que les N plus proches voisins. La similarité entre u et u_a étant déterminée dans notre cas à partir de leur profils sémantiques en appliquant la formule (3.13).

Le calcul des prédictions est exécuté en temps réel lors de la connexion de l'utilisateur courant u_a au e-service. Ainsi, les mises à jours de son profil usage PUU_{u_a} seront immédiatement prises en compte lors du calcul des prédictions.

En fonction des besoins de l'application, il est possible de recommander à l'utilisateur courant u_a, soit une liste Top-N des N items les plus pertinents, c'est-à-dire ayant les plus grandes valeurs de prédiction, soit de recommander tous les items triés par ordre décroissant de pertinence accompagnés de leur prédiction. Un item i est jugé pertinent pour un utilisateur u, si la valeur de sa prédiction ($pred(u,i)$) est supérieure ou égale à un seuil donné $v_s \in V$. Le seuil v_s est une valeur parmi les valeurs possibles d'un vote.

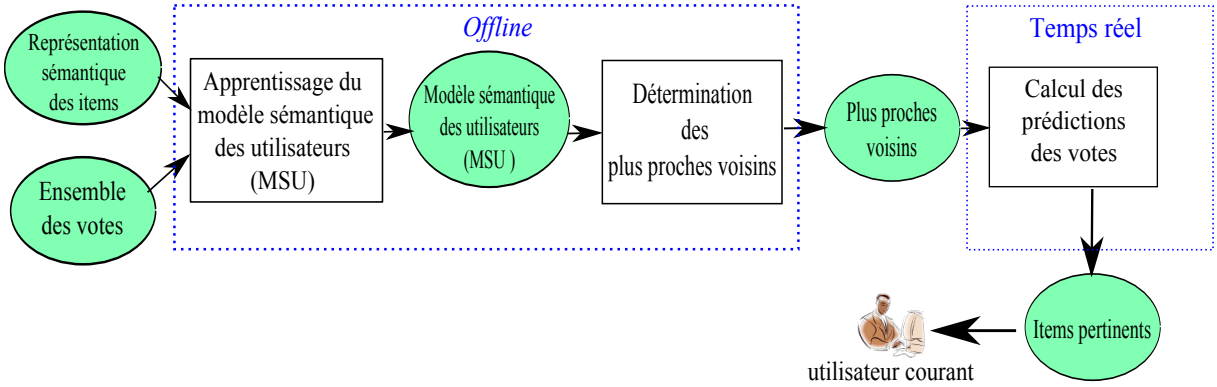


FIGURE 3.3 – User Semantic Collaborative Filtering (USCF)

3.6 Synthèse de USCF

Le système de recommandation hybride, User Semantic Collaborative Filtering (USCF), que nous proposons est représenté par un système unifié intégrant les informations sémantiques pouvant être extraites à partir de la description des items dans un algorithme de filtrage collaboratif. Nous nous sommes intéressés dans ce travail à modéliser l'intérêt que porte l'utilisateur au contenu décrivant les items et leurs influences sur ses préférences en modélisant un nouveau profil utilisateur : le Modèle Sémantique des Utilisateurs. Ce modèle est par la suite utilisé dans un algorithme de recommandation collaborative basée sur les utilisateurs pour déterminer les plus proches voisins de chaque utilisateur.

La solution que nous proposons est une approche basée sur un modèle. L'apprentissage du modèle sémantique des utilisateurs est effectué en mode *offline*, le calcul des prédictions des votes est quant à lui effectué en temps réel lors de l'interaction de l'utilisateur courant avec le e-service (voir figure 3.3). Pour réduire la complexité temporelle du calcul de la prédiction des votes, la détermination des plus proches voisins de chaque utilisateur est effectuée également en mode *offline* en ne conservant que les k plus proches d'entre eux.

L'apprentissage du Modèle Sémantique des Utilisateurs (MSU) est décomposé en L processus totalement parallélisables exécutés en mode *offline*. Chaque processus correspondant à l'apprentissage d'un Modèle Sémantique des Utilisateurs par Attribut (MSUA) relatif à un attribut pertinent.

Deux classes d'attributs ont été identifiées, les attributs dépendants et les attributs indépendants en fonction du lien existant entre le nombre de leurs descripteurs et le nombre d'items qu'ils décrivent. Pour chaque classe d'attributs, nous proposons des méthodes appropriées pour l'apprentissage du Modèle Sémantique des Utilisateurs par Attribut (MSUA). Les chapitres 4 et 5 présentent respectivement nos solutions pour les attributs indépendants et pour les attributs dépendants.

3.6.1 Points forts

Comme nous l'avons mentionné plus haut, notre approche doit faire face au problème de la sensibilité aux données manquantes et au problème du passage à l'échelle liés à l'utilisation de l'algorithme de filtrage collaboratif basé sur les voisins.

Données manquantes : dans sa version standard, l'algorithme de filtrage collaboratif utilise la matrice des votes (Mv) pour calculer la similarité entre les utilisateurs. La matrice des votes étant une matrice creuse avec un taux de valeurs manquantes avoisinant les 95% (comme c'est le cas du jeux de données MovieLens [MovieLens, 2014]), il n'est par conséquent pas possible de calculer la similarité entre tous les utilisateurs, d'où l'incapacité de l'algorithme à fournir des recommandations pour les utilisateurs n'ayant pas d'items en commun avec assez d'utilisateurs. Dans notre approche, nous utilisons le modèle sémantique des utilisateurs, représenté par la matrice Q, pour calculer la similarité entre les utilisateurs. Le problème des données manquantes est alors traité en augmentant la densité de la matrice Q, et ce, en réduisant la dimension de l'espace D des descripteurs, c'est-à-dire en réduisant le nombre de colonnes de Q. Afin de conserver l'indépendance des différents Modèles Sémantiques des Utilisateurs par Attribut (MSUA), réduire la dimension du Modèle Sémantique des Utilisateurs (MSU) revient à réduire la dimension de chaque MSUA le composant, c'est-à-dire réduire le nombre de descripteurs décrivant l'attribut qui lui est associé. Différentes méthodes de réduction sont proposées, principalement pour les attributs dépendants (voir chapitre 5). L'augmentation de la densité du modèle sémantique des utilisateurs a pour effet de réduire le taux de données manquantes permettant ainsi, la détermination des similarités entres des utilisateurs ayant très peu d'items en commun.

Passage à l'échelle : il est traité à quatre niveaux lors de la construction du profil sémantique des utilisateurs. Premièrement, en ne sélectionnant que les attributs pertinents. Deuxièmement, par le découpage du profil sémantique des items par attribut, ainsi, la construction du Modèle Sémantique des Utilisateurs (MSU) est décomposée en L processus, chacun construisant un Modèle Sémantique des Utilisateurs par Attribut (MSUA). Troisièmement, en parallélisant le processus d'apprentissage du MSU, chaque MSUA est construit de manière indépendante. Quatrièmement, en réduisant la dimension du modèle sémantique des utilisateurs par attribut. Par ailleurs, la complexité des calculs due aux calculs des similarités entre les utilisateurs est également traitée, d'une part, en réduisant la dimension du modèle sémantique des utilisateurs (MSU), et d'autre part, en ne conservant que les k plus proches voisins de chaque utilisateur. Ainsi, lors de l'étape de calcul des prédictions des votes de l'utilisateur courant, seuls ses k plus proches voisins seront pris en compte.

Diversité des approches : il est possible d'appliquer des méthodes différentes pour la construction des Modèle sémantique des Utilisateurs par Attribut (MSUA) en appliquant pour chaque attribut ou type d'attribut la méthode la plus adéquate.

Adaptation et mise à jour : l'adaptation et la mise à jour du modèle sémantique des utilisateurs (MSU) peuvent se faire de façon partielle. La fréquence de l'adaptation du modèle sémantique des utilisateurs par attribut (MSUA) peut varier en fonction de la méthode choisie et de la nature de l'attribut.

3.7 Évaluation

Lorsqu'un système de recommandation est développé, il est important d'être en mesure d'évaluer son fonctionnement et sa capacité à répondre aux objectifs qui lui ont été fixés. Un algorithme de recommandation a pour objectif d'améliorer l'utilité d'un e-service vis-à-vis de ses utilisateurs en augmentant leur satisfaction. Ainsi, mesurer la satisfaction des utilisateurs en terme de recommandation représente un critère d'évaluation important pour tout algorithme de recommandation. Pour évaluer notre approche, nous avons opté pour le mode d'évaluation offline (voir section 1.3). Ce mode d'évaluation consiste à tester le système sur des données offline à la place de données en ligne. En plus du fait qu'elle soit plus simple à mettre en œuvre que l'évaluation en ligne, l'évaluation offline permet également de comparer les performances de plusieurs algorithmes de recommandations de façon objective.

Le principe de fonctionnement de l'évaluation offline est inspiré de l'approche utilisée dans le domaine de la fouille de données. Le jeu de données initial est partagé en deux parties distinctes, l'une dédiée à l'apprentissage qu'on appellera *jeu apprentissage* et l'autre à l'évaluation qu'on appellera *jeu test* (voir section 1.3.2).

Dans cette section, nous présentons à la section 3.7.1 les jeux de données sur lesquels nous avons expérimenté notre approche, puis à la section 3.7.2 les algorithmes de recommandation avec lesquels nous avons comparé les performances de notre approche, en fin, à la section 3.7.3 les critères d'évaluation que nous avons utilisés pour mesurer les performances d'un algorithme de recommandation.

3.7.1 Jeux de données

Pour l'évaluation de notre approche, nous avons utilisé le jeu de données de MovieLens mis à la disposition de la communauté des chercheurs par le groupe de recherche GroupLens [27]. MovieLens compte parmi les jeux de données les plus utilisés par la communauté des chercheurs dans le domaine des systèmes de recommandation. Nous avons utilisé deux jeux parmi les 3 jeux de données proposés par MovieLens (voir section 1.3.1), MovieLens100K qui contient en plus des votes des utilisateurs la description des genres de chaque film, c'est pour cette raison que nous l'avons utilisé dans nos premières publications [Ben Ticha *et al.*, 2011b; Ben Ticha *et al.*, 2011a] pour expérimenter la construction du profil sémantique des utilisateurs sur l'attribut *genre*. Le jeu de données MovieLens1M a été utilisé dans le reste de nos publications [Ben Ticha *et al.*, 2012; Ben Ticha *et al.*, 2013; Ben Ticha *et al.*, 2014; Ben Ticha *et al.*, 2015] pour expérimenter d'autres attributs sur les films. Ces jeux ont été utilisés pour représenter les données issues de l'analyse des usages c'est-à-dire les données permettant la construction de la matrice des votes des utilisateurs Mv.

Pour les données sur le contenu des items, plusieurs sources ont été utilisées en fonction des attributs des films. Les attributs décrivant les films que nous avons expérimentés sont *genre* et *origine* pour les attributs indépendants, *réalisateur*, *acteurs* et *Tag* pour les attributs dépendants, *Mot Clé* pour décrire les synopsis des films résultat d'une indexation. Seul l'attribut *genre* du film est disponible dans les jeux de MovieLens. Le jeu de données HetRec2011 [28] [HetRec2011, 2011] qui relie les films du jeu de données MovieLens10M avec leur page web correspondante dans

27. http ://www.grouplens.org/
28. http ://grouplens.org/datasets/hetrec-2011/

Internet Movie Database (IMDb) [IMDB, 2014] et Rotten Tomatoes movie review systems [29] a été utilisé pour extraire les informations sur les attributs *origine*, *réalisateur*, *acteur* et *Tag* des films. Pour l'attribut *mot clé*, nous l' avons extrait à partir de la base des films IMDb [IMDB, 2014].

3.7.1.1 Jeu MovieLens100k

Le jeu MovieLens100k est composé de 100000 votes entiers définis sur une échelle de 1 à 5. Les votes sont explicites, et horodatés, attribués par 943 utilisateurs à 1682 films, tout utilisateur a évalué au moins 20 films. Le taux de valeurs manquantes est de l'ordre de 95%. Les films sont décrits par leur genre, 18 genres sont définis.

Le partage entre apprentissage et test est fait selon la proportion 80/20, c'est-à-dire que 80% des votes de chaque utilisateur sont utilisés pour l'apprentissage et 20% pour le test. Le découpage apprentissage test est réalisé de deux manières différentes : aléatoire ou en utilisant l'horodatage. Pour le partage aléatoire, le jeu offre 5 découpages différents ($u1$ à $u5$) pour une validation croisée (voir la section 1.3.2). Dans les expérimentations utilisant le jeu MovieLen100k nous avons utilisé la validation croisée, soit les 5 découpages proposés et nous avons calculé la moyenne des résultats obtenus par chaque découpage.

3.7.1.2 Jeu MovieLens1M

Le jeu MovieLens1M est composé de 992071 votes entiers définis sur une échelle de 1 à 5. Les votes sont également horodatés attribués par 6040 utilisateurs à 3952 films, chaque utilisateur a évalué au moins 20 films. Contrairement au jeu MovieLens100k, le nombre d'utilisateurs est largement supérieur au nombre d'items dans le jeu MovieLens1M. Comme nous l'avons déjà mentionné, les données sémantiques sont extraites à partir du jeu de données HetRec2011 [HetRec2011, 2011] et de la base de données des films IMDb [IMDB, 2014].

Une opération de filtrage a été réalisée sur les films pour ne conserver que ceux ayant des valeurs pour les attributs *Genre*, *Origine*, *Réalisateur* et *Acteur*. Parmi ces films, seuls les films ayant 20 votes ont été sélectionnés. Nous avons ainsi obtenu un jeu de données constitué de 3552 films et 6020 utilisateurs.

La liste des attributs que nous avons expérimentés est constituée de :

Attribut indépendant :

Genre **:** décrit les genres des films, chaque film peut avoir plusieurs genres. Les films du jeu Movielens1M sont décrits par 19 genres.

Origine **:** représente le pays origine du film. Chaque film a une seule origine. Après l'opération de filtrage 44 origines ont été retenues.

Attribut dépendant :

Réalisateur **:** contient 1825 réalisateurs de films différents.

29. http ://www.rottentomatoes.com

***Acteur* :** décrit les acteurs des films, chaque acteur est défini par un rang indiquant l'importance de son rôle dans le film. Seuls les acteurs de rang 1 et 2 ont été retenus. Après l'opération de filtrage 4237 acteurs ont été retenus.

***Tag* :** représente les annotations des utilisateurs. Chaque film est décrit par le nombre d'occurrences de chaque annotation. Il est à noter, que seuls 2953 films ayant 20 votes parmi les 3952 sont annotés. Le jeu de données est alors constitué de 4184 annotations attribuées à 2953 films évalués par 5951 utilisateurs.

Item non structuré : les films sont décrits par un ensemble de termes représentant des mots clés. Les mots clés sont le résultat d'une indexation appliquée sur les synopsis des films. Nous avons récupéré ces mots clés à partir de la base de données des films IMDb. Le nombre de mots clés décrivant les films du jeu MovieLens1M s'élève à 35050.

***mot clé* :** en raison du problème de passage à l'échelle, nous avons appliqué un filtre en ne conservant que les mots clés apparaissant dans au moins 6 films. Le nombre de mots clés est alors passé à 6840 décrivant 3544 films évalués par 6020 utilisateurs. Dans tout ce qui suit, les mots clés seront considérés comme des attributs dépendants vu que leur nombre est étroitement lié au nombre d'items.

Nous avons réalisé un découpage apprentissage test selon la proportion 80/20 en se basant sur l'horodatage (timestamp exprimé en seconde). Pour ce faire, nous avons trié les votes de chaque utilisateur par ordre croissant du timestamp. Les 80% premiers votes de chaque utilisateur ont été ajoutés à la base d'apprentissage, les 20% des votes restant à la base de test. Ainsi, pour chaque utilisateur, les votes de la base de test sont ultérieurs à ceux de la base d'apprentissage.

3.7.2 Comparaison des performances de notre système USCF

Nous avons comparé notre système de recommandation hybride (USCF) à une approche basée sur l'usage, à une approche basée sur le contenu et à une approche hybride.

Approche basée sur l'usage

Nous avons évalué notre approche par rapport aux deux standards des systèmes de recommandation collaboratifs, à savoir le filtrage collaboratif basé sur les utilisateurs [Resnick *et al.*, 1994] et le filtrage collaboratif basé sur les items [Sarwar *et al.*, 2001].

Le filtrage collaboratif basé sur les utilisateurs (User Based Collaboratif Filtering UBCF) : calcule la prédiction des votes en se basant sur la similarité entre les utilisateurs calculée à partir de leur profil usage. Le principe de cet algorithme a été introduit dans la section 2.1.1.1. Nous avons appliqué la formule (2.5) pour calculer la prédiction des votes. Le coefficient de Pearson (voir formule (2.11)) a été utilisé pour calculer la similarité entre les utilisateurs. Dans toute la suite de ce manuscrit, cet algorithme sera désigné par UBCF.

Le filtrage collaboratif basé sur les items (Item Based Collaboratif Filtering IBCF) : calcule la prédiction des votes en se basant sur la similarité entre les items calculée à partir de leur profil usage. Le principe de cet algorithme est décrit dans la section 2.1.1.3. La formule (2.7) a été utilisée pour calculer la prédiction des votes. La similarité entre les items a été calculée en utilisant le cosinus ajusté (voir formule 2.13 page 40). Dans toute la suite de ce manuscrit, cet algorithme sera désigné par IBCF.

Approche basée sur le contenu

Nous avons comparé notre algorithme à une approche basée sur le contenu appliquant l'algorithme des plus proches voisins (voir section 2.2.2 page 48) . Le principe de l'algorithme traduit l'idée de base du filtrage basé sur le contenu. Pour recommander des items pertinents à un utilisateur u_a, on commence par déterminer la liste de ses items préférés (ayant un vote supérieur ou égal à un seuil donné), pour le jeu MovieLens, la valeur du vote seuil est égale à 4. Pour chaque item préféré de u_a, on détermine ses k plus proches voisins parmi les items qui lui sont inconnus (non observés). Parmi tous les plus proches voisins on recommande les Top-N de la liste. Dans toute la suite de ce manuscrit, cet algorithme sera désigné par CB pour (Content Based).

Approche Hybride

Nous avons défini un algorithme se basant sur le même principe que notre approche sauf que le modèle sémantique des utilisateurs est construit sur la base de la moyenne des votes de l'utilisateur. Ainsi, le profil sémantique d'un utilisateur est défini par la formule (3.10) dans laquelle la fonction d'agrégation n'est autre que la moyenne des votes de l'utilisateur. Dans toute la suite de ce manuscrit, on utilisera le terme *Moyenne* pour désigner cet algorithme.

3.7.3 Critères d'évaluation

Pour évaluer les performances d'un algorithme de recommandation nous avons mesuré deux aspects importants. Le premier concerne la pertinence des recommandations pour l'utilisateur à travers la mesure de la précision des recommandations (voir section 1.3.3). Le second, la qualité des recommandations à travers la diversité des recommandations fournies (voir section 1.3.4).

Précision des recommandations

La mesure de la précision d'un système de recommandation reste le critère dominant lors de la comparaison des performances de plusieurs systèmes de recommandation [Herlocker *et al.*, 2004; Shani and Gunawardana, 2011]. Nous rappelons que la précision mesure la capacité d'un système de recommandation à prédire des recommandations pertinentes pour ses utilisateurs. Il peut s'agir de mesurer le potentiel du système à prédire la valeur d'un vote, ou à prédire une liste de recommandations pertinentes.

***Précision de la prédiction des votes* :** nous avons mesuré la précision en utilisant la Racine de l'Erreur Quadratique Moyenne (RMSE) (voir la formule 1.3). Cette mesure est devenue la mesure de référence surtout après son utilisation dans le challenge de Netflix [Bennett and Lanning, 2007] pour la recommandation de films. Elle est utilisée pour les algorithmes de recommandation qui calculent la prédiction des votes. C'est le cas de notre approche USCF, et des deux algorithmes de filtrage collaboratif UBCF et IBCF. Elle mesure la racine carrée de la moyenne des écarts entre un vote prédit et son correspondant dans le jeu test. Plus la RMSE est petite, meilleure est la précision des prédictions.

***Précision des recommandations d'une Top-N liste* :** Plusieurs systèmes de recommandation ne calculent pas la prédiction de la valeur d'un vote mais recommandent plutôt à l'utilisateur courant une liste contenant N items susceptibles de l'intéresser, on parle alors de recommandation Top-N, c'est le cas de l'algorithme de filtrage basé sur le contenu CB. La précision d'une

liste (voir formule (1.5)) mesure la proportion des recommandations réellement pertinentes pour l'utilisateur courant u_a parmi les items recommandés dans la Top-N liste. Pour comparer les performances de notre système par rapport à l'algorithme CB, nous avons mesuré la précision des recommandations d'une Top-N liste. La liste Top-N de USCF est constituée des N items parmi l'ensemble des items pertinents à recommander ayant les plus grandes valeurs de votes prédits. Un item est pertinent si sa valeur de vote prédit est supérieure ou égale à un seuil de vote donné. Pour la base MovieLens le seuil est égal à 4. Plus la précision est élevée, meilleure est la pertinence des recommandations.

Diversité des recommandations

Elle mesure la capacité du système à offrir une liste de recommandations diversifiées (voir section 1.3.4). La diversité est le point fort des algorithmes de filtrage collaboratif basés sur la mémoire et plus particulièrement ceux basés sur les utilisateurs (UBCF). Elle est cependant le point faible des algorithmes basés sur le contenu.

La diversité est appliquée sur une Top-N liste de recommandations en se basant sur la similarité entre les items la composant, les items étant comparés selon leur contenu. Pour évaluer la diversité, nous avons mesuré la Dissimilarité Intra Liste (DIL) (voir la formule 1.13) qui somme les dissimilarités entre les paires d'items de la Top-N liste par utilisateur et calcule par la suite la moyenne sur tous les utilisateurs. Les items de la Top-N liste sont représentés par les trois attributs *Genre*, *Origine* et *Réalisateur* et le Cosinus est utilisé pour mesurer la similarité entre deux items. Plus la DIL est élevée, meilleure est la diversité des recommandations.

3.8 Conclusion

Dans ce chapitre, nous avons présenté l'architecture générale de notre approche User Semantic Collaboratif Filtering (USCF). L'originalité de USCF se situe au niveau de la construction du modèle sémantique des utilisateurs (MSU). Pour ce faire, nous avons défini la notion d'attribut pertinent : attribut influant dans la discrimination des items par les utilisateurs. Nous avons également défini deux classes d'attributs : les attributs dépendants et les attributs indépendants. La construction du modèle sémantique des utilisateurs (MSU) est alors un processus modulable par attribut pertinent. Ainsi, un modèle sémantique des utilisateurs par attribut (MSUA) est construit pour chaque attribut pertinent. Une approche spécifique par classe d'attribut est utilisée pour la construction du MSUA. Pour les attributs indépendants nous proposons des approches issues de l'apprentissage automatique, pour les attributs dépendants nous proposons des approches issues de la recherche et filtrage d'informations [Bezdek, 1981]. Les deux chapitres suivants sont entièrement consacrés à la présentation des approches utilisées pour les attributs indépendants (chapitre 4) et pour les attributs dépendants 5. Les différentes approches ont été expérimentées sur des données réelles issues des jeux de données MovieLens100K et MovieLens1M.

Nous serons amenés dans les chapitres suivants à utiliser les notions que nous avons définies dans ce chapitre. Nous utiliserons parfois leurs sigles pour y faire référence. Pour simplifier la lecture, tous nos sigles obéissent à la même logique, à savoir : M pour désigner Modèle ou Matrice, S pour le terme Sémantique, I pour Item, U pour utilisateur ou Usage et P pour profil. Ainsi, MSU désigne le Modèle Sémantique des Utilisateurs, PUU désigne le Profil Usage des Utilisateurs, PSA_i désigne Profil Sémantique par Attribut de l'item i et PSI le Profil Sémantique des Items.

Nous rappelons par ailleurs, que la signification de tout symbole ou sigle utilisé est décrite dans *La liste des notations* présentée au début de ce manuscrit.

Chapitre 4

Clustering pour l'apprentissage du modèle sémantique des utilisateurs

Dans le chapitre précédent, nous avons présenté l'architecture générale de notre système de recommandation hybride User Semantic Collaboratif Filtering (USCF). USCF est constitué d'un composant chargé de la personnalisation et d'un autre chargé de la recommandation. Au niveau de la personnalisation, nous avons défini un nouveau profil utilisateur le Modèle Sémantique des Utilisateurs (MSU) (voir section 3.4). Le MSU est construit à partir des différents MSUA associés à chaque attribut représentant l'ensemble des items à recommander. On rappelle que MSUA signifie le modèle sémantique des utilisateurs par attribut, il représente le profil sémantique des utilisateurs pour un seul attribut (voir section 3.4.3).

Dans ce chapitre, nous présentons nos solutions pour la construction du Modèle sémantique des utilisateurs par attribut (MSUA) pour les attributs indépendants. Nous rappelons, qu'un attribut indépendant est un attribut dont le nombre de descripteurs (valeurs possibles) est stable et ne varie pas en fonction du nombre d'items. C'est-à-dire que le nombre de valeurs possibles pour un tel attribut est généralement très inférieur au nombre d'items (voir section 3.4.2). Nous rappelons également que le MSU est utilisé pour calculer la similarité entre les utilisateurs. Similarité qui sera par la suite utilisée pour calculer la prédiction des votes de l'utilisateur courant et faire de la recommandation. Le calcul des prédictions de votes pour l'utilisateur courant est effectué en appliquant l'algorithme de filtrage collaboratif des N plus proches voisins (voir section 3.5). Dans toutes les expérimentations que nous présenterons dans ce chapitre, le MSU ne sera composé que d'un seul modèle sémantique des utilisateurs par attribut (MSUA), celui correspondant à l'attribut traité.

Nous commencerons par présenter à la section 4.1 le principe général de notre approche qui repose sur l'application d'un clustering sur l'ensemble des items représentés par leur profil usage. Nous présenterons par la suite, à la section 4.2, un bref aperçu de quelques méthodes de clustering non hiérarchiques, expliquant la motivation de notre choix pour la méthode K-Means et sa variante en clustering flou, la méthode Fuzzy C-Means. La section 4.3 sera consacrée à la description de notre solution, qui a fait l'objet de deux publications ([Ben Ticha *et al.*, 2011b; Ben Ticha *et al.*, 2011a], pour la construction du MSUA en appliquant un clustering flou, suivie d'une étude expérimentale pour l'évaluation de la solution sur l'attribut *Genre* du jeu de données MovieLens. La section 4.4 détaillera la solution, qui a été publiée dans [Ben Ticha *et al.*, 2012], pour la construction du MSUA en appliquant un K-Means, elle sera également suivie d'une

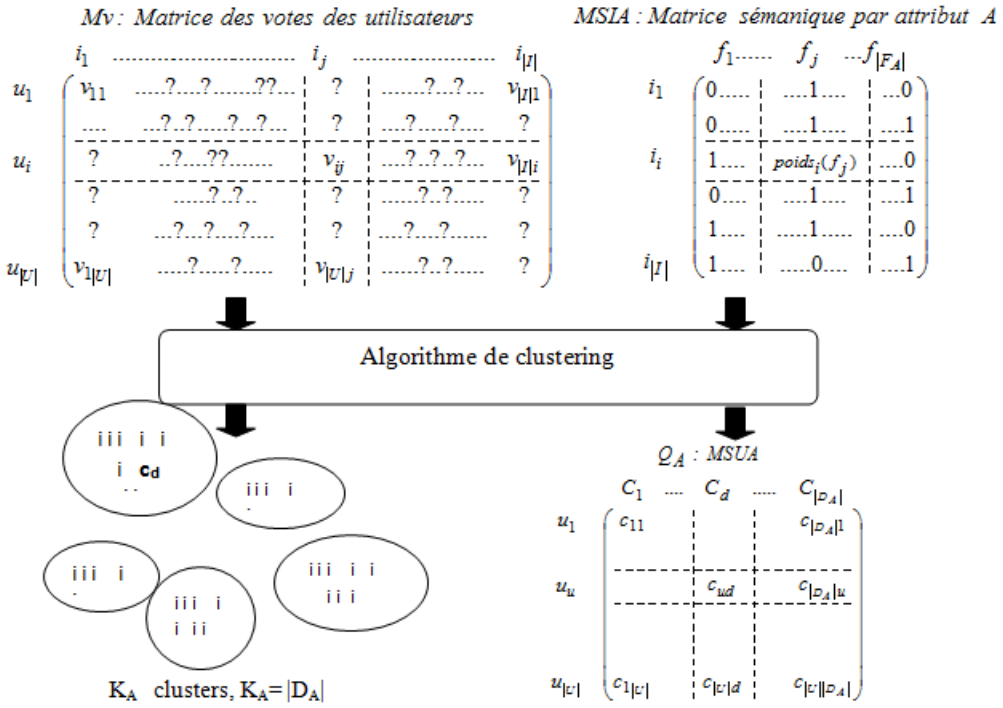


FIGURE 4.1 – Apprentissage du MSUA par clustering

étude expérimentale sur l'attribut *Origine* des films. Enfin, une conclusion clôturera le chapitre en synthétisant tout ce qui a été présenté et en commentant les résultats obtenus.

4.1 Principe général

Soit A un attribut, F_A la liste de ses descripteurs, f_j le descripteur d'indice j dans F_A. Comme nous l'avons vu dans la section 3.3, un item i de I est modélisé par son profil usage PUI_i (voir formule (3.3)), et par son profil sémantique par attribut PSA_{i_A} (voir formule (3.8)). Le profil usage est défini dans l'espace des utilisateurs U et le profil sémantique par attribut est défini dans l'espace des descripteurs F_A de l'attribut A. Par ailleurs, chaque ligne d'indice i de la matrice sémantique des items par attribut $MSIA_A$ (voir formule 3.9) correspond à la transposé du vecteur profil sémantique par attribut PSA_{i_A} de l'item i.

Par ailleurs, et afin de tenir compte de la différence de jugement dans les votes des utilisateurs (tous les utilisateurs ne notent pas les items de la même façon (voir section 2.1.1.5)), tout item i de I sera représenté par son profil usage ajusté $PUIa_i$ (voir formule 3.5) au lieu de PUI_i. Dans toute la suite de ce chapitre, le profil usage d'un item correspondra à son profil usage ajusté.

Nous rappelons par ailleurs, que le modèle sémantique des utilisateurs par attribut $MSUA$ est représenté par la matrice $Q_{A(|U|,|D_A|)}$ (voir formule (3.12)). D_A étant la liste des descripteurs éventuellement latents de l'attribut A décrits dans la dimension des utilisateurs. Chaque descripteur de D_A est défini par un vecteur colonne de la matrice Q_A. On notera par $K_A = |D_A|$ la dimension du MSUA.

Étant donné que le nombre de descripteurs d'un attribut indépendant est stable et ne varie pas en fonction du nombre d'items, l'idée est de partitionner l'ensemble des items représentés par leurs profils usages en clusters. Chaque cluster C_d étant étiqueté par un descripteur f_d de F_A. Chaque cluster C_d est initialement constitué des items décrits par le descripteur f_d qui lui est associé. Après l'exécution d'un algorithme de clustering, on obtient une partition C_A contenant K_A clusters, chaque cluster d'indice $d \in [1, K_A]$ est représenté par son centre défini par le vecteur $centre_d = (centre_d(u))_{(u=1..|U|)}$. Les K_A centres représentent en fait K_A variables latentes modélisant l'attribut A et définis dans l'espace U des utilisateurs. Par conséquent, l'attribut A est modélisé par K_A descripteurs latents, chaque descripteur étant modélisé par le vecteur centre $centre_d$ défini dans l'espace des utilisateurs U. La matrice Q_A modélisant MSUA est construite alors à partir des K_A vecteurs centres comme le définit l'équation 4.1. La liste D_A des descripteurs latents de A est alors constituée des vecteurs centres des K_A clusters.

$$Q_A = ((q(u,d)))_{(u=1..|U|,d=1..|D_A|)} = (centre_d(u))_{(u=1..|U|,d=1..K_A)} \tag{4.1}$$

La figure 4.1 illustre le principe de construction du modèle sémantique des utilisateurs par attributs (MSUA) pour l'attribut indépendant A. Nous avons représenté les données manquantes de la matrice des votes M_v par ' ?'. Le nombre de descripteurs latents $|D_A|$ modélisant le MSUA est égal au nombre de clusters K_A.

La question qui se pose alors est quelle(s) méthode(s) de clustering utiliser ? la section 4.2 étudie quelques méthodes de clustering et présente celles que nous avons choisies.

4.2 Quelle méthode de clustering utiliser ?

4.2.1 Aperçu sur les principales méthodes de clustering

Un algorithme de clustering cherche à faire émerger la structure d'un ensemble de données en déterminant des groupes d'éléments proches selon des relations de distances pré-établies [Jain *et al.*, 1999]. Le but est de partitionner un ensemble de données $D = \{D_1, ..., D_n\}$ en K sous-ensembles (ou clusters) $C = \{C_1, ..., C_k\}$ tels que :

$$\begin{cases} \forall l \in \{1, ..., K\}, C_l \neq \emptyset \\ \forall l, m \in \{1, ..., K\}^2, C_l \cap C_m = \emptyset \\ \bigcup_{l=1}^{K} C_l = D \end{cases} \tag{4.2}$$

De très nombreuses méthodes de clustering, dont un état de l'art est donné dans [Berkhin, 2006], ont été proposées dans la littérature, chacune présentant des caractéristiques propres qui la rendent plus ou moins adaptée à telle ou telle application. Une étude intéressante sur l'utilisation des méthodes de clustering dans le domaine du filtrage de l'information est donnée dans [Lamprier, 2008]. On distingue les *méthodes hiérarchiques* [Johnson, 1967], dont le principe est de créer un arbre dans lequel chaque nœud correspond à un cluster regroupant des clusters de plus bas niveau pour fournir une hiérarchie représentative de la structure du jeu de données concerné, des *méthodes non-hiérarchiques* (ou Flat Clustering) qui ne produisent qu'un seul niveau de clusters. Le principal avantage des méthodes hiérarchiques est qu'elles ne nécessitent pas la détermination du nombre de clusters a priori [Willett, 1988], contrairement aux méthodes non-hiérarchiques qui requièrent bien souvent l'instanciation d'un grand nombre de paramètres (nombre de clusters, taille des clusters, critères d'optimisation, etc...). Toutefois,

ces méthodes présentent elles aussi leurs avantages : les partitions obtenues sont généralement de meilleure qualité que les partitions que l'on peut extraire d'une hiérarchie pour un nombre de groupes donné [Tufféry, 2005]. Par ailleurs, les méthodes non-hiérarchiques sont généralement bien plus rapides que les méthodes hiérarchiques surtout si $K <<< D$ [Rocchio, 1966; Xu and Wunsch, 2005]. Néanmoins, une limite bien connue de ces méthodes est que leur partition finale dépend souvent de la partition initiale [Salton and Wong, 1978].

Nous rappelons que notre objectif est de partitionner un ensemble d'items en sous-ensembles d'items dont chaque cluster de la partition initiale est constitué des items décrits par un descripteur de F_A. Le nombre de clusters sera alors égal au nombre de descripteurs de A. Comme A est un attribut indépendant, alors le nombre de ses descripteurs est très inférieur au nombre d'items à partitionner. Les méthodes de clustering non-hiérarchiques sont plus adaptées à notre problématique, c'est pourquoi nous nous limiterons dans ce qui suit à la présentation des méthodes de clustering non-hiérarchiques.

L'idée centrale de la plupart des méthodes de clustering non-hériachiques consiste à choisir une partition initiale de l'ensemble de données (ici les items) que l'on transforme itérativement en réalisant des réaffectations d'appartenance aux groupes tant que de telles opérations permettent d'optimiser un ou plusieurs critères donnés [Anderberg, 1973]. Les méthodes de clustering non-hiérarchiques ont recours à des heuristiques pour approximer la solution optimale. Une étude récente détaillant les nombreuses méthodes de clustering non-hiérarchiques existantes se trouve dans [Berkhin, 2006].

La méthode Single-Pass : certainement l'algorithme de clustering le plus rapide de tous, l'algorithme Single-Pass [Frakes and Baeza-Yates, 1992] ne nécessite qu'un seul examen de chaque élément pour décider de son affectation dans un cluster donné. Le principe est très simple : on considère les éléments les uns après les autres pour décider s'ils doivent être affectés au cluster courant ou si l'on doit créer une nouveau groupe pour les y insérer. Pour chaque élément, la décision est prise en fonction de sa distance au cluster courant. Cette méthode est loin d'être optimale puisque la partition obtenue est étroitement liée à l'ordre dans lequel sont examinés les éléments [Zamir and Etzioni, 1998].

La méthode des Nuées Dynamiques : la méthode de clustering par Nuées Dynamiques [Diday *et al.*, 1982; Kamber, 2006] est un exemple de méthode dite de "'ré-allocation"'. Contrairement à la méthode Single-Pass où, l'on ne réalise qu'un seul passage, la méthode des Nuées Dynamiques relance le processus d'affectation tant que des transferts d'éléments d'un cluster dans un autre sont observés. La méthode vise à raffiner une partition initiale obtenue par une méthode de clustering externe en réaffectant les éléments à des clusters différents. Un avantage non négligeable de cette méthode par rapport à la méthode Single-Pass réside dans le fait que la partition obtenue est indépendante de l'ordre dans lequel les éléments ont été considérés. De plus, une certaine stabilité de la partition produite a été observée [Rasmussen, 1992]. Enfin, il est à noter que la méthode n'est que peu sensible à la métrique utilisée pour calculer les distances entre objets [Schütze and Silverstein, 1997].

La méthode K-Mean : la plus célèbre des méthodes de clustering, la méthode K-Means [MacQueen, 1967], cherche à produire une partition d'un ensemble d'objets qui minimise la distance des objets avec le centre du cluster qui les contient. L'objectif est de trouver l'ensemble

optimal de centres de clusters.

La méthode commence par choisir aléatoirement k (pour k clusters) points dans l'espace vectoriel. Les points peuvent être produits par une méthode externe. Ces points représentent des prototypes de clusters. Tant que le processus n'a pas atteint une partition stable, la méthode (ré)affecte les objets au cluster de leur plus proche prototype pour former les nouvelles partitions. Les nouveaux centres des clusters constitueront les nouveaux points prototypes de la prochaine itération. Il a été observé à plusieurs reprises que l'algorithme converge rapidement vers une partition satisfaisante dans la plupart des cas [Kaufman and Rousseeuw, 2009]. Du fait de sa simplicité et de ses performances, cet algorithme a été utilisé de manière intensive durant ces trois dernières décennies dans le domaine de la recherche et filtrage d'information [Manning *et al.*, 2008], il a également été employé dans les systèmes de recommandation [Schafer *et al.*, 2007; Lops *et al.*, 2011; Verma *et al.*, 2013] pour traiter entre autres le problème du passage à l'échelle.

Une des principales difficultés que pose l'utilisation de la méthode K-means est la détermination du nombre k de clusters à produire. Néanmoins, de nombreuses approches, telles que la méthode AIC (Akaike information criterion) ou la Gap Statistic [Kamber, 2006] proposent une estimation automatique de ce nombre. De nombreuses variantes de cette méthode existent, notamment la méthode Fuzzy C-Means [Bezdek, 1981] qui permet l'affectation possible d'un même élément à plusieurs clusters, on parle alors de clustering flou (Fuzzy clustering), la méthode QT (Quality Threshold) [Heyer *et al.*, 1999] qui n'impose pas de spécifier un nombre de clusters mais se base sur la taille des clusters ou la méthode K-Medoïd [Kaufman and Rousseeuw, 2009] dont les prototypes associés aux clusters sont des éléments de l'ensemble à partitionner (les Medoïdes) plutôt que des points de l'espace vectoriel (ou centroïdes).

4.2.2 Notre choix

Nous avons opté pour la méthode K-Means comme algorithme de clustering pour construire le MSUA. Les raisons sont en fait assez évidentes :

1. le K-Mean permet de partitionner un ensemble en sous-ensembles représentés par leur vecteur centre (centroïde), dans notre approche on cherche à définir un représentant unique pour chaque cluster, et le centre répond parfaitement à ce besoin.
2. il exige une étape d'initialisation au cours de laquelle est définie le nombre K de clusters et leur centre respectif (les points prototypes).
3. comme toutes les méthodes de clustering non-hiérarchiques, la partition finale dépend de la partition initiale. Or dans notre cas, ceci ne représente pas un inconvénient mais au contraire un avantage puisque l'étape d'initialisation aura pour but d'orienter le clustering.
4. en plus de sa simplicité, il est considéré parmi les algorithmes les plus performants [Xu and Wunsch, 2005; Mingoti and Lima, 2006] en terme de complexité algorithmique, ce qui explique son utilisation intense depuis une trentaines d'années essentiellement dans le domaine de la recherche d'information [Manning *et al.*, 2008].

4.2.2.1 Attribut multivalué et Attribut mono-valué

Soit A un attribut, $F_A = \{f_1, ..., f_l, ..., f_{|F_A|}\}$ la liste de ses descripteurs et F_{A_i} l'ensemble des descripteurs de A décrivant l'item i. Nous rappelons que le profil sémantique par attribut de l'item i pour l'attribut A est défini par le vecteur PSA_{i_A} (voir formule 3.8). L'élément d'indice l de

PSA_{i_A} correspond au poids, $poids_i(f_l)$, du descripteur f_l dans l'item i. Dans toute la suite de ce chapitre, le $poids_i()$ sera égal à la fonction $présence_i()$ telle que nous l'avons définie dans la section (3.8) et plus particulièrement par la formule (3.6). Nous rappelons que $présence_i(f_l), f_l \in F_A$ indique si le descripteur f_l décrit ou pas l'item i, c'est-à-dire, que si $f_l \in F_{A_i}$ alors $présence_i(f_l)$ est égale à 1, 0 dans le cas contraire.

Définition *(Attribut monovalué)* un attribut A est dit monovalué, si à tout item i de I ne peut être associé qu'une et une seule valeur de A. On parle aussi d'attribut à valeur atomique dans d'autres domaines de recherche telles que les bases de données par exemple. Ainsi, chaque item i de I est décrit par un et un seul descripteur de A. Le $PSA_{i_A} = (poids_i(f_l))_{(l=1..|F_A|)}$ pour un attribut monovalué est alors défini par :

$$\begin{cases} \exists \text{ un et un seul } f_q \in F_A \; poids_i(f_q) = 1. \\ \forall \; f_l \in F_A \text{ et } f_l \neq f_q, \; poids_i(f_l) = 0. \end{cases}$$

Plus simplement, lorsqu'un attribut A est monovalué, le vecteur profil sémantique par attribut de l'item i, PSA_{i_A} contient un seul élément dont la valeur est égale à 1, tous les autres ont la valeur zéro.

Définition *(Attribut multivalué)* un attribut A est dit multivalué, si à un item i de I peut être associé une ou plusieurs valeurs de A. Ainsi, un item i peut être décrit par plusieurs descripteurs de A. Le $PSA_{i_A} = (poids_i(f_l))_{(l=1..|F_A|)}$ pour un attribut multivalué est alors défini par :

$$poids_i(f_l) = \begin{cases} 1 \;\; si \;\; f_l \in F_{A_i}. \\ 0 \;\; sinon. \end{cases}$$

Ainsi, le vecteur profil sémantique par attribut d'un item i pour un attribut multivalué A, PSA_{i_A}, contient plusieurs éléments dont la valeur est égale à 1.

A titre d'exemple, le genre d'un film, la catégorie d'un livre sont des exemples d'attributs multivalués. L'origine d'un film, la cuisine d'un restaurant sont des exemples d'attribut monovalués.

Lorsqu'un attribut A est multivalué, un même item peut être décrit par plusieurs descripteurs de A. Or comme nous l'avons déjà mentionné, au démarrage de l'algorithme de clustering, pour la construction du MSUA, chaque cluster correspond à un descripteur de A. Ainsi, si l'attribut est multivalué, et si un item est décrit par 3 de ses descripteurs alors il sera affecté aux trois clusters correspondant à ces trois descripteurs. Ce qui veut dire qu'un même item peut appartenir à plusieurs clusters, plus précisément que l'intersection des clusters n'est pas nécessairement vide comme l'exige un algorithme de clustering "'hard"'. L'algorithme K-Means est une méthode de clustering "'hard"' qui produit des clusters (partitions) disjoints, c'est-à-dire que leur intersection est égale à l'ensemble vide, elle n'est donc pas adaptée aux attributs indépendants multivalués. L'idée est d'utiliser une variante de la méthode K-Means produisant des clusters non disjoints. La méthode Fuzzy C-Means [Bezdek, 1981] est une variante du K-Means produisant une partition composée de K clusters non disjoints où chaque élément est affecté à un cluster avec un coefficient (un poids) mesurant son degré d'appartenance. L'ensemble de tous les poids est représentée par une *matrice de partition*, les clusters en ligne et les éléments à partitionner en colonne. Comme la méthode K-Means, la méthode Fuzzy C-Means est simple à implémenter, mais elle nécessite au démarrage la détermination du nombre de clusters à produire et la définition de la matrice de partition initiale. La partition finale obtenue dépend également de l'étape d'initialisation. En plus de sa simplicité, Fuzzy-C-Means est également considéré comme un algorithme de clustering performant en terme de coût de calcul [Mingoti and Lima, 2006; Ghosh and Dubey, 2013].

En résumé

Nous rappelons que dans ce chapitre nous présentons notre approche pour la construction du Modèle Sémantique des Utilisateurs par Attribut (MSUA) pour les attributs indépendants. L'idée est d'utiliser une méthode de clustering non hiérarchique nécessitant une étape d'initialisation qui en plus a une influence sur la partition finale produite. L'étape d'initialisation consiste à déterminer le nombre de clusters et le prototype initial de chaque cluster. Initialement, chaque cluster est étiqueté par un descripteur de l'attribut A, il est composé des items décrits par ce descripteur. Pour les attributs multivalués la méthode de clustering retenue est le Fuzzy C-Means, la section 4.3 décrit en détail l'algorithme Fuzzy-C-Means ainsi que l'étape d'initialisation. Pour les attributs monovalués, la méthode de clustering retenue est le K-Means, la section 4.4 décrit la méthode K-Means pour la construction du MSUA ainsi que la méthode d'initialisation que nous proposons.

4.2.2.2 Quelle mesure de distance appliquer ?

Il s'agit de définir une mesure permettant d'évaluer la proximité entre deux objets et plus particulièrement entre un objet et un cluster représenté par son prototype. Dans les algorithmes de clustering, cette mesure est représentée par une distance. Plusieurs distances ont été définies dans la littérature dont un résumé est donné dans [Xu and Wunsch, 2005]. On se limite ici à présenter les deux mesures les plus utilisées à savoir la distance de Manhattan (appelée aussi 1-distance) et la distance Euclidienne (appelée aussi 2-distance) définies respectivement par les formules 4.3 et 4.4.

$$distance_{Manhattan}(X,Y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|} \tag{4.3}$$

$$distance_{Euclidienne}(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4.4}$$

Ces deux mesures évaluent la distance qui sépare les deux objets X et Y, plus la distance est élevée et moins les objets sont similaires. Elles sont par ailleurs sensibles à la taille des vecteurs représentant les objets, et les éléments à grandes valeurs et variances ont tendance à dominer les autres éléments. Dans les domaines de la recherche documentaire et des systèmes de recommandation, la similarité entre deux objets X et Y (documents ou items) est généralement mesurée en utilisant le cosinus de l'angle formé par les vecteurs représentant les deux objets. Contrairement aux deux distances, le cosinus est invariant par rapport à la taille des vecteurs représentant les objets, ce qui peut être intéressant lorsque on traite des objets présentant des valeurs manquantes. La valeur du cosinus est comprise entre 0 et 1, la dissimilarité est alors obtenue en prenant le complément à 1 de la similarité.

$$dissimilarité(X,Y) = 1 - similarité(X,Y) = 1 - cosinus(X,Y) \tag{4.5}$$

La mesure de dissimilarité basée sur le cosinus n'est pas une distance telle que définie mathématiquement. En effet, le cosinus ne vérifie pas la propriété de l'inégalité triangulaire. Toutefois, elle a été largement utilisée dans les algorithmes de clustering de documents dans le domaine de la recherche et du filtrage d'informations [Salton, 1989]. Les comparaisons effectuées dans [Willett, 1983] et [Rorvig, 1999] entre différentes mesures ont fait apparaître que le Cosinus permet de

réaliser des clusterings de documents d'une qualité légèrement supérieure.

Dans les expérimentations que nous avons menées, nous avons employé les 3 mesures de dissimilarités définies par les trois formules (4.3), (4.4) et (4.5). Les résultats des expérimentations sont présentés aux sections 4.3.3 et 4.4.3.

4.3 Fuzzy C-Mean pour la construction du MSUA pour les attributs multivalués

La méthode Fuzzy-C-Mean (FCM) compte parmi les méthodes largement utilisées pour effectuer un clustering flou. Cette technique de clustering a été introduite par Jim Bezdek en 1981 [Bezdek, 1981]. L'algorithme FCM est une variante de la méthode K-Means, la différence entre les deux se situe au niveau du contenu des clusters produits. Contrairement au K-Means, FCM produit k clusters non disjoints, c'est-à-dire qu'un même élément peut être affecté à des clusters différents avec pour chacun un poids indiquant son degré d'appartenance. Les travaux que nous présentons dans cette section ont fait l'objet des deux publications [Ben Ticha *et al.*, 2011b; Ben Ticha *et al.*, 2011a].

Nous utilisons un clustering flou pour les attributs indépendants multivalués. L'idée est de partitionner l'ensemble des items I en K_A clusters (ou sous-ensembles) $C_A = \{C_1, ..., C_d, ..., C_{K_A}\}$ associés à une matrice de partition $P_A = (c_{d,i})_{(l=1..K_A, i=1...|I|)}$ où $(c_{d,i})$ est un coefficient mesurant le degrés d'appartenance de l'item i au cluster C_d. La partition C_A doit vérifier les critères suivants :

$$\begin{cases} \forall\, d \in \{1, ..., K_A\}, \quad C_d \neq \emptyset \\ \exists\, d, b \in \{1, ..., K_A\}^2, \quad C_d \cap C_b \neq \emptyset \\ \forall\, d, i \in \{1, ..., K_A\} \times I, \quad (c_{d,i}) \in [0, 1] \\ \forall\, i \in I, \quad \sum_{d=1}^{K_A} (c_{d,i}) = 1 \end{cases} \tag{4.6}$$

Chaque cluster C_d est représenté, par son vecteur centre $centre_d$ obtenu en calculant la moyenne des items du cluster pondérés par leurs degrés d'appartenance à C_d. Un item i de I est représenté par son profil usage ajusté $PUIa_i$ (voir équation (3.5) du chapitre 3) défini dans l'espace des utilisateurs. Le centre d'un cluster est alors donné par l'équation 4.7.

$$centre_d = \frac{\sum_{i \in I} c_{d,i}^m \times PUIa_i}{\sum_{i \in I} c_{d,i}^m} \quad m > 1 \tag{4.7}$$

$c_{d,i}$ est le coefficient mesurant le degré d'appartenance d'un item i à un cluster C_d. Il est inversement proportionnel à la distance séparant l'item i, représenté par le vecteur $PUIa_i$, du centre du cluster C_d. Dans l'équation 4.8, le coefficient est normalisé et "'fuzzifié"' par le paramètre m de sorte que la somme des coefficients d'un même item soit égale à 1, vérifiant ainsi les critères définis dans 4.6. m est un paramètre réel > 1.

$$c_{d,i} = \frac{1}{\sum_{l=1}^{K_A} (\frac{distance(centre_d, PUIa_i)}{distance(centre_l, PUIa_i)})^{2/(m-1)}} \tag{4.8}$$

4.3.1 Algorithme Fuzzy C-Means

L'algorithme de clustering Fuzzy C-Means est composé de deux étapes : une étape d'initialisation (ligne 2) et une étape de clustering (ligne 5) comme le montre l'algorithme 4.1.

Étape d'initialisation : au cours de laquelle sont définis K_A points représentant les prototypes des clusters, un point par cluster. Chaque point représente le centre d'un cluster et est défini dans l'espace vectoriel des utilisateurs. Les points peuvent être définis de façon aléatoire ou par une méthode externe. Pour la construction du MSUA, et pour les raisons évoquées ci-dessus, l'étape d'initialisation a été définie par une méthode externe. La méthode d'initialisation (voir section 4.3.2) utilise la matrice sémantique des items par attribut $MSIA_A$ de l'attribut A pour construire la matrice de partition initiale P_A et en déduire les centres initiaux des clusters en appliquant la formule 4.7. A l'issue de l'exécution de la méthode d'initialisation, on obtient K_A vecteurs centres représentant les prototypes des K_A clusters. Chaque cluster C_d est décrit par un vecteur $centre_d$ défini dans l'espace des utilisateurs comme le montre la ligne 3 de l'algorithme 4.1.

Étape de clustering : c'est une étape itérative constituée de deux parties : une étape de réaffectation et une étape de mise à jour comme le montrent les lignes 7 et 16 de l'algorithme 4.1. Au cours de l'étape de (ré)affectation, la méthode Fuzzy C-Means recalcule la matrice de partition P_A en calculant le nouveau coefficient de chaque item i de I dans chacun des K_A clusters en appliquant la formule 4.8 comme l'illustre la ligne 13 de l'algorithme 4.1. L'étape de mise à jour consiste à recalculer les nouveaux centres des clusters en appliquant la formule 4.7 (voir la ligne 20 de l'algorithme 4.1). Étant donné que chaque item i de I est représenté par son profil usage ajusté $PUIa_i$, le vecteur centre $centre_d$ d'un cluster C_d est défini dans l'espace des utilisateurs. L'algorithme converge lorsque entre deux itérations successives, la partition C_A obtenue représentée par les vecteurs centres des K_A clusters reste inchangée.

Algorithme 4.1 : Algorithme Fuzzy-C-Means

Données :

A : attribut indépendant multivalué,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
m : paramètre réel > 1.

Résultat :

$C_A = \{centre_d\}_{d\in[1,K_A]}$, K_A vecteurs centres définis dans l'espace des utilisateurs représentant chacun un cluster.

1 **début**
2 **Étape d'initialisation**
/* produit K_A vecteurs centres représentant les K_A clusters. Un cluster C_d est représenté par son vecteur centre $centre_d$ défini dans l'espace des utilisateurs. */
3 $\{centre_d\}_{d\in[1,K_A]} = initialisationFCM(Mv, MSIA_A)$
4 $C_A = \{centre_d\}_{d\in[1,K_A]}$
5 **Étape de clustering**
6 **répéter**
7 ***Étape de (Ré)affectation : calculer la matrice de partition*** P_A /* (ré)affecter à chaque item i de I un coefficient $c_{d,i}$ mesurant le degré d'appartenance de i à chaque cluster C_d de C_A */
8
9 **pour** *($i = 1$ à $|I|$)* **faire** /* pour chaque item */
10
11 **pour** *($d = 1$ à K_A)* **faire** /* pour chaque cluster */
12
13 Calculer le coefficient $c_{d,i}$ d'appartenance de l'item i au cluster C_d en appliquant la formule 4.8
14 **fin**
15 **fin**
16 ***Étape de mise à jour*** /* recalculer les centres des clusters */
17 $C' = C_A$ /* conserver l'ancienne partition représentée par ses vecteurs centres */
18 **pour** *($d = 1$ à K_A)* **faire** /* pour chaque cluster */
19
/* vecteur centre du cluster C_d est égal à la moyenne des items de I pondérés par leur coefficient d'appartenance */
20 Calculer le vecteur centre $centre_d$ du cluster C_d en appliquant la formule 4.7
21 **fin**
22 **jusqu'à** *($C_A == C'$)* /* pas de changement */
23 **fin**

4.3.2 Méthode d'initialisation de l'algorithme Fuzzy C-Means

L'étape d'initialisation consiste à déterminer la partition initiale $C_A = \{C_1, ..., C_{K_A}\}$ composée de K_A clusters. Chaque cluster est représenté par son vecteur centre, obtenu en calculant la moyenne des votes le composant pondérés par leurs coefficients d'appartenance. Comme nous cherchons à orienter l'algorithme de clustering, nous avons défini une méthode d'initialisation qui définit le nombre de clusters K_A et les vecteurs centres initiaux des clusters à partir des profils sémantiques des items pour l'attribut A. L'idée est d'associer à chaque descripteur f_d de A un cluster C_d composé des items décrits par f_d. Tout descripteur est alors initialement représenté par le vecteur centre du cluster qui lui est associé. Le centre d'un cluster est obtenu en calculant la moyenne des items le composant pondérés par leurs degrés d'appartenance (voir formule 4.7). Les items étant représentés par leurs profils usages ajustés $PUIa$ constitués des votes des utilisateurs, le taux de données manquantes dans le vecteur centre est directement lié au taux de valeurs manquantes dans le profil usage des items qui le définissent. On rappelle que plus un item i possède de votes et moins son profil usage contient de données manquantes.

En raison des données manquantes, il est possible d'avoir des clusters composés d'items ayant très peu de votes. Ainsi, les centres de tels clusters ne pourront pas être représentatifs des descripteurs auxquels ils sont associés. En plus, le taux de valeurs manquantes dans les vecteurs centres peut être également élevé. Pour faire face aux données manquantes, et garantir que les centres des clusters soient assez représentatifs des descripteurs auxquels ils sont associés, nous avons défini un seuil $NbMinVC$ représentant le nombre minimum de votes que doit avoir un item pour être initialement affecté à un cluster. Ainsi, lors de l'étape d'initialisation, seuls les items de I ayant un nombre de votes supérieur ou égal au seuil $NbMinVC$ seront considérés. La valeur du seuil est étroitement liée aux données et est définie de façon empirique.

Formellement, soit $NbVotes(i)$ le nombre de votes de l'item i, seuls les items appartenant à l'ensemble $I_{NbMinVC} \subset I$ défini par la formule 4.9 sont considérés lors de l'étape d'initialisation.

$$I_{NbMinVC} = \{i \in I \,/\, \ NbVotes(i) >= NbMinVC\} \tag{4.9}$$

Détermination du nombre de clusters

Soit A un attribut multivalué et F_A la liste de ses descripteurs décrivant les items de I. En raison des données manquantes, le nombre de descripteurs de A décrivant les items de $I_{NbMinVC}$ peut être inférieur à celui décrivant tous les items de I. On définit par $F_A^{NbMinVC}$ (voir formule 4.10) l'ensemble des descripteurs de A décrivant les items de $I_{NbMinVC}$.

$$F_A^{NbMinVC} = \{f_d \in F_A \,/\, \exists\, i \in I_{NbMinVC} \ vérifiant \ poids_i(f_d) = 1\} \tag{4.10}$$

Ainsi, pour chaque descripteur de $F_A^{NbMinVC}$ est associé un cluster de la partition initiale C_A. Le nombre K_A de clusters est alors égal au nombre de descripteurs $F_A^{NbMinVC}$ soit $K_A = \left|F_A^{NbMinVC}\right|$.

Détermination de la matrice de partition initiale

Chaque cluster C_d est constitué des items de $I_{NbMinVC}$ décrits par le descripteur $f_d \in F_A^{NbMinVC}$ qui lui est associé. Pour l'algorithme Fuzzy C-Means, il faut définir pour chaque item i son coefficient d'appartenance, $c_{d,i}$, au cluster C_d étiqueté par le descripteur f_d de A.

La matrice de partition initiale P_A définit les coefficients d'appartenance de tous les items de $I_{NbMinVC}$ à tous les clusters de la partition initiale C_A. Ainsi $P_A = (c_{d,i})_{(d=1...K_A\, i=1...|I_{NbMinVC}|)}$ et $c_{d,i}$ défini par la formule 4.11. Nous avons déterminé la matrice de partition initiale à partir de la matrice sémantique des items par attribut $MSIA_A$ pour l'attribut A en ne considérant que les items de $I_{NbMinVC}$ et les descripteurs de $F_A^{NbMinVC}$. Ainsi, le coefficient initial d'appartenance d'un item i à un cluster C_d étiqueté par un descripteur f_d est donné par la formule 4.11.

$$c_{d,i} = \begin{cases} \frac{1}{\sum_{l=1}^{K_A} poids_i(f_l)} \; si \; poids_i(f_d) = 1 \\ 0 \; si \; poids_i(f_d) = 0 \end{cases} \tag{4.11}$$

Nous avons fait l'hypothèse que si un item appartient à n clusters alors son coefficient d'appartenance initial est égal à $\frac{1}{n}$ pour chacun des n clusters et 0 pour les $K_A - n$ clusters restants. C'est-à-dire que nous supposons que les items sont initialement affectés aux clusters de façon équiprobable.

Dans ce qui suit, nous allons expliquer la construction de la matrice initiale des partition P_A par un exemple. Soit $I_{NbMinVC} = \{i_1, i_2, i_3, i_4\}$, soit $\{f_1, f_2, f_3\}$ la liste des descripteurs de A décrivant les items de $I_{NbMinVC}$. La $MSIA_A$ relative aux items de $I_{NbMinVC}$ est donnée par le tableau 4.1.

TABLE 4.1 – Exemple de $MSIA_A$

	f_1	f_2	f_3
i_1	0	1	1
i_2	1	1	1
i_3	0	1	0
i_4	1	1	0

La partition initiale $C_A = \{C_1, C_2, C_3\}$ est constituée de $K_A = 3$ clusters, chaque cluster C_d est associé à un descripteur f_d. $C_1 = \{i_2, i_4\}$, $C_2 = \{i_1, i_2, i_3, i_4\}$ et $C_3 = \{i_1, i_2\}$. La matrice de partition initiale P_A est alors donnée par le tableau 4.2

TABLE 4.2 – Matrice de partition initiale P_A

	C_1	C_2	C_3
i_1	0	1/2	1/2
i_2	1/3	1/3	1/3
i_3	0	1	0
i_4	1/2	1/2	0

Détermination des vecteurs centres des clusters

A partir de la matrice de partition initiale P_A, on calcule les vecteurs centres des K_A clusters en appliquant la formule 4.7.

Algorithme

L'algorithme 4.2 décrit la méthode d'initialisation $InitialisationFCM(Mv, MSIA_A)$ de la ligne 3 de l'algorithme 4.1 décrivant la méthode Fuzzy C-Means pour la construction du MSUA.

Algorithme 4.2 : InitialisationFCM

Données :

A : attribut indépendant multivalué,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
$NbMinVC$: seuil du nombre de votes par item

Résultat :

$\{centre_d\}_{d\in[1,K_A]}$, K_A vecteurs centres initiaux représentant les clusters d'une partition $C_A = \{C_1, ..., C_{K_A}\}$.

```
1  début
2      Déterminer l'ensemble I_NbMinVC = {i ∈ I / NbVotes(i) >= NbMinVC}
3      Déterminer l'ensemble F_A^NbMinVC /* voir formule 4.10                    */
4
5      K_A = |F_A^NbMinVC|
       /* Calculer la matrice de partition initiale P_A                         */
6      pour (i = 1 à |I_NbMinVC|) faire                  /* pour chaque item  */
7
8          pour (d = 1 à K_A) faire                     /* pour chaque cluster */
9
10             Calculer le coefficient initial c_{d,i} d'appartenance de l'item i au cluster C_d en
               appliquant la formule 4.11
11         fin
12     fin
       /* Calculer les vecteurs centres initiaux                                */
13     pour (d = 1 à K_A) faire                     /* pour chaque cluster */
14
15         Calculer le vecteur centre centre_d du cluster C_d en appliquant la formule 4.7
16     fin
17 fin
```

4.3.3 Évaluation

Dans cette section, nous présentons les résultats de l'évaluation de notre approche User Semantic Collaboratif Filtering (USCF) utilisant la méthode Fuzzy C-Means pour construire le modèle sémantique des utilisateurs par attribut (MSUA). Dans tout ce qui suit, nous utiliserons le terme USCF-Fuzzy pour désigner l'algorithme de recommandation USCF appliquant la méthode Fuzzy C-Means pour l'apprentissage du MSUA. Les expérimentions ont été menées sur les deux jeux de données MovieLens1M et MovieLens100k (voir section 3.7.1), et l'attribut *genre* du film.

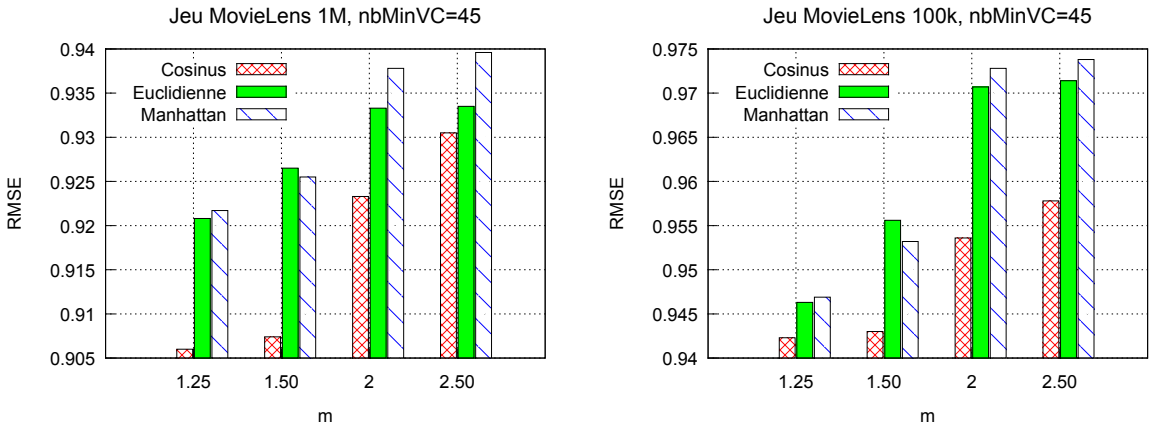
USCF-Fuzzy a été évalué par rapport aux deux algorithmes de filtrage collaboratif basés sur la mémoire (voir section 2.1.1) à savoir la prédiction basée sur les utilisateurs (UBCF sur les graphiques) et la prédiction basée sur les items (IBCF sur les graphiques), un algorithme de recommandation basé sur le contenu (CB sur les graphiques), et l'algorithme hybride (Moyenne

sur les graphiques) qui détermine le profil sémantique des utilisateurs par attribut en calculant la moyenne des votes des utilisateurs pour l'attribut concerné (voir section 3.7.2).

Nous avons évalué ces algorithmes en termes de précision et de diversité des recommandations (voir section 3.7.3). La Racine de l'Erreur Quadratique Moyenne RMSE (voir la formule 1.3), la Précision (voir la formule 1.5) sont utilisées pour mesurer la précision des recommandations. La RMSE est utilisée pour évaluer notre approche par rapport aux algorithmes de recommandation qui calculent la prédiction des votes (UBCF, IBCF et Moyenne), plus la RMSE est petite, meilleure est la précision des prédictions. La Précision est utilisée pour évaluer la précision d'une Top-N liste, nous l'utilisons pour évaluer notre approche par rapport à l'algorithme basé sur le contenu (CB), l'algorithme le plus pertinent est celui ayant les plus grandes valeurs. La Diversité Intra-Liste (voir formule 1.13), DIL sur les graphiques, est utilisée pour mesurer la diversité des recommandations. La DIL est calculée sur la base des trois attributs *genre*, *origine* et *réalisateur*. Plus la DIL est élevée, meilleure est la diversité de l'algorithme de recommandation.

Dans tout ce qui suit et sauf indication, le nombre N des plus proches voisins est égal à 60.

Détermination de la mesure de la distance

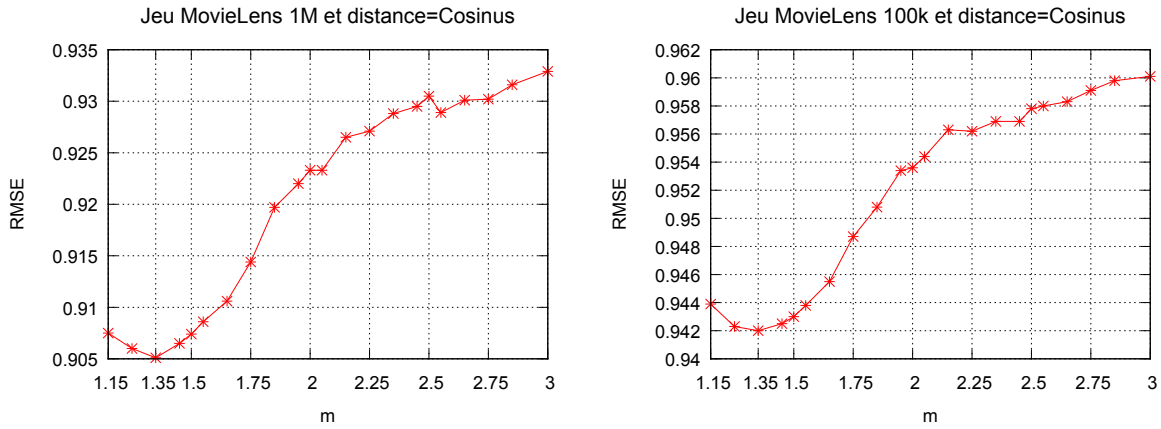


(a) (b)

FIGURE 4.2 – Évaluation des mesures de distance pour USCF-Fuzzy

Comme nous l'avons déjà évoqué, nous avons expérimenté les trois mesures de distance : le *Cosinus*, la distance *Euclidienne* et la distance de *Manhattan*. La figure 4.2 représente la précision de l'algorithme USCF-Fuzzy en fonction du paramètre de "fuzzification" m pour les trois différentes distances, appliquées au jeu MovieLens1M (figure 4.2(a)) et au jeu MovieLens100k (figure 4.2(b)). On y voit que le Cosinus donne les meilleures performances pour les deux jeux de données et pour les différentes valeurs de m. La différence est importante par rapport aux deux autres distances pour m en dessous de 2. Ainsi, pour $m = 1.5$, la $RMSE$ est égale à 0.9074 pour le *Cosinus*, à 0.9265 pour la distance *Euclidienne* et à 0.9255 pour la distance de *Manhattan* pour le jeu MovieLens1M. La différence est approximativement équivalente pour le jeu MovieLens100k pour la même valeur de m, ainsi la $RMSE$ est égale à 0.943 pour le Cosinus, à 0.9556 pour la distance Euclidienne et à 0.9553 pour la distance de *Manhattan*.

Détermination du paramètre m de la méthode Fuzzy C-Means



(a) (b)

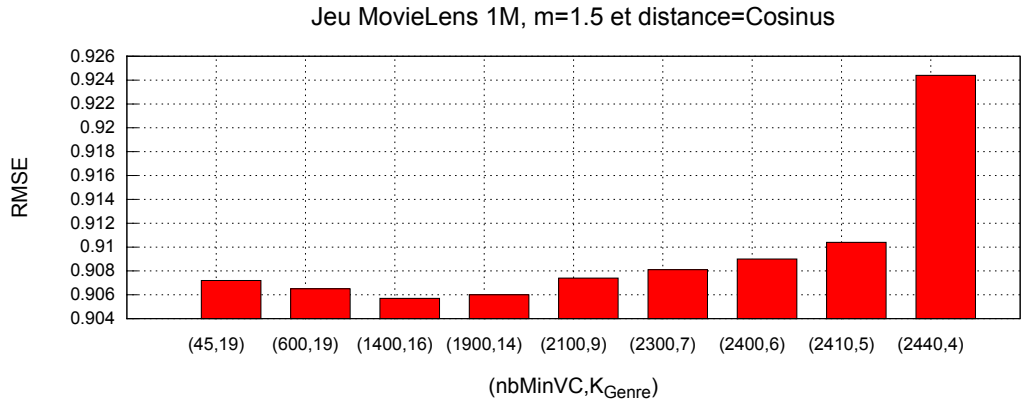
FIGURE 4.3 – Impact du paramètre m sur la pertinence des recommandations USCF-Fuzzy

On rappelle que m est un paramètre propre à l'algorithme Fuzzy-C-Means. Il contrôle la valeur du poids affecté au centre le plus proche d'un item. Ainsi, pour des petites valeurs de m, c'est-à-dire proches de 1, le poids associé au centre du cluster le plus proche d'un item est élevé. Lorsque m atteint sa limite inférieure 1, le poids converge vers 1 ou 0 et l'algorithme se comporte comme un K-Means. Si au contraire la valeur de m est élevée, ceci implique que le poids associé au centre le plus proche est faible, augmentant ainsi le "flou" du clustering. La figure 4.3 représente la précision en fonction du paramètre m pour la distance Cosinus. On rappelle que $NbMinVC$ représente le nombre minimum de votes que doit avoir un item pour participer à la construction du centre initial d'un cluster (lors de l'étape d'initialisation voir section (3.8)). Les courbes sont représentées pour $NbMinVC$ égal à 45 et $Cosinus$ comme mesure de la distance. On remarque que les deux courbes ont la même tendance pour les deux jeux de données, la précision diminue lorsque m augmente. Ainsi, la précision est améliorée pour de petites valeurs de m, l'optimum est atteint pour m égal à 1.35.

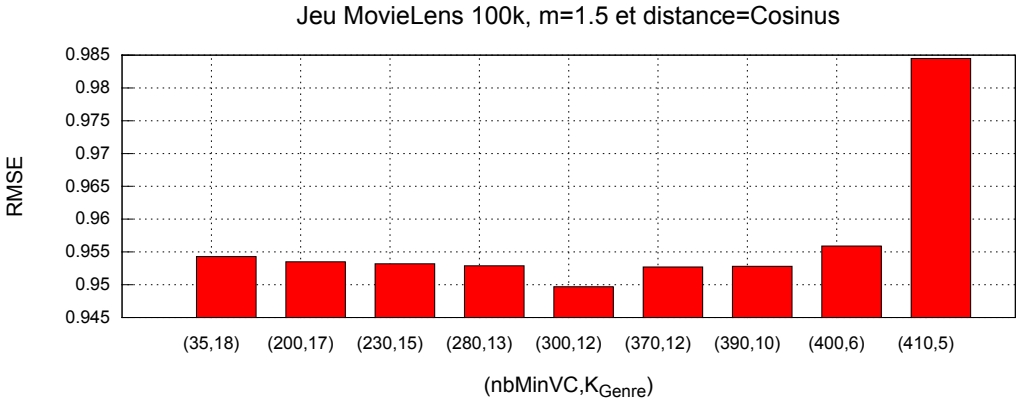
Détermination du nombre de clusters

Le nombre de clusters est déterminé par l'algorithme d'initialisation (voir section 4.3.2). Initialement, chaque cluster représente un descripteur de l'attribut *genre*. Il y a 18 genres pour le jeu MovieLens100k et 19 pour le jeu MovieLens1M. Seuls les descripteurs décrivant les items ayant un nombre de votes supérieur ou égal au seuil $NbMinVC$ sont retenus. La figure 4.4 représente la précision en fonction du seuil $NbMinVC$ et du nombre de clusters correspondants K_{Genre}. Comme nous l'avons détaillé à la section 3.7.1, nous avons appliqué une validation croisée pour le jeu MovieLens100 en utilisant les 5 découpages u_1 à u_5 disponibles. Ainsi, la précision obtenue est la moyenne des résultats des cinq découpages u_1 à u_5. Or, vu que le nombre de clusters peut être différent pour chaque découpage, les résultats représentés dans la figure 4.4(b) sont relatifs au découpage u_1 du jeu MoviLens100k.

Il est à noter que dans les expérimentations que nous menons dans ce chapitre, le modèle sémantique des utilisateurs (MSU) est identique au modèle sémantique des utilisateurs par attri-



(a)



(b)

FIGURE 4.4 – Impact du nombre de clusters sur la pertinence des recommandations USCF-Fuzzy

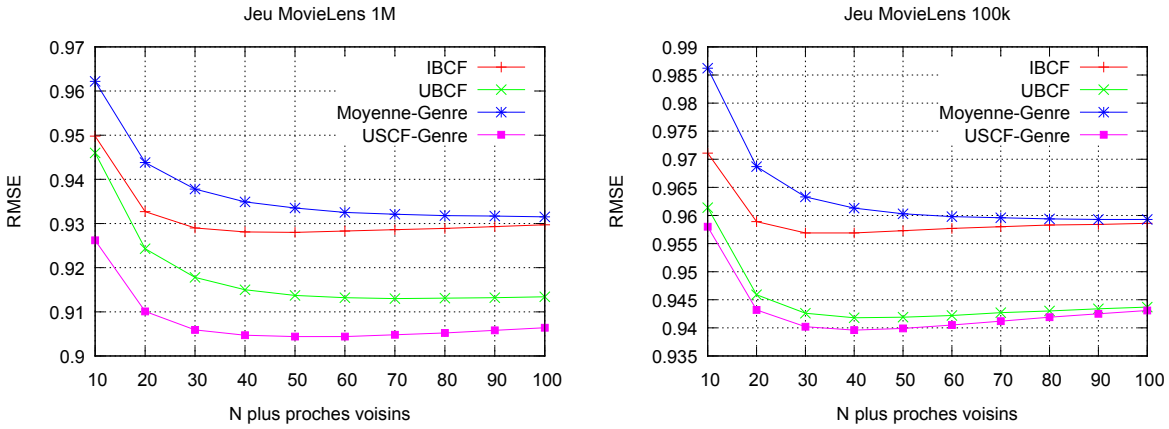
but (MSUA) puisqu'un seul attribut est traité. Ainsi, K_{Genre} représente la dimension du MSU. En observant les deux graphiques, on constate la même tendance pour les deux jeux de données. La valeur du RMSE diminue puis augmente, cependant la variation de sa valeur n'est par importante tant que le nombre de clusters K_{Genre} reste au dessus de 5, c'est-à-dire tant que la dimension du profil sémantique des utilisateurs PSU reste au dessus de 5. La détérioration de la pertinence des recommandations peut s'expliquer par le fait que la similarité entre les utilisateurs est calculée à partir de très peu de valeurs, ici 5. Ainsi, pour le jeu MovieLens1M, RMSE varie entre 0.9057 et 0.9080 pour K_{Genre} inférieur à 6 et le minimum est atteint pour K_{Genre} égal à 16. Pour le jeu MovieLens100k, RMSE varie entre 0.9404 et 0.9435 pour K_{Genre} inférieur à 6 et l'optimum est atteint pour K_{Genre} égal à 12. Cependant, RMSE augmente considérablement lorsque le nombre de clusters K_{Genre} est inférieur à 6, elle est égale à 0.9245 pour MovieLens1M et 0.9695 pour MovieLens100k.

A partir de ces graphiques, on peut conclure que la précision de l'algorithme USCF-Fuzzy est liée à la dimension du modèle sémantique des utilisateurs, MSU, représenté par le nombre de clusters, ici K_{Genre} qui est utilisé pour calculer la similarité entre les utilisateurs. La similarité

entre les utilisateurs est calculée sur la base des K_{Genre} descripteurs latents, si le nombre est trop petit (ici 5), les similarités sont erronées d'où la dégradation de la précision des prédictions.

Il est à noter que le seuil $NbMinVC$ permet de définir le nombre de clusters à retenir. Le fait que la précision ne varie pas beaucoup pour un nombre de clusters au dessus d'un seuil, ici 5, permet de sélectionner la plus petite valeur de K_{Genre} garantissant la précision voulue. La complexité en temps de calcul de l'algorithme Fuzzy C-Means est proportionnelle au nombre de clusters, ainsi plus le nombre de clusters est petit, meilleure est l'efficacité de l'algorithme Fuzzy C-Means. Par ailleurs, le nombre de clusters définit également la dimension D du MSU dans l'algorithme USCF-Fuzzy. Le coût du calcul de la similarité entre deux utilisateurs dans l'algorithme USCF-Fuzzy est de $O(D)$. La réduction de la dimension de MSU améliore également l'efficacité de l'algorithme USCF-Fuzzy et plus particulièrement le composant de l'algorithme calculant la prédiction des votes.

Évaluation de la précision des prédictions de votes



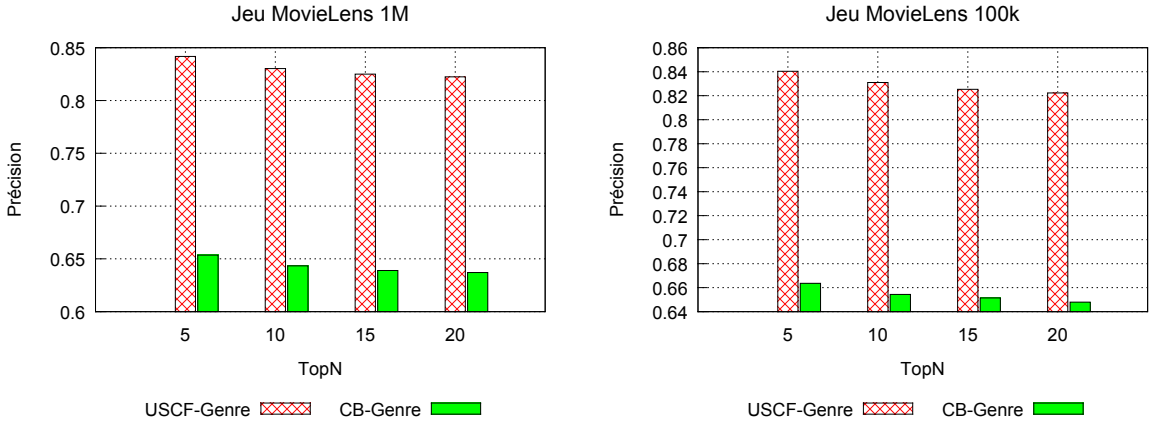
(a) (b)

FIGURE 4.5 – Évaluation de la précision des prédictions de USCF-Fuzzy vs IBCF, UBCF et Moyenne

La figure 4.5 compare la précision des prédictions de notre approche USCF (USCF-Genre sur les graphiques) par rapport aux algorithmes (voir section 3.7.2) purement basés sur l'usage UBCF (User Based Collaborative Filtering, page 73) et IBCF (Item Based Collaborative Filtering, page 73), à l'algorithme hybride Moyenne-Genre appliqué à l'attribut Genre (voir page 74), pour les deux jeux de données MovieLens1M (graphique 4.5(a)) et MovieLens100k (graphique 4.5(b)). Pour les deux jeux de données ainsi que les différents algorithmes, toutes les courbes ont la même allure. La RMSE diminue jusqu'à une valeur donnée des N plus proches voisins puis augmente, elles convergent toutes pour N entre 50 et 60 voisins. Par ailleurs, les différentes approches se comportent de façons identiques pour les deux jeux de données. Ainsi, USCF-Genre et UBCF affichent les meilleures performances en terme de précisions des recommandations alors que Moyenne-Genre affiche les plus mauvais résultats. Toutefois, notre approche reste la plus performante avec une nette différence pour le jeu MovieLens1M dont la valeur de la RMSE est égale à 0.9044 pour 60 voisins contre 0.9132 pour UBCF soit une différence d'un point.

En plus de la réduction de la dimension, notre approche affiche une supériorité au niveau de la précision par rapport à l'algorithme UBCF.

Évaluation de la précision d'une Top-N liste



(a) (b)

FIGURE 4.6 – Évaluation de la précision d'une Top-N liste de USCF-Fuzzy vs CB

Pour comparer notre approche par rapport à un algorithme de recommandation basé sur le contenu (voir section 3.7.2, page 74), nous avons utilisé les mesures permettant d'évaluer une Top-N liste vu que l'algorithme basé sur le contenu ne calcule pas de prédiction de votes. Pour ce faire, nous avons utilisé la Précision (voir formule 1.5) qui estime le taux de recommandations pertinentes pour différentes tailles de la Top-N liste . *CB-Genre* utilisé dans le graphique signifie que l'algorithme basé sur le contenu est exécuté sur l'attribut *Genre*.

La figure 4.6 représente les résultats pour les deux jeux de données MovieLens1M (figure 4.6(a)) et MovieLens100k (figure 4.6(b)). On y voit que notre approche, USCF-Fuzzy sur le graphique, est largement supérieure à une approche purement basée sur le contenu, CB-Genre sur les graphiques.

Évaluation de la diversité

On rappelle que la diversité des recommandations est très souvent un critère d'évaluation de la qualité d'un système de recommandation [Shani and Gunawardana, 2011]. La diversité est le point fort des algorithmes de filtrage collaboratif basés sur la mémoire et plus particulièrement ceux basés sur les utilisateurs (UBCF). Elle est généralement évaluée pour les algorithmes basés sur le contenu. Dans notre cas, nous évaluons la diversité de notre approche par rapport à l'algorithme CB appliqué à l'attribut Genre (CB-Genre sur les graphiques) et l'algorithme de filtrage collaboratif basé sur les utilisateurs (UBCF). On rappelle par ailleurs que la diversité a été calculée sur la base des attributs *genre*, *origine* et *réalisateur* (voir section 3.7.3). Nous avons utilisé la Dissimilarité Intra Liste (DIL) pour mesurer la diversité des recommandations. DIL (voir formule 1.13) somme les dissimilarités entre les paires d'items de la Top-N liste par

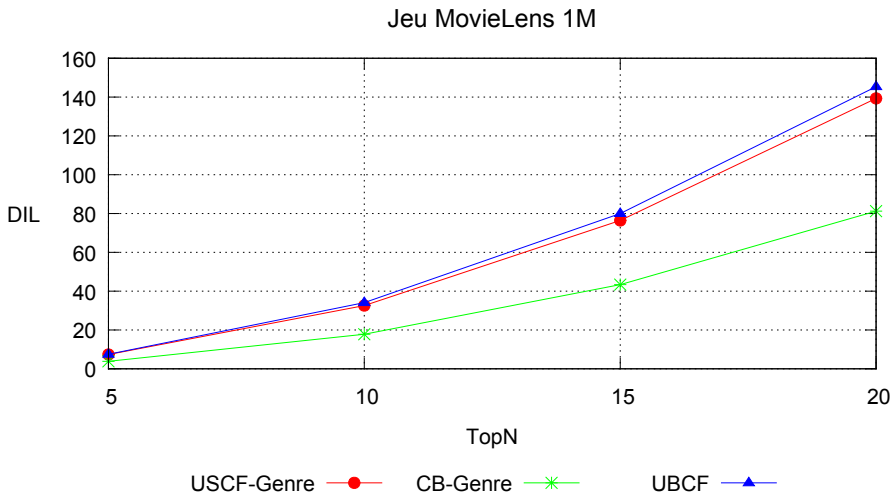


FIGURE 4.7 – Évaluation de la diversité d'une Top-N liste de USCF-Fuzzy par rapport à CB et UBCF

utilisateur et calcule par la suite la moyenne sur tous les utilisateurs.

Ainsi, pour une Top-5 liste ($N = 5$), la valeur maximale que peut avoir la DIL par utilisateur est égale à 10. En effet, pour une Top-5 liste on a 10 paires d'items (i, j) différentes, si toutes les paires d'items (i, j) de la liste sont complètement dissimilaires, c'est-à-dire que leur $Cosinus(i, j)$ est égal à 0, alors la somme des dissimilarités est égale à 10 pour un utilisateur donné.

TABLE 4.3 – Diversité (DIL) d'une Top-5 liste et Top-10 liste

	Top-5	Top-10
UBCF	7.48	34.11
USCF-Genre	7.34	32.60
CB-Genre	3.86	17.79

La figure 4.7 représente la diversité des recommandations pour différentes Top-N listes produites par notre approche USCF-Fuzzy (USCF-Genre sur le graphique), CB appliqué à l'attribut *Origine* (CB-Origine sur le graphique) et UBCF. Comme nous l'avons précédemment dit, la diversité est le point fort de l'algorithme UBCF, c'est pour cette raison que sa courbe est la plus dominante. USCF-Fuzzy affiche des performances en terme de diversité assez proche de celles obtenues par UBCF et loin devant celles obtenues par l'algorithme CB-Genre. Le tableau 4.3 représente les diversités d'une Top-5 et Top-10 liste recommandées par USCF-Fuzzy, CB et UBCF. La diversité d'une Top-5 liste affichée par CB-Genre est de l'ordre de 3.86, très inférieure à celle obtenue par notre approche USCF-Genre qui atteint la valeur de 7.34 très proche des performances obtenues par UBCF (7.48).

4.4 K-Means pour l'apprentissage du MSUA pour les attributs monovalués

Pour les attributs indépendants monovalués, nous avons appliqué la méthode K-Means pour partitionner les items de I et construire de ce fait le modèle sémantique des utilisateurs par attribut (MSUA). Comme nous l'avons déjà mentionné, la partition finale produite par l'algorithme K-Means est très sensible à la partition initiale. La partition initiale étant généralement représentée par K_A points représentant les prototypes initiaux des K_A clusters. Dans cette section nous présentons l'algorithme K-Means tel que nous l'avons utilisé pour l'apprentissage du MSUA (section 4.4.1). La section 4.4.2 présente la méthode d'initialisation que nous avons définie pour orienter l'algorithme de clustering. Les travaux présentés dans cette section ont fait l'objet de la publication [Ben Ticha *et al.*, 2012].

4.4.1 Algorithme K-Means pour l'apprentissage du MSUA

La méthode K-Means accepte en entrée l'attribut indépendant et monovalué A, la matrice des votes des utilisateurs Mv, les profils sémantiques des items par attribut représenté par la matrice $MSIA_A$. Il produit en résultat une partition $C_A = \{C_1, ..., C_d, ..., C_{K_A}\}$ constituée de K_A clusters définis par leur vecteurs centres. L'algorithme est composé de deux étapes : une première étape d'initialisation (ligne 2) et une seconde étape de clustering (ligne 5) comme le montre l'algorithme 4.3.

Étape d'initialisation : au cours de laquelle sont définis K_A points représentés dans un espace vectoriel et représentant les prototypes des clusters, un point par cluster. Les points peuvent être définis de façon aléatoire ou par une méthode externe. Dans notre cas, nous avons défini une méthode d'initialisation externe qui utilise une ontologie du domaine décrit par l'attribut A (voir section 4.4.2) pour déterminer le nombre de clusters et leur vecteur centre. A l'issue de l'exécution de la méthode d'initialisation, on obtient K_A points représentant les prototypes des K_A clusters. Chaque cluster C_d est décrit par un vecteur $centre_d$ défini dans l'espace des utilisateurs comme le montre la ligne 3 de l'algorithme 4.3.

Étape de clustering : c'est une étape itérative constituée de deux parties : une étape de réaffectation et une étape de mise à jour comme le montrent les lignes 7 et 12 de l'algorithme 4.3. Au cours de l'étape de (ré)affectation, la méthode K-Means réaffecte chaque item i au cluster le plus proche. Pour cela, la distance entre l'item i et le vecteur centre $centre_d$, $d \in [1, K_A]$ de chaque cluster est calculée. L'item i est alors affecté au cluster C_d dont la distance séparant son profil usage $PUIa_i$ du vecteur centre $centre_d$ est la distance minimale (voir ligne 11 de l'algorithme 4.3). L'étape de mise à jour consiste à recalculer les nouveaux centres des clusters. Le centre d'un cluster est obtenu en calculant la moyenne des items le composant. Chaque item i est représenté par son profil usage ajusté $PUIa_i$. Ainsi, le centre d'un cluster est défini dans l'espace des utilisateurs. L'algorithme converge lorsqu' entre deux itérations successives, la partition C_A obtenue, représentée par les vecteurs centres des clusters, reste inchangée.

Algorithme 4.3 : Algorithme K-Means

Données :

A : attribut indépendant monovalué,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.

Résultat :

$\{centre_d\}_{d\in[1,K_A]}$, K_A vecteurs centres représentant les clusters de la partition $C_A = \{C_1, ..., C_{K_A}\}$.

1 **début**
2 **Étape d'initialisation**
`/* produit` K_A `points représentant les prototypes des` K_A `clusters. Un cluster` C_d `est représenté par un vecteur` $centre_d$ `défini dans l'espace des utilisateurs. */`
3 $\{centre_d\}_{d\in[1,K_A]} = initialisationKMeans(Mv, MSIA_A)$
4 C_A = ensemble de K_A clusters vides, $centre_d$ étant le représentant du cluster C_d
5 **Étape de clustering**
6 **répéter**
7 ***Étape de Réaffectation*** `/* (ré)affecter chaque item` i `de` I `au cluster le plus proche en se basant sur la distance entre` $PUIa_i$ `et les centres des clusters */`
8
9 $C' = C_A$ `/* conserver l'ancienne partition */`
10 C_A = ensemble de K_A clusters vides
`/* Chaque item` i `est affecté à un et un seul cluster */`
11 $C_j = \{i \in I \,/\, distance(centre_j, PUIa_i) = min\{distance(centre_d, PUIa_i)\}_{d\in[1,K_A]}\}$
12 ***Étape de mise à jour*** `/* (re)calculer les centres des clusters */`
13 **pour** $(d = 1$ **à** $K_A)$ **faire**
`/* vecteur centre du cluster` C_d `est égal à la moyenne des items le composant */`
14 $centre_d = \frac{1}{|C_d|} \sum_{i\in C_d} PUIa_i$
15 **fin**
16 **jusqu'à** $(C_A == C')$ `/* pas de changement */`
17 **fin**

4.4.2 Méthode d'initialisation

Les attributs indépendants sont généralement des attributs pouvant être décrits par une ontologie comme par exemple l'attribut *catégorie* dans un système de recommandation de publications scientifiques qui peut être représenté par une ontologie décrivant les catégories des articles de recherche, l'attribut *origine* d'un film dans un système de recommandation de films pouvant être décrit par une ontologie de pays.

La méthode d'initialisation que nous présentons dans cette section exploite une ontologie du

domaine décrit par l'attribut indépendant A, pour construire la partition initiale C_A, nécessaire au démarrage de l'algorithme K-Means.

Comme pour la méthode Fuzzy C-Means, la méthode K-Means nécessite une étape d'initialisation durant laquelle est définie une partition initiale $C_A = \{C_1, ..., C_{K_A}\}$ constituée de K_A clusters représentés par leurs vecteurs centres obtenus en calculant la moyenne des items les composant. La méthode d'initialisation que nous proposons est semblable à celle définie pour l'initialisation du Fuzzy C-Means. Ainsi, un seuil $NbMinVC$ [30] est utilisé pour déterminer l'ensemble $I_{NbMinVC}$ (voir formule (4.9) page 87) des items qui seront initialement utilisés. Seuls les descripteurs décrivant les items de $I_{NbMinVC}$ seront alors considérés, c'est-à-dire les descripteurs de l'ensemble $F_A^{NbMinVC}$ défini par la formule (4.10). Chaque cluster est étiqueté par un descripteur de $F_A^{NbMinVC}$ et composé des items de $I_{NbMinVC}$ décrits par ce descripteur. Or comme l'attribut A est monovalué, un item n'est décrit que par un seul descripteur de A et de ce fait affecté qu'au seul cluster qui lui est associé. En raison des données manquantes, il est possible d'avoir des descripteurs de A décrivant un très faible nombre d'items de $I_{NbMinVC}$ ce qui conduit à des clusters composés de très peu d'items. Le centre d'un tel cluster n'est alors pas assez représentatif du descripteur auquel il est associé.

Pour remédier à ce manque de représentativité, nous avons défini un autre indicateur : le nombre minimum d'items composant initialement un cluster, $NbMinIC$ défini également de façon empirique. On définit par $NbItems(f_d)$ la fonction qui renvoie le nombre d'items décrit par le descripteur f_d. Ainsi seuls les descripteurs de A décrivant au moins $NbMinIC$ items de $I_{NbMinVC}$ seront considérés. La liste L_A des descripteurs sélectionnés est alors donnée par l'équation (4.12).

$$L_A = \{f_d \in F_A^{NbMinVC} \ / \ NbItems(f_d) \geq NbMinIC\} \tag{4.12}$$

Si le taux de données manquantes est élevé, beaucoup de descripteurs peuvent être ainsi éliminés. Afin de tenir compte des descripteurs de $F_A^{NbMinVC}$ n'appartenant pas à L_A, nous proposons de les intégrer en utilisant une ontologie du domaine décrit par l'attribut A.

Les ontologies ont été utilisées dans les systèmes de recommandation [Aciar *et al.*, 2007; Middleton *et al.*, 2004a] pour modéliser les items. Dans [Schickel-Zuber, 2007], l'auteur a développé un système de recommandation basé sur les connaissances (Case-Based) [Smyth *et al.*, 2005] qui exploite les ontologies décrivant les attributs pour calculer la distance entre les descripteurs d'un même attribut. Dans notre approche, on utilise une ontologie pour regrouper des descripteurs liés sémantiquement par une ontologie et les associer à un cluster.

[Gruber, 1995] définit une ontologie comme suit (traduction extraite à partir du site Wikipédia [31]) :

Définition *(Ontologie)* Une ontologie est un réseau sémantique qui regroupe un ensemble de concepts décrivant complètement un domaine. Ces concepts sont liés les uns aux autres par des relations taxinomiques (hiérarchisation des concepts) d'une part, et sémantiques d'autre part.

Dans notre cas, nous nous intéressons aux ontologies de domaine représentées par leur forme la plus simple. Ainsi, une ontologie est une structure hiérarchique (une taxonomie) ou les concepts

30. Nous rappelons que le seuil $NbMinVC$ représente le nombre minimal de votes que doit avoir un item pour être initialement affecté à un cluster voir section 4.3.2

31. http ://fr.wikipedia.org/wiki/Ontologie_%28informatique%29, juillet 2014

sont liés entre eux de manière hiérarchique des plus généraux aux plus spécifiques. Les concepts y sont souvent désignés par des nœuds et les relations par des arcs. Ainsi, toute relation reliant deux concepts est de type "'*is-a*"' (*est un*) qui définit un lien de généralisation entre un concept *parent* et un concept *enfant*. Le concept parent est une généralisation du concept enfant, et le concept enfant est une spécialisation du concept parent. Un lien de généralisation décrit également un lien d'inclusion.

Définition *(Concept Ascendant)* Soient C_1 et C_2 deux concepts d'une ontologie liés par un lien de généralisation. On dit que C_1 est un concept ascendant de C_2 si C_1 est une généralisation de C_2.

Définition *(Concept Ascendant direct)* Soient C_1 un concept ascendant de C_2, on dit que C_1 est l'ascendant direct de C_2, et C_2 le fils de C_1, si le nombre d'arcs liants C_1 à C_2 est égal à 1.

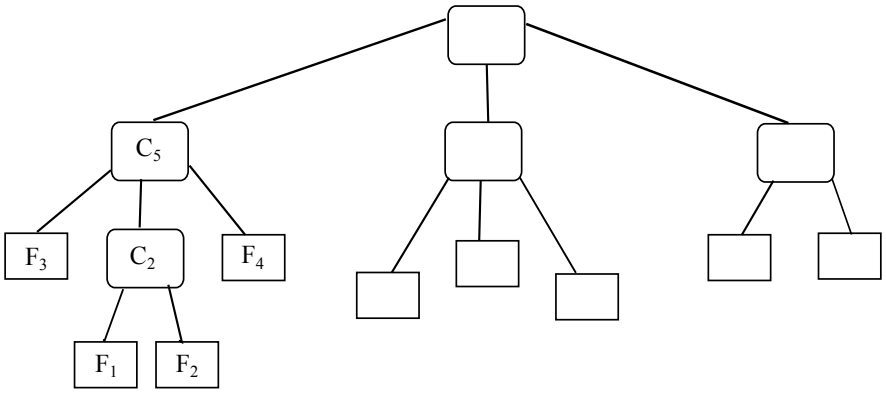


FIGURE 4.8 – Exemple d'une ontologie décrivant un attribut indépendant A

La figure 4.8 représente un exemple d'une ontologie décrivant le domaine représenté par un attribut indépendant A. Les concepts de l'ontologie sont représentés par des carrés et les relations entres les concepts par des arcs. Le concept C_2 est l'ascendant direct des concepts F_1 et F_2, et les concepts F_1 et F_2 sont inclus dans C_2. Le concept C_5 est l'ascendant direct du concept C_2, et le concept C_5 est l'ascendant des concepts C_2, F_1 et F_2. Les carrés aux coins non arrondis représentent les descripteurs F_A de A.

Nous supposons disposer d'une ontologie du domaine décrivant l'attribut A. Les descripteurs de l'attribut A sont des concepts de l'ontologie. L'idée de l'algorithme d'initialisation (algorithme 4.4) est de remplacer chaque descripteur de l'attribut A ne vérifiant pas le critère défini par l'équation 4.12 par son plus proche ascendant dans l'ontologie vérifiant ce critère. La distance entre un concept et son ascendant est égale au nombre d'arcs reliant les deux concepts. Ainsi, dans l'exemple de la figure 4.8, entre les deux concepts C_2 et C_5, tous les deux ascendants de F_1, le concept C_2 est le plus proche de F_1.

Algorithme

Le principe de fonctionnement de l'algorithme est le suivant :

1. Créer un cluster étiqueté par chaque descripteur $F_A^{NbMinVC}$. Ajouter les clusters ayant un nombre d'items inférieur au seuil $NbMinIC$ à la liste initiale de clusters LIC (ligne 10), le reste à la partition initiale C_A (ligne 11).
2. Appliquer l'ontologie pour les descripteurs associés aux clusters de LIC.
   — Récupérer le cluster C en tête de liste de LIC, s'il contient le nombre suffisant d'items (ligne 16) alors l'ajouter à la partition initiale C_A et le supprimer de LIC. Sinon (ligne 20) créer un nouveau cluster étiqueté par son ascendant direct (s'il n'existe pas déjà dans LIC) dans l'ontologie, l'ajouter à la fin de LIC et y ajouter les items de C qui sera supprimé de LIC (lignes 21 à 29).
   — Répéter le traitement précédent jusqu'à ce que la liste LIC soit vide. Comme les nouveaux clusters (étiquetés par des concepts ascendants directs dans l'ontologie) sont ajoutés à la fin de la liste LIC, le traitement d'un ascendant direct dans l'ontologie ne se fait que lorsque tous ses concepts fils sont traités.
3. Une fois la partition initiale C_A déterminée, le nombre de clusters est égal à $|C_A|$. Le vecteur centre de chaque cluster est obtenu en calculant la moyenne des items le composant. Les items sont représentés par leurs profils usages ajustés $PUIa$ (ligne 33).

Algorithme 4.4 : Initialisation K-Means avec ontologie

Données :

A : attribut indépendant monovalué,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
Ont_A : ontologie de domaine de l'attribut A
Critères de sélection
$NbMinVC$ nombre minimum de votes par item
$NbMinIC$ nombre minimum d'items par cluster

Résultat :

$\{centre_d\}_{d\in[1,K_A]}$, K_A vecteurs centres représentant les clusters d'une partition initiale $C_A = \{C_1, ..., C_{K_A}\}$.

début
  Déterminer l'ensemble $I_{NbMinVC} = \{i \in I\ NbVotes(i) >= NbMinVC\}$
  Déterminer l'ensemble $F_A^{NbMinVC}$ `/* voir formule 4.10 */`
  `/* Initialisation: construction de la liste initiale des clusters` LIC `*/`
  $LIC = \emptyset$ `/* liste initiale des clusters */`
  $C_A = \emptyset$
  pour chaque *(*$f_d \in F_A^{NbMinVC}$*)* **faire**
    `/* créer le cluster labellisé par` f_d `*/`
    $C = \{i \in I_{NbMinVC} /\ poids_i(f_d) = 1\}$
    créer le cluster vide $C_d = \emptyset$
    ajouter les items de C dans C_d
    si *(*$f_d \notin L_A$*)* **alors** ajouter le cluster C_d à LIC
    sinon ajouter le cluster C_d à C_A
  fin
  `/* Construction de la partition initiale` C_A `*/`
  tant que *(*$LIC \neq \emptyset$*)* **faire**
    récupérer le cluster C_d se trouvant à la tête de la liste LIC
    si *(*$|C_d| \geq NbMinIC$*)* **alors** `/* contient un nombre suffisant d'items */`
      ajouter le cluster C_d à C_A `/*` C_d `labellisé par` $concept_d$ `de` Ont_A `*/`
      supprimer C_d de LIC
    fin
    sinon `/* accéder à l'ascendant direct de` $concept_d$ `*/`
      si *(ascendant direct de* $concept_d$ *existant)* **alors**
        $concept_j$ =concept ascendant direct de $concept_d$ labellisant le cluster C_d
        si *(pas de cluster associé au concept* $concept_j$ *dans* LIC*)* **alors**
          créer le cluster $C_j = \emptyset$ labellisé par le $concept_j$
          ajouter C_j à la fin de LIC
        fin
        ajouter le contenu du cluster C_d dans C_j
      fin
      supprimer C_d de LIC
    fin
  fin
  `/* Calculer le vecteur centre initial de chaque cluster de` C_A `*/`
  pour chaque *(*$C_d \in C_A$*)* **faire**
    $centre_d = \frac{1}{|C_d|} \sum_{i \in C_d} PUIa_i$
  fin
  $K_A = |C_A|$
fin

4.4.3 Évaluation

Dans cette section, nous présentons les résultats de l'évaluation de notre approche USCF utilisant la méthode K-Means pour construire le modèle sémantique des utilisateurs par attribut (MSUA). Dans tout ce qui suit, nous utiliserons le terme USCF-KMeans pour désigner l'algorithme USCF appliquant la méthode K-Means pour l'apprentissage du MSUA. Les expérimentions ont été menées sur le jeu de données MovieLens1M (voir section 3.7.1), et l'attribut *origine* du film. Nous avons, par ailleurs, utilisé l'ontologie décrivant le pays d'origine des films décrit dans [Bouza, 2010]. Une description détaillée de l'ontologie est donné à l'annexe A.

USCF-KMeans a été évalué par rapport aux deux algorithmes de filtrage collaboratif basés sur la mémoire (voir section 2.1.1) à savoir la prédiction basée sur les utilisateurs (UBCF sur les graphiques) et la prédiction basée sur les items (IBCF sur les graphiques), un algorithme de recommandation basé sur le contenu (CB-Origine sur les graphiques), et l'algorithme hybride (Moyenne-Origine sur les graphiques) qui détermine le profil sémantique des utilisateurs en calculant la moyenne des votes des utilisateurs pour l'attribut origine (voir section 3.7.2).

Détermination de la mesure de la distance

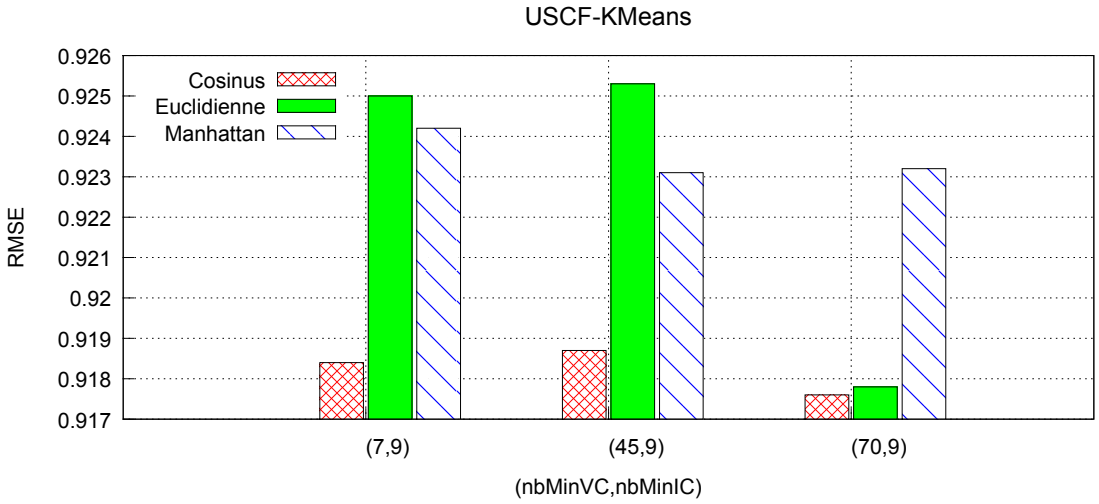


FIGURE 4.9 – Évaluation des mesures de distance pour USCF-Kmeans

Comme pour l'algorithme Fuzzy C Means, nous avons expérimenté les trois mesures de distance : le Cosinus, la distance Euclidienne et la distance de Manhattan. La figure 4.9 représente la RMSE de l'algorithme USCF-KMeans pour les trois distances et en fonction de différentes valeurs de $nbMinVC$. Comme pour l'algorithme USCF-Fuzzy, le Cosinus donne les meilleures performances par rapport aux deux autres distances.

Détermination du nombre de clusters

Le nombre de clusters est déterminé par l'algorithme d'initialisation (voir section 4.4.2). Initialement, chaque cluster représente un descripteur de l'attribut origine. Il y a 43 origines dans le jeu MovieLens100k. Le nombre de clusters est directement lié aux paramètres $NbMinVC$ et

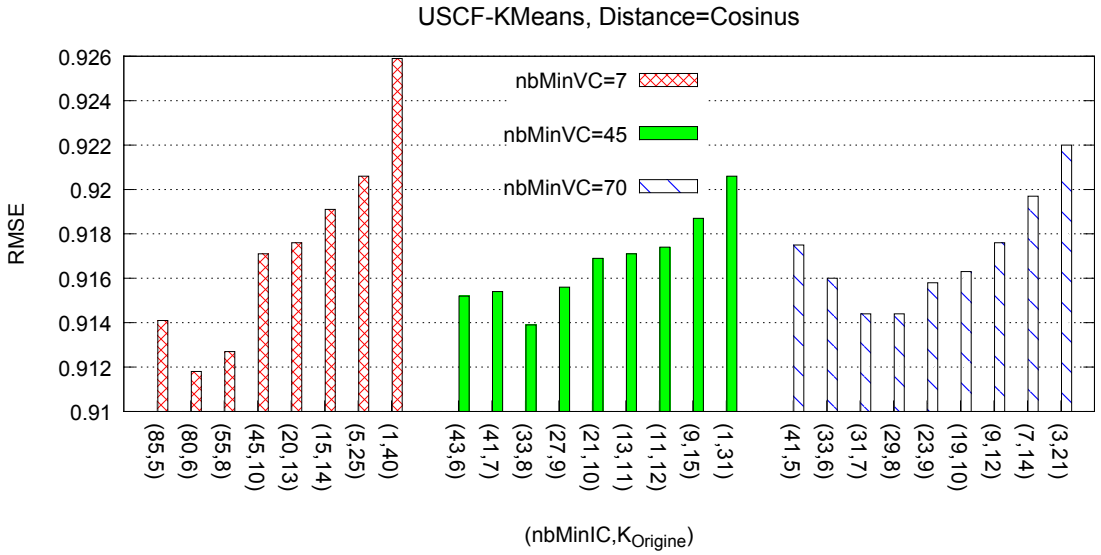


FIGURE 4.10 – Impact du nombre de clusters sur la pertinence des recommandations USCF-KMeans

$NbMinIC$. $NbMinVC$ donne le nombre minimal de votes que doit avoir un item pour être affecté initialement à un cluster. $NbMinIC$ indique le nombre minimal d'items que doit contenir un cluster pour qu'il soit créé. La figure 4.10 représente la RMSE pour 3 valeurs de $NbMinVC$ à savoir 7, 45 et 70. Pour chacune de ces valeurs, nous avons effectué des expérimentations en faisant varier $NbMinIC$ et enregistré le nombre de clusters $K_{origine}$ défini par l'algorithme d'initialisation.

En analysant la figure 4.10, on peut en conclure que la précision est liée au nombre de clusters. Pour les 3 valeurs de $NbMinVC$, on a la même tendance, la RMSE diminue jusqu'à atteindre une certaine valeur de $K_{origine}$ puis augmente. L'optimum est atteint pour un nombre de clusters entre 6 et 8 en fonction des valeurs de $NbMinVC$. Pour $NbMinVC$ égal à 7, la meilleure performance, $RMSE = 0.911$, est atteinte avec $K_{origine}$ égal à 6, pour $NbMinVC$ égal à 45 ou 70, la meilleure performance, $RMSE = 0.914$, est atteinte avec $K_{origine}$ égal à 8. Par ailleurs, la valeur de la précision se dégrade lorsque le nombre de clusters est élevé et ce pour les 3 valeurs de $NbMinVC$. Ainsi, pour $K_{origine}$ égal à 40 la RMSE est égale à 0.9260, $K_{origine}$ égal à 21 la RMSE est égale à 0.9220.

Comme pour l'algorithme Fuzzy C-Means, l'efficacité de la méthode K-Means est directement liée au nombre de clusters. Par ailleurs, le nombre de clusters retenus définit la dimension de la matrice Q_A modélisant MSUA, qui à son tour influe sur la dimension de MSU. Bien que le nombre de descripteurs, ici 43, d'un attribut indépendant, ici l'*origine*, est très inférieur au nombre d'items, réduisant ainsi le coût du calcul des similarités entre les utilisateurs, l'utilisation de K-Means permet la réduction de la dimension tout en augmentant la précision des recommandations.

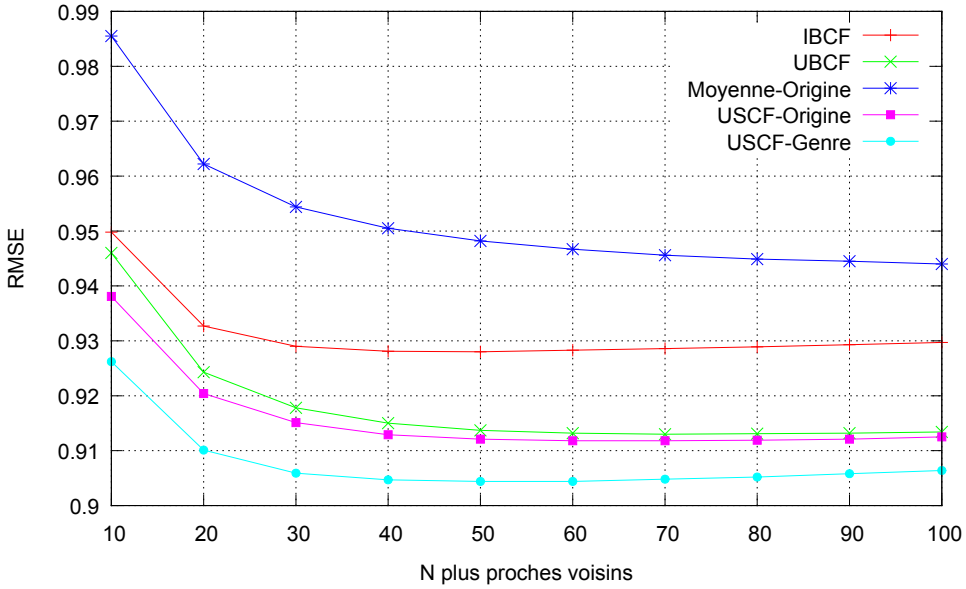


FIGURE 4.11 – Évaluation de la précision des prédictions de USCF vs IBCF, UBCF et Moyenne

Évaluation de la précision des prédictions de votes

La figure 4.11 compare la précision des prédictions de votes de notre approche USCF-KMeans (représentée par la courbe USCF-Origine sur les graphiques) par rapport aux algorithmes purement basés sur l'usage UBCF et IBCF, et à l'algorithme hybride Moyenne-Origine (voir section 3.7.2) appliqué à l'attribut *Origine*. En analysant ce graphique, on constate que toutes les courbes ont la même allure, la RMSE diminue jusqu'à une valeur donnée des N plus proches voisins puis augmente. Tous les algorithmes convergent pour N entre 50 et 60 voisins. USCF-Origine est légèrement meilleure que UBCF et ce pour toutes les valeurs de N. Moyenne-Origine présente la plus mauvaise performance.

La figure compare également les performances de nos deux approches, USCF-Fuzzy appliquée à l'attribut *Genre* dont les performances sont représentées par la courbe intitulée USCF-Genre et USCF-KMeans (USCF-Orgine sur le graphique). En analysant les deux courbes, on constate que USCF-Genre reste le plus performant. Ceci peut s'expliquer par le fait que l'attribut *Genre* soit plus prédictif que l'attribut *Origine*, c'est-à-dire qu'il est plus discriminant dans le choix de l'utilisateur, ce qui est intuitivement compréhensible. En effet, le genre d'un film peut représenter un critère de choix plus important que son origine.

Évaluation de la précision d'une Top-N liste

La figure 4.12 représente la Précision en fonction de la taille d'une Top-N liste pour 60 voisins. Comme pour le USCF-Fuzzy, USCF-KMeans appliqué à l'attribut *Origine* (USCF-Origine sur la figure) est plus performant que l'algorithme purement basé sur le contenu appliqué à l'attribut *Origine* (CB-Origine sur le graphique) et la différence est assez importante. En effet, pour $TopN = 5$ la Précision enregistrée par notre approche est égale à 0.84 alors que celle obtenue par CB-Origine est égale à 0.63. Ceci signifie que 8 recommandations sur 10 sont pertinentes en utilisant notre approche alors que seulement 6 sur 10 sont pertinentes avec une approche

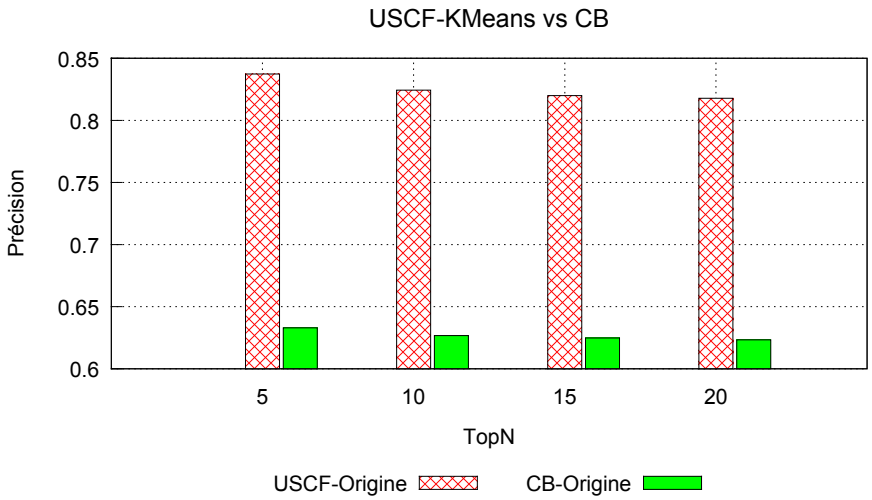


FIGURE 4.12 – Évaluation de la précision d'une Top-N liste USCF-Kmeans vs CB

purement basée sur le contenu ;

Évaluation de la diversité d'une Top-N liste

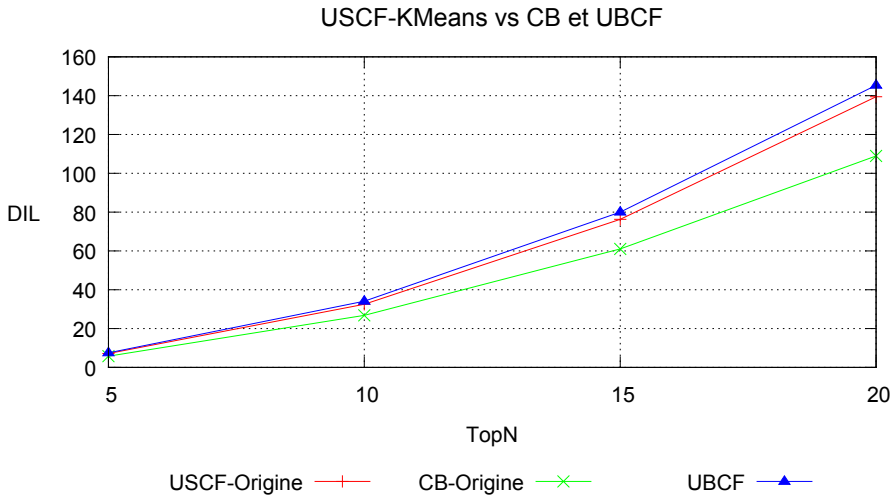


FIGURE 4.13 – Évaluation de la diversité d'une Top-N liste de USCF-Kmeans par rapport à CB et UBCF

TABLE 4.4 – Diversité (DIL) d'une Top-5 liste et Top-10 liste

	Top-5	Top-10
UBCF	7.48	34.11
USCF-Origine	7.18	32.48
CB-Origine	5.79	26.78

La figure 4.13 compare les diversités de différentes Top-N listes recommandées par notre approche USCF-KMeans (USCF-Origine sur le graphique), CB (CB-Origine sur le graphique) et UBCF. Comme nous l'avons précédemment dit, la diversité est le point fort de l'algorithme UBCF, c'est pour cette raison que sa courbe est la plus dominante. La diversité des recommandations produites par USCF-Kmeans est assez proche de celle de UBCF et loin devant celle obtenue par l'algorithme CB-Origine. Le tableau 4.4 représente les différentes valeurs de la diversité pour une Top-5 liste et une Top-10 liste. La diversité de la Top-5 liste recommandée par CB appliquée à l'attribut *Origine* (CB-Origine) est égale à 5.79, alors qu'elle est égale à 7.18 pour notre approche USCF-Kmeans appliquée à l'attribut Origine (USCF-Origine). Cette valeur est très proche de celle obtenue par l'approche purement collaborative UBCF qui est égale à 7.48.

Bien que la diversité obtenue par CB-Origine est meilleure que celle de CB-Genre, notre approche affiche une diversité de recommandations nettement supérieure, assez proche de celle affichée par l'algorithme de recommandation collaborative basé sur les utilisateurs UBCF, qui est considéré comme une référence dans la diversité des recommandations.

4.5 Conclusion

Dans ce chapitre nous avons présenté nos travaux sur l'apprentissage du modèle sémantique des utilisateurs par attribut MSUA pour les attributs indépendants. Nous avons utilisé un clustering flou en appliquant la méthode Fuzzy C-Means pour la construction de MSUA pour les attributs multivalués. La méthode K-Means a été utilisée pour la construction du MSUA pour les attributs monovalués. Le nombre, ainsi que les centres initiaux des clusters sont déterminés automatiquement en utilisant l'aspect sémantique des items. L'approche que nous proposons permet de réduire la dimension de la matrice Q modélisant le modèle sémantique des utilisateurs MSU. Comme le MSU est utilisé pour calculer la similarité entre les utilisateurs dans un algorithme de filtrage collaboratif basé sur les utilisateurs, le coût du calcul des similarités sera également réduit. Nous rappelons que l'un des principaux inconvénients des algorithmes de filtrage collaboratif basés sur les voisins est le passage à l'échelle, essentiellement dû au coût nécessaire aux calculs des similarités entre les utilisateurs (voir section 2.1.3).

Nous avons évalué les deux approches (USCF-Fuzzy et USCF-KMeans) sur les jeux de données de MovieLens, l'un des jeux les plus utilisés dans le domaine des systèmes de recommandation. Nous avons appliqué notre approche USCF-Fuzzy sur l'attribut genre et USCF-KMeans sur l'attribut origine. Nous avons évalué notre approche par rapport à des algorithmes purement basés sur l'usage (l'algorithme de FC basé sur les utilisateurs UBCF, et l'algorithme FC basé sur les items IBCF), à un algorithme purement basé sur le contenu CB, et à un algorithme hybride qui calcule le profil sémantique des utilisateurs par attribut en calculant la moyenne des votes des utilisateurs sur les descripteurs de l'attribut. Nous avons évalué ces algorithmes en termes de précision des prédictions, précision d'une Top-N liste et diversité des recommandations d'une Top-N liste.

L'attribut *genre* a donné de meilleurs résultats que l'attribut *origine*. Ceci peut s'expliquer par le fait que l'attribut genre est plus pertinent dans le choix de l'utilisateur que l'attribut origine ce qui est intuitivement cohérent. Notre approche, USCF appliqué à l'attribut genre, a donné les meilleures précisions de prédiction comparé à UBCF, IBCF. Ceci peut avoir plusieurs explications. Premièrement que l'attribut *genre* est un attribut pertinent dans le choix

des utilisateurs. Deuxièmement que l'introduction de l'aspect sémantique des items dans une approche collaborative peut améliorer la précision des recommandations. Troisièmement, que le MSU ne contient pas de données manquantes, ce qui a permis de calculer la similarité entre tous les utilisateurs même ceux n'ayant aucun vote en commun. Enfin, que le processus permettant de construire le modèle sémantique des utilisateurs (MSU) est lui même collaboratif, les votes de tous les utilisateurs sont utilisés pour déterminer le profil sémantique de chaque utilisateur. Notre approche a donné de meilleurs résultats qu'une approche purement basée sur le contenu et ce pour les deux attributs *genre* et *origine*.

La diversité est un des points faibles des systèmes de recommandation basés sur le contenu (CB) en raison du principe de fonctionnement même de ces algorithmes. Cependant, les algorithmes de filtrage collaboratif, et particulièrement ceux basés sur les utilisateurs (UBCF), offrent une large diversité dans les recommandations qu'ils proposent. Nous avons comparé notre approche à UBCF et CB en terme de la diversité des recommandations. Pour ce faire, nous avons calculé la DIL (Diversité Intra Liste) de chaque algorithme. Les résultats obtenus sont largement supérieurs à ceux obtenus par un algorithme CB et très proches des performances de l'algorithme UBCF.

Dans le chapitre suivant, nous présentons nos travaux pour les attributs que nous avons défini comme étant dépendants. Les solutions que nous proposons sont inspirées du domaine de la recherche et du filtrage d'information.

Chapitre 5

Apprentissage du profil sémantique des utilisateurs pour les attributs dépendants

Ce chapitre est consacré à la présentation du composant *construction du Modèle Sémantique des Utilisateurs par Attribut* MSUA que nous avons introduit à la section 3.4.3. Nous rappelons que ce composant permet de construire le profil sémantique des utilisateurs pour un seul attribut pertinent. Les solutions que nous détaillons dans ce chapitre sont essentiellement destinées aux attributs dépendants (voir section 3.4.3), cependant, il est également possible de les appliquer aux attributs indépendants. Nous rappelons qu'un attribut dépendant est un attribut dont le nombre de descripteurs varie avec le nombre d'items. C'est-à-dire que le nombre de valeurs possibles pour un tel attribut est en général assez élevé et peut dans certains cas être supérieur au nombre d'items.

Nous rappelons par ailleurs, que le Modèle Sémantique des Utilisateurs par Attribut, MSUA, est représenté par la matrice $Q_{A_{(|U|,|D_A|)}}$ (voir équation (3.12)). D_A étant la liste des descripteurs de l'attribut A décrits dans la dimension des utilisateurs. Chaque descripteur de D_A est défini par un vecteur colonne de la matrice Q_A. On notera par $K_A = |D_A|$ la dimension du MSUA.

Nous présentons dans ce chapitre deux contributions pour l'apprentissage du MSUA pour les attributs dépendants. La section 5.1 décrit la première approche dont les résultats ont été publiés dans [Ben Ticha *et al.*, 2013]. La section 5.2 présente la seconde approche publiée dans [Ben Ticha *et al.*, 2014] qui a été nominée au *best student paper*. Une version étendue [Ben Ticha *et al.*, 2015] a été sollicitée et soumise pour publication. Les deux approches s'appuient sur des techniques issues de la recherche et du filtrage d'information [Salton, 1989; Buckley and Salton, 1995b; Belkin and Croft, 1992]. Pour chacune des solutions, nous proposons différentes techniques pour la réduction de la dimension du MSUA.

5.1 Fréquence des votes par descripteur pour l'apprentissage du MSUA

L'idée est de déterminer l'importance (le poids) d'un descripteur pour un utilisateur en utilisant la mesure Term-Frequency/Inverse Document Frequency (TFIDF). TFIDF compte parmi les mesures les plus utilisées et les plus populaires dans le domaine de la recherche documentaire et le filtrage d'informations [Baeza-Yates and Ribeiro-Neto, 1999]. Elle a été utilisée dans les systèmes de recommandation, essentiellement ceux basés sur le contenu [Lang, 1995; Niwa *et al.*, 2006], mais également dans les systèmes de recommandation hybrides [Diederich and Iofciu, 2006; Symeonidis *et al.*, 2007; Shepitsen *et al.*, 2008].

Cette section est composée de trois parties. La première partie (section 5.1.1) est consacrée à la présentation de notre approche pour l'apprentissage du $MSUA$. Étant donné que la dimension du $MSUA_A$ est déterminée par le nombre de descripteurs de l'attribut A, et vu que le nombre de descripteurs d'un attribut dépendant peut être élevé, la dimension du MSUA peut également être élevée. C'est pourquoi, dans la seconde partie (section 5.1.2), nous présentons deux méthodes pour réduire la dimension du MSUA. La première en filtrant les descripteurs de l'attribut A pour ne conserver que les plus pertinents (section 5.1.2.1). La seconde en appliquant une Analyse Sémantique Latente (LSA) [Dumais, 2004] sur le MSUA. La troisième partie (section 5.1.3) compare d'une part, les résultats obtenus par les deux méthodes de réduction de dimension, et évalue d'autre part, les performances de notre approche par rapport à une approche purement collaborative et une approche basée sur le contenu en termes de précision et de diversité des recommandations.

5.1.1 Apprentissage du MSUA

TFIDF est une mesure statistique utilisée initialement pour refléter l'importance d'un terme dans un document faisant parti d'un corpus de documents. Dans notre cas, nous l'appliquons pour mesurer l'importance d'un descripteur pour un utilisateur. La mesure TFIDF exige au préalable la détermination d'une fréquence reliant un utilisateur à un descripteur. Dans le cas de la recherche documentaire, la fréquence représente le nombre d'occurrences d'un terme dans un document. Dans notre cas, la fréquence est définie à partir du nombre de votes attribué par l'utilisateur au descripteur. Pour ce faire, nous proposons quatre méthodes différentes pour calculer la fréquence des votes (voir section 5.1.1.1). Le MSUA est alors obtenu en appliquant la mesure du TFIDF à la matrice des fréquences des votes (voir section 5.1.1.2). La section 5.1.1.3 résume dans un algorithme les différentes étapes pour l'apprentissage du MSUA. Enfin, la section 5.1.1.5 compare les performances des quatre méthodes pour le calcul de la fréquence des votes sur les attributs *Réalisateur*, *Acteur*, *Mot-clé* et *Tag*.

5.1.1.1 Construction de la Matrice des Fréquences des Votes

On définit la fonction de fréquence des votes par utilisateur par $freq_u : f_d \in F_A \mapsto freq_u(f_d) \in \mathbb{N}$. $freq_u(f_d)$ compte le nombre de votes associés par l'utilisateur u au descripteur f_d à partir de ses votes pour les items décrits par f_d. La Matrice des Fréquences des Votes MFV est alors définie comme suit :

$$MVF = (freq_u(f_d))_{(u=1..|U| \ \ et \ \ d=1..|F_A|)} \tag{5.1}$$

La question qui se pose est alors : comment définir la fonction $freq_u$? Est-ce qu'il faut tenir compte de tous les items notés par l'utilisateur u ? Est ce qu'il est préférable de ne considérer que les items pertinents ? Est-ce qu'il faut traiter tous les items de façon identique ou les pondérer par leurs votes ? Nous présentons dans ce qui suit quatre méthodes pour définir la fonction de fréquence $freq_u$. Il est à noter que nous avons conservé la notation anglaise présentée dans nos travaux de recherche [Ben Ticha *et al.*, 2013].

Pour illustrer les quatre méthodes par un exemple, nous supposons disposer de la matrice des votes Mv décrite par le tableau 5.2 et de la Matrice Sémantique des Items par Attribut (MSIA) (voir formule (3.9), page 64) décrite par le tableau 5.1. Les votes sont entiers ($\in \mathbb{N}$), définis sur l'échelle de valeurs $V = 1..5$. Le tableau 5.2 donne également pour chaque utilisateur u la moyenne de ses votes $\overline{v_u}$.

TABLE 5.1 – Matrice sémantique de items par attribut MSIA

	f_1	f_2	f_3	f_4
i_1	0	1	0	1
i_2	1	0	1	1
i_3	1	0	1	0
i_4	0	0	1	1
i_5	1	0	1	0
i_6	1	1	0	1

TABLE 5.2 – Matrice des votes des utilisateurs Mv avec moyenne des votes

	i_1	i_2	i_3	i_4	i_5	i_6	$\overline{v_u}$
u_1	1	?	?	3	4	?	2.66
u_2	?	4	1	?	?	2	2.33
u_3	3	4	5	2	?	?	3.5
u_4	4	?	2	?	1	1	2

Fonction de fréquence pour tous les items :$freq_{u_{\text{NW\_NR}}}$

La première fonction $freq_{u_{\text{NW\_NR}}}$ considère tous les items notés par l'utilisateur u sans aucune distinction. $freq_{u_{\text{NW\_NR}}}(f_d)$ consiste à compter le nombre de fois où u a voté pour un item i décrit par le descripteur f_d telle que le définit la formule ((5.2)). Le suffixe NW_NR signifie "*No Weighted No Relevant*" et indique qu'aucune pondération n'est appliquée pour le calcul de la fonction de fréquence et que tous les items sont pris en compte (les pertinents et les non pertinents).

$$freq_u(f_d) = freq_{u_{\text{NW\_NR}}}(f_d) = \sum_{i \in I_u} fréqence_i(f_d) \tag{5.2}$$

Nous rappelons que la fonction $fréqence_i(f_d)$, définie par la formule (3.7) à la page 63, donne le nombre d'occurrences du descripteur f_d dans l'item i, et que I_u est l'ensemble des items notés par u. En appliquant la fonction $freq_{u_{\text{NW\_NR}}}$ sur l'exemple ci-dessus, nous obtenons la MFV illustrée par le tableau 5.3.

TABLE 5.3 – Matrice des fréquences des votes MFV en appliquant la fonction $freq_{u_{\text{NW\_NR}}}$

	f_1	f_2	f_3	f_4
u_1	1	1	2	2
u_2	3	1	2	2
u_3	2	1	3	3
u_4	3	2	2	2

Fonction de fréquence pondérée par la valeur des votes pour tous les items :$freq_{u_{\mathbf{W\_NR}}}$

La formule (5.2) traite tous les items de la même façon. Pour tenir compte de la différence entre les items, nous introduisons le vote de chaque item dans la définition de la fonction de fréquence $freq_{u_{\text{W\_NR}}}$. Ainsi, la fonction $fréqence_i$ est pondérée par le vote v_{ui} de u sur i. Le suffixe W_NR signifie "*Weighted No Relevant*", il indique qu'une pondération est appliquée pour le calcul de la fonction de fréquence et que tous les items sont sélectionnés (les pertinents et les non pertinents).

$$freq_u(f_d) = freq_{u_{\text{W\_NR}}}(f_d) = \sum_{i \in I_u} fréqence_i(f_d) \times v_{ui} \tag{5.3}$$

Le tableau 5.4 illustre la MVF construite en appliquant la fonction $freq_{u_{\text{W\_NR}}}$ sur l'exemple ci-dessus.

TABLE 5.4 – Matrice des fréquences des votes MFV en appliquant la fonction $freq_{u_{\text{W\_NR}}}$

	f_1	f_2	f_3	f_4
u_1	4	1	7	4
u_2	7	2	5	6
u_3	9	3	11	9
u_4	4	5	3	5

Fonction de fréquence pour les items pertinents :$freq_{u_{\mathbf{NW\_R}}}$

Dans ce cas, seuls les items pertinents pour l'utilisateur u sont considérés. On note par $I_{u_{pertinent}}$ l'ensemble des items pertinents de l'utilisateur u défini par la formule 5.4. Nous considérons qu'un item i est pertinent pour un utilisateur u de U s'il vérifie les deux conditions suivantes :

1. le vote de u pour l'item i doit être supérieur ou égal à la moyenne de ses votes :$v_{ui} \geq \overline{v_u}$.
2. soit $V = v_{min}..v_{max}$ l'échelle de valeurs des votes, on définit par $v_{neutre} = v_{max}/2$ le vote neutre. Le vote $v_{u,i}$ de u doit être supérieur au vote neutre ($v_{ui} > v_{neutre}$). Cette condition garantit que le vote d'un item pertinent soit supérieur au vote neutre.

$$\begin{cases} V = v_{min}..v_{max} \; et \; v_{neutre} = v_{max}/2 \\ I_{u_{pertinent}} = \{i \in I_u / \; v_{ui} \geq \overline{v_u} \; et \; v_{ui} > v_{neutre}\} \end{cases} \tag{5.4}$$

L'utilisation de la moyenne des votes de l'utilisateur comme seuil pour déterminer la pertinence d'un item a deux avantages. Le premier étant d'éviter l'ajout d'un nouveau paramètre. Le second est la personnalisation du seuil qui permet de tenir compte de la variation dans l'attribution des notes puisque tous les utilisateurs ne notent pas de la même façon. La fonction de fréquence des votes $freq_{u_{\text{NW\_R}}}$ est alors donnée par la formule (5.5). Le suffixe NW_R signifie

"*No Weighted Relevant*" et indique qu'aucune pondération n'est appliquée pour le calcul de la fonction de fréquence et que seuls les items pertinents sont sélectionnés.

$$freq_u(f_d) = freq_{u_{\text{NW\_R}}}(f_d) = \sum_{i \in I_{u_{pertinent}}} fréqence_i(f_d) \tag{5.5}$$

La matrice des fréquences des votes MFV obtenue en appliquant la fonction $freq_{u_{\text{NW\_R}}}$ sur l'exemple illustré par les tableaux 5.2 et 5.1 est représentée par le tableau 5.5, il est à noter que le vote neutre est égal à $v_{neutre} = 2.5$.

TABLE 5.5 – Matrice des fréquences des votes MFV en appliquant la fonction $freq_{u_{\text{NW\_R}}}$

	f_1	f_2	f_3	f_4
u_1	1	0	2	1
u_2	1	0	1	1
u_3	2	0	2	1
u_4	0	1	0	1

Fonction de fréquence des votes pondérée pour les items pertinents :$freq_{u_{\text{W\_R}}}$

La fonction de fréquence $freq_{u_{\text{W\_R}}}$ pondère la fonction $fréqence_i(f_d)$ des items pertinents par la valeur du vote. Le suffixe W_R signifie "*Weighted Relevant*" et indique qu'une pondération est appliquée pour le calcul de la fonction de fréquence et que seuls les items pertinents sont sélectionnés.

$$freq_u(f_d) = freq_{u_{\text{W\_R}}}(f_d) = \sum_{i \in I_{u_{pertinent}}} fréqence_i(f_d) \times v_{ui} \tag{5.6}$$

Le tableau 5.6 illustre la MFV obtenue en appliquant la fonction $freq_{u_{\text{W\_R}}}$ sur l'exemple décrit par les tableaux 5.2 et 5.1.

TABLE 5.6 – Matrice des fréquences des votes MFV en appliquant la fonction $freq_{u_{\text{W\_R}}}$

	f_1	f_2	f_3	f_4
u_1	4	0	7	3
u_2	4	0	4	4
u_3	9	0	9	7
u_4	0	4	0	4

5.1.1.2 Construction du modèle sémantique des utilisateurs par attribut MSUA

La matrice Q_A modélisant le MSUA est obtenue en appliquant la mesure Term Frequency/Inverse User Frequency (TFIDF) sur la matrice des fréquences des votes MFV. Comme nous l'avons déjà dit ci-dessus, TFIDF est une mesure statistique qui reflète l'importante d'un terme dans un document faisant partie d'un corpus de documents. Dans notre cas, nous remplaçons le terme par le descripteur (feature en anglais) et le document par l'utilisateur, nous obtenons ainsi Feature Frequency/Inverse User Frequency (FFIUF) [32]. La mesure FFIUF est donnée par les formules

32. Nous avons conservé la notation en anglais telle qu'elle a été publiée dans nos travaux de recherche

(5.7), (5.8) et (5.9).

$$FF(u, f_d) = \frac{freq_u(f_d)}{max_j freq_u(j)} \tag{5.7}$$

La mesure de Inverse User Frequency IUF est définie par :

$$IUF(f_d) = \log \frac{|U|}{|U_{f_d}|} \tag{5.8}$$

avec U_{f_d} étant l'ensemble des utilisateurs ayant une fréquence de votes non nulle, c'est-à-dire $freq_u(f_d) \neq 0$.

La mesure FFIUF est alors donnée par :

$$FFIUF(u, f_d) = FF(u, f_d) \times IUF(f_d) \tag{5.9}$$

Ainsi, la matrice $Q_A = (q(u, f_d))_{(u=1..|U|,d=1..|F_A|)}$ modélisant le MSUA est donnée par l'équation (5.10). Le nombre K_A de colonnes de Q_A est alors égal à $|F_A|$.

$$Q_A = (FFIUF(u, f_d))_{(u=1..|U|,d=1..|F_A|)} \tag{5.10}$$

5.1.1.3 Algorithme

La méthode de construction du MSUA appliquant la fréquence des votes par descripteur sera appelée dans tout ce qui suit FFIUF pour (Feature Frequency Inverse User Frequency). Le nom de la méthode sera suffixé par la fonction de calcul de la fréquence utilisée. Ainsi, si par exemple la fonction $freq_{u_{\text{NW\_R}}}$ est utilisée pour construire la matrice des fréquences des votes MFV alors le nom de la méthode sera FFIUF-NW_R.

La construction du MSUA modélisé par la matrice Q_A est décrite par l'algorithme 5.1. Il est composé de deux étapes. La première (ligne 2) construit la matrice des fréquences des votes MFV en appliquant l'une des quatre méthodes présentées ci-dessus. La deuxième étape applique la mesure du FFIUF sur la matrice MFV et produit la matrice Q_A.

Algorithme 5.1 : FFIUF

Données :
- A : attribut,
- Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
- $MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
- Méthode de calcul de la fonction de fréquence parmi les 4 méthodes définies dans la section 5.1.1.1.

Résultat :
- Q_A : matrice modélisant MSUA, les $|U|$ utilisateurs en ligne et K_A descripteurs de A en colonne.
- K_A : nombre de colonnes de Q_A

```
début
    /* Construction de la matrice des fréquences des votes MFV_(|U|,|F_A|) en
       appliquant la méthode choisie                                          */
    pour chaque (u ∈ U) faire
        pour chaque (f_d ∈ F_A) faire
            MFV(u, f_d) = freq_u(f_d)
        fin
    fin
    /* Application de la mesure TFIDF sur la MVF                              */
    Q_A = FFIUF(MFV)
    K_A = |F_A|
fin
```

5.1.1.4 Recommandation

Nous noterons dans ce qui suit par USCF-FFIUF, l'algorithme de recommandation $USCF$ appliquant l'algorithme FFIUF pour la construction du modèle sémantique des utilisateurs par attribut MSUA. Comme nous l'avons déjà mentionné dans la section 3.5, la prédiction des votes dans notre approche $USCF$ est calculée en appliquant l'algorithme de filtrage collaboratif basé sur les utilisateurs. Dans sa version standard $UBCF$ (voir section 2.1.1.1), la similarité entre les utilisateurs est calculée à partir de la matrice des votes Mv. Dans notre cas nous utilisons la matrice Q_A modélisant le MSUA pour calculer la similarité entre les utilisateurs. En reprenant l'exemple de la section 5.1.1.1, et en raison des données manquantes, nous constatons que les utilisateurs u_1 et u_2 n'ont aucun item en commun. $UBCF$ ne peut par conséquent pas détecter les similarités entre ces deux utilisateurs. Cependant, en analysant les matrices des fréquences des votes issues des quatre méthodes de calcul de la fonction de fréquence des votes (voir section 5.1.1.1), il est aisé de constater que les deux utilisateurs u_1 et u_2 sont assez proches.

Le profil utilisateur construit à partir de l'algorithme FFIUF permet de détecter des similarités entre des utilisateurs n'ayant aucun item en commun. Ce qui permet de traiter le problème de la transitivité des voisins (voir section 2.1.3), conséquence du problème des données manquantes, auquel sont confrontés les algorithmes de filtrage collaboratif basés sur les voisins. Ainsi, le MSUA permet d'affiner le processus de sélection des plus proches voisins de l'utilisateur courant. La sélection des plus proches voisins a une influence directe sur la pertinence des recommandations dans un algorithme collaboratif.

5.1.1.5 Évaluation des quatre méthodes de calcul de la fonction de fréquence

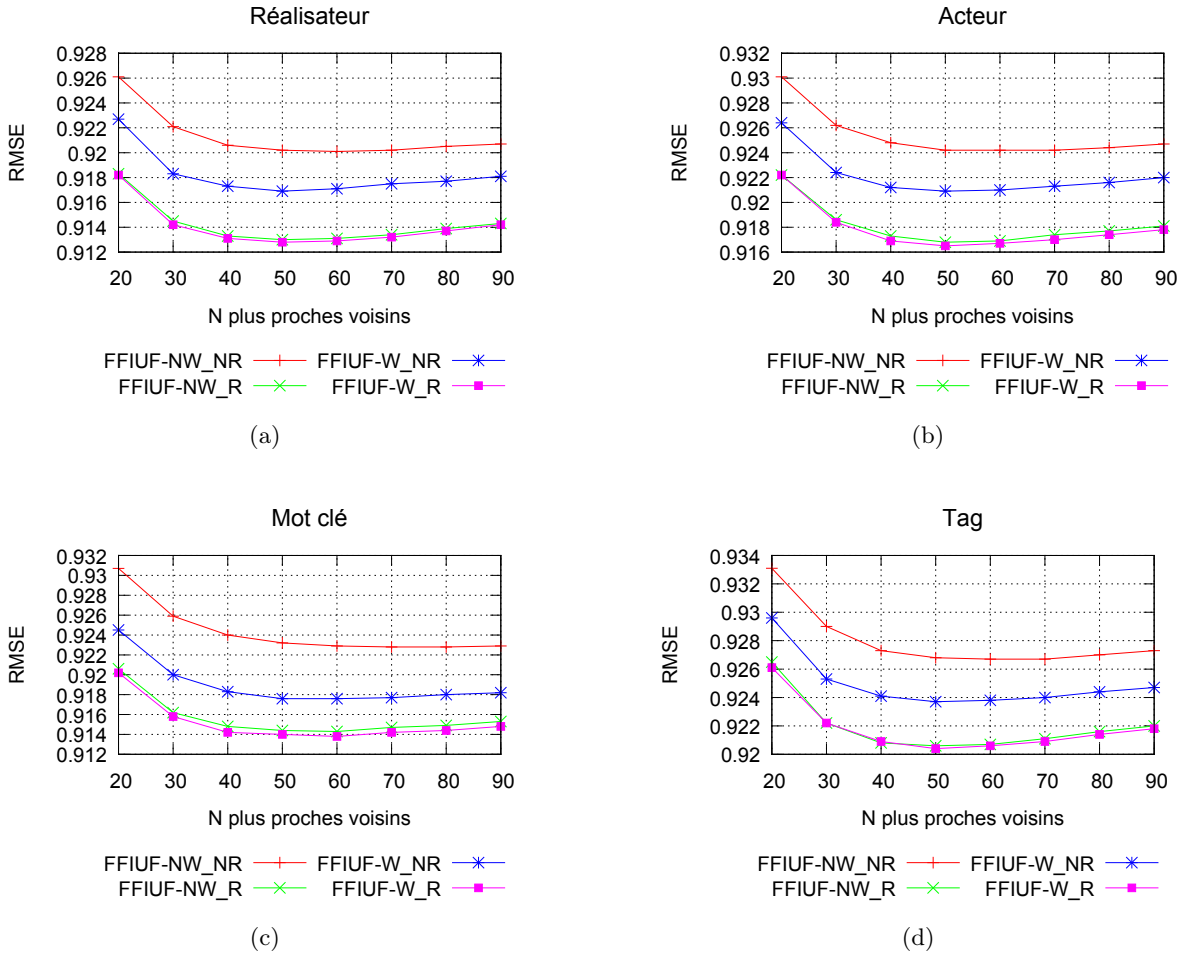


FIGURE 5.1 – Évaluation des quatre méthodes de calcul de la fonction de fréquence

Nous présentons dans cette section l'évaluation des quatre méthodes de calcul de la fonction de fréquence présentées à la section 5.1.1.1 pour la construction du MSUA. Les expérimentations ont été menées sur les attributs dépendants *Réalisateur* (figure 5.1(a)), *Acteur* (figure 5.1(b)) , *Tag* (figure 5.1(d)) et *Mot Clé* (figure 5.1(c)). L'évaluation est effectuée en utilisant la RMSE qui mesure la précision des prédictions. Nous rappelons que plus la RMSE est faible, meilleure est la précision.

La figure 5.1 représente pour chacun des attributs cités ci-dessus la RMSE en fonction du nombre N des plus proches voisins pour chacune des méthodes. En observant la figure 5.1(a), on constate d'une part, que les 4 courbes relatives aux 4 méthodes de calcul ont la même tendance et qu'elles convergent pour N entre 50 et 60 voisins, d'autre part, que l'élimination des items non pertinents dans le calcul de la fonction de fréquence améliore la précision des prédictions. Les meilleures performances sont enregistrées par la méthode FFIUF-W_R qui présente des résultats très proches de la méthode FFIUF-NW_R, les plus mauvaises sont enregistrées par la méthode FFIUF-NW_NR. Ce constat reste le même pour les autres attributs. Dans tout ce qui

suit, la méthode FFIUF-W_R sera appliquée pour la construction du MSUA dans l'algorithme USCF-FFIUF.

5.1.2 Réduction de la dimension du MSUA

La dimension du MSUA est définie par le nombre K_A de colonnes de la matrice Q_A. Deux approches sont alors possibles pour réduire la dimension du MSUA. La première consiste à réduire initialement la dimension de la matrice sémantique des items par attribut MSIA. La seconde consiste à réduire la dimension du MSUA en appliquant une méthode de réduction directement sur la matrice Q_A.

Pour la réduction de la dimension de la matrice MSIA, nous proposons une méthode de réduction basée sur le filtrage des descripteurs (voir section 5.1.2.1). Pour la réduction du MSUA nous proposons l'application d'une Analyse Sémantique Latente (LSA) sur la matrice Q_A (voir section 5.1.2.2).

5.1.2.1 Réduction de la dimension de la MSIA par filtrage des descripteurs

Comme la dimension de la matrice MSIA définit la dimension de la matrice Q_A alors réduire la dimension de Q_A revient à réduire la dimension de la MSIA. Il s'agit de sélectionner parmi l'ensemble initial des descripteurs F_A de A, un sous ensemble de descripteurs pertinents ayant un nombre de votes supérieur ou égal à un seuil donné μ. Pour ce faire, on définit par η la fonction déterminant le nombre de votes par descripteur comme suit :

$$\eta : f_d \in F_A \mapsto \eta(f_d) = \sum_{u \in U} \sum_{i \in I_u} présence_i(f_d) \tag{5.11}$$

On rappelle que $présence_i(f_d)$ (voir formule (3.6)) est une fonction binaire qui est égale à 1 si f_d décrit l'item i, 0 sinon. La fonction $\eta(f_d)$ donne le nombre de votes associés au descripteur f_d. Le sous-ensemble de descripteurs pertinents F_{A_μ} est alors défini par :

$$F_{A_\mu} = \{f_d \in F_A \,/\, \eta(f_d) \geq \mu\} \tag{5.12}$$

Algorithme

En se limitant aux descripteurs de F_{A_μ}, la dimension de la matrice sémantique des items par attribut MSIA sera réduite, ce qui aura pour effet de réduire la dimension de la matrice Q_A. La valeur du seuil μ est définie de façon empirique. La construction du MSUA en appliquant le filtrage des descripteurs comme mécanisme de réduction de la dimension est décrite par l'algorithme 5.2. Une étape de filtrage des descripteurs (ligne 3) précédant l'étape de construction de la matrice des fréquences des votes MFV a été ajoutée par rapport à l'algorithme initial 5.1. La méthode de fréquence $freq_{u_{W\_R}}$ a été retenue pour calculer les fréquences des votes des utilisateurs (ligne 10 de l'algorithme 5.2) puisqu'elle a donné les meilleurs résultats comme le montre la figure 5.1.

Algorithme 5.2 : FFIUF avec filtrage des descripteurs de A

Données :

A : attribut,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
μ : seuil déterminant les descripteurs pertinents.

Résultat :

Q_A : matrice modélisant MSUA, les $|U|$ utilisateurs en ligne et K_A descripteurs pertinents de A en colonne.
K_A : nombre de colonnes de Q_A.

début
  /* Filtrage des descripteurs de A */
  $F_{A_\mu} = \emptyset$
  pour chaque *($f_d \in F_A$)* **faire**
    /* Construire l'ensemble F_{A_μ} des descripteurs pertinents selon la formule (5.12) */
    . **si** $\eta(f_d) \geq \mu$ **alors**
      ajouter f_d à F_{A_μ}
    fin
  fin
  /* Construction de la matrice des fréquences des votes $MFV_{|U|,|F_{A_\mu}|}$. */
  pour chaque *($u \in U$)* **faire**
    pour chaque *($f_d \in F_{A_\mu}$)* **faire**
      $MFV(u, f_d) = freq_{u_{\text{W\_R}}}(f_d)$
    fin
  fin
  /* Application de la mesure TFIDF sur la MVF */
  $Q_A = \text{FFIUF}(MFV)$
  $K_A = |F_{A_\mu}|$
fin

Évaluation

La figure 5.2 illustre la performance de l'algorithme USCF-FFIUF en appliquant le filtrage des descripteurs comme mécanisme de réduction de la dimension par rapport au même algorithme sans réduction de la dimension, et ce pour les quatre attributs dépendants *Réalisateur*, *Acteur*, *Mot-clé* et *Tag*. Les courbes représentent la RMSE en fonction du nombre de descripteurs de l'attribut correspondant, c'est-à-dire la dimension du modèle sémantique des utilisateurs par attribut (MSUA). En observant les courbes de réduction de la dimension, on constate qu'elles affichent la même tendance pour les quatre attributs. La réduction de la dimension augmente la précision des recommandations essentiellement pour les attributs *Réalisateur* (figure 5.2(a)) et *Acteur* (figure 5.2(b)). Ceci peut s'expliquer par le fait que ces deux attributs ont un intérêt pour l'utilisateur et influent sur ses préférences. En effet, le réalisateur ainsi que les acteurs d'un film peuvent jouer un rôle important dans la détermination des préférences des utilisateurs dans un système de recommandation de films. L'algorithme FFIUF-W_R détermine le profil sémantique des utilisateurs (MSUA) en comptabilisant les votes pertinents par descripteur. Ainsi, le filtrage

des réalisateurs (ou des acteurs) en éliminant ceux n'ayant pas d'influence dans le choix des utilisateurs, c'est-à-dire n'ayant pas assez de votes (nombre de votes inférieur au seuil μ), a amélioré la sélection des voisins d'un utilisateur. Pour les attributs *Tag* et *Mot Clé*, le filtrage des descripteurs n'améliore pas la précision des prédictions. Le tableau 5.7 donne les valeurs de la RMSE avec et sans réduction de la dimension pour les quatre attributs. Pour les attributs *Tag* et *Mot Clé*, la réduction de la dimension peut atteindre le taux de 80% avec une perte en précision ne dépassant pas 0.002.

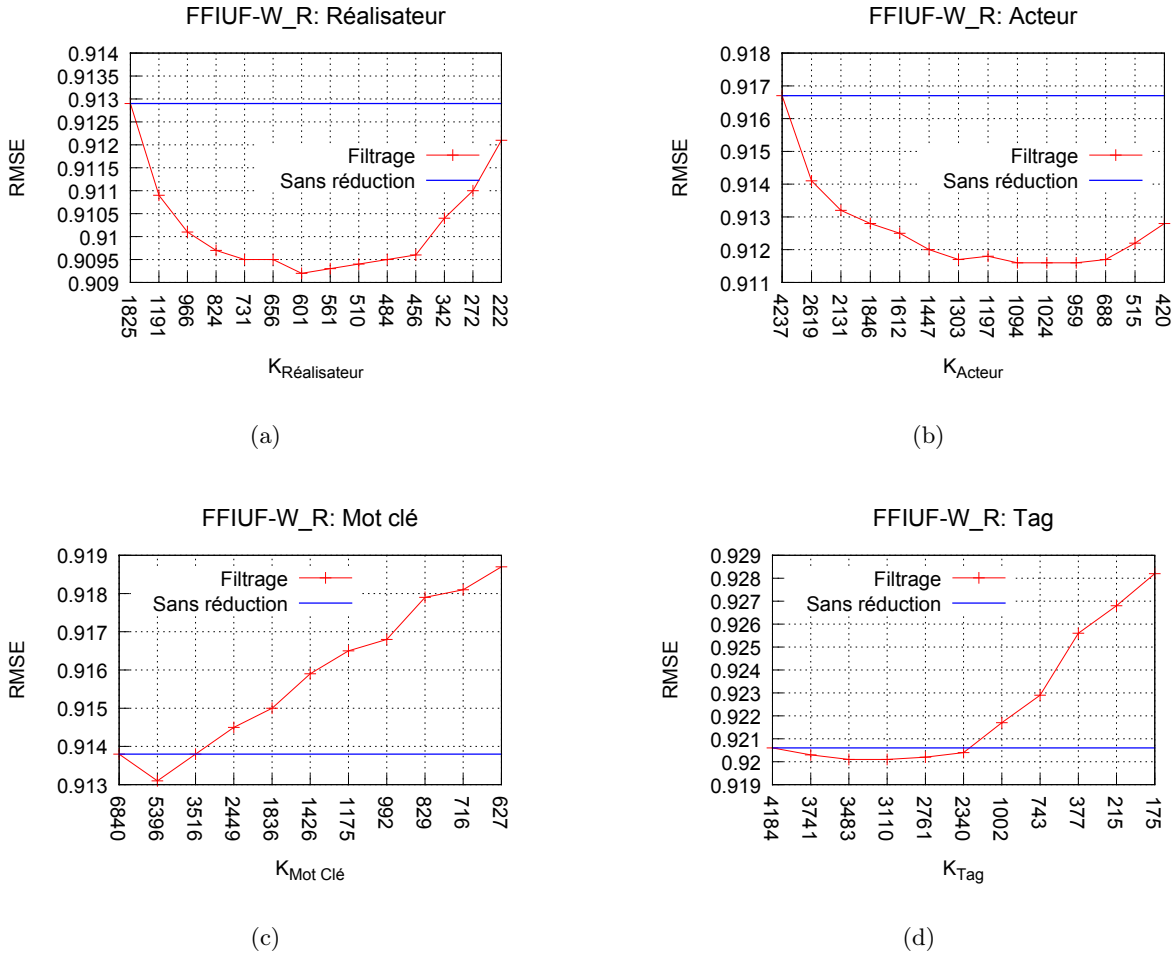


FIGURE 5.2 – USCF-FFIUF avec filtrage des descripteurs pour réduire la dimension du MSUA

TABLE 5.7 – Précision des prédictions de USCF-FFIUF avec filtrage des descripteurs

A	$\|F_A\|$	$RMSE_{\|F_A\|}$	K_A	$RMSE_{K_A}$	Gain RMSE	% Réduction
Réalisateur	1825	0.9130	601	0.9093	+0.0037	67.07%
Acteur	4237	0.9167	1075	0.9116	+0.0052	74.63%
Mot Clé	6840	0.9138	5396	0.9131	+0.0007	21.11%
			3516	0.9138	0.000	48.6%
			1426	0.9159	−0.0021	79.15%
Tag	4184	0.9206	3277	0.92	+0.0006	21.68%
			1821	0.9209	−0.0003	56.48%
			839	0.9225	−0.0019	79.95%

5.1.2.2 Réduction de la dimension de la matrice Q_A par application d'une LSA

Pour la réduction du MSUA, nous proposons comme seconde solution, d'appliquer une Analyse Sémantique Latente (LSA)[Dumais, 2004] de rang K sur la matrice Q_A. Le rang K doit être inférieur au nombre de descripteurs de l'attribut A ($K << |F_A|$). Comme nous l'avons déjà présenté à la section 2.1.4, la LSA utilise une Décomposition en Valeurs Singulières (Singular Value Decomposition SVD) tronquée (voir formule 2.15), conservant uniquement les K plus grandes valeurs singulières et leurs vecteurs associés. Elle permet ainsi de projeter les utilisateurs dans une dimension réduite définie par des variables latentes.

La décomposition en valeurs singulières (SVD) est une méthode de factorisation de matrice ayant l lignes et c colonnes. Elle permet de projeter une dimension de la matrice (soit les lignes soit les colonnes) sur une autre dimension définie par des variables latentes décrites par les valeurs singulières de la matrice initiale. La dimension de la projection est définie par le nombre de valeurs singulières de la matrice initiale qui est égal au minimum entre l et c. La LSA réduit la dimension de projection en ne conservant que les k plus grandes valeurs singulières. La SVD a été utilisée dans les systèmes de recommandation afin de traiter les problèmes du passage à l'échelle en réduisant la dimension des données (en appliquant une LSA) ou les problèmes des données manquantes dans des algorithmes de recommandation collaboratifs [Sarwar *et al.*, 2002; Funk., 2006; Koren, 2008; Luo *et al.*, 2013] ou hybrides [Canny, 2002; Mobasher *et al.*, 2004; Barragáns-Martínez *et al.*, 2010; Manzato, 2012]. La factorisation de matrice appliquée aux algorithmes de recommandation a connu un intérêt croissant de la part des chercheurs à la suite des travaux de [Koren, 2009] ayant remporté le premier prix du Netflix Challenge [Bennett and Lanning, 2007].

La formule 5.13 donne la matrice d'approximation de rang K de la matrice Q_A après lui avoir appliqué une LSA de rang K.

$$Q_A \approx B_{|U|,k} * \Sigma_{k,k} * V^t_{k,|F_A|} \tag{5.13}$$

Les lignes dans $B_{|U|,k}$ représentent les vecteurs utilisateurs définis dans l'espace LSA. Les lignes dans $V_{|F_A|,k}$, représentent les vecteurs descripteurs définis dans l'espace LSA. Ainsi, chaque utilisateur est modélisé dans l'espace LSA par k variables latentes. La nouvelle dimension du MSUA est alors définie par le rang k de la LSA appliquée. Par ailleurs, les variables latentes générées

représentent un groupe de descripteurs fortement corrélés dans les données initiales. Ainsi, la réduction de la dimension a pour effet de réduire potentiellement la quantité de bruit associée aux données sémantiques.

Algorithme

L'algorithme 5.3 décrit le principe de construction du MSUA en appliquant une LSA de rang K_A comme mécanisme de réduction de la dimension. Une étape supplémentaire a été ajoutée (ligne 8) par rapport à l'algorithme initial 5.1 succédant à l'étape d'application de la mesure du TFIDF à la matrice des fréquences des votes MFV (ligne 7). Pour les mêmes raisons invoquées ci-dessus, la méthode de fréquence $freq_{u_{W\_R}}$ a été retenue pour calculer les fréquences des votes des utilisateurs (ligne 4 de l'algorithme 5.3).

Algorithme 5.3 : FFIUF avec LSA

Données :
A : attribut,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
k : le rang de la LSA.

Résultat :
Q_A : matrice modélisant MSUA, les $|U|$ utilisateurs en ligne et K_A descripteurs latents de A en colonne.
K_A : nombre de colonnes de Q_A.

1 **début**
  `/* Construction de la matrice des fréquences des votes` $MFV_{|U|,|F_A|}$ `*/`
2 **pour chaque** *(*$u \in U$*)* **faire**
3 **pour chaque** *(*$f_d \in F_A$*)* **faire**
4 $MFV(u, f_d) = freq_{u_{W\_R}}(f_d)$
5 **fin**
6 **fin**
  `/* Application de la mesure TFIDF sur la` MVF `*/`
7 $Q_A = \text{FFIUF}(MFV)$
  `/* Application d'une SVD tronquée de rang` k `sur` Q_A `*/`
8 $Q_A = LSA_k(Q_A)$ `/* voir formule 5.13 */`
9 $K_A = k$
10 **fin**

Évaluation

La figure 5.3 illustre les performances de l'algorithme USCF-FFIUF en réduisant la dimension du MSUA par application d'une LSA de rang K_A pour les quatre attributs. Les performances sont comparées à celles obtenues sans réduction de la dimension (courbe en bleu).

En observant les courbes de la réduction de la dimension, on peut en conclure que la factorisation dégrade considérablement les performances de l'algorithme USCF-FFIUF. Nous rappelons

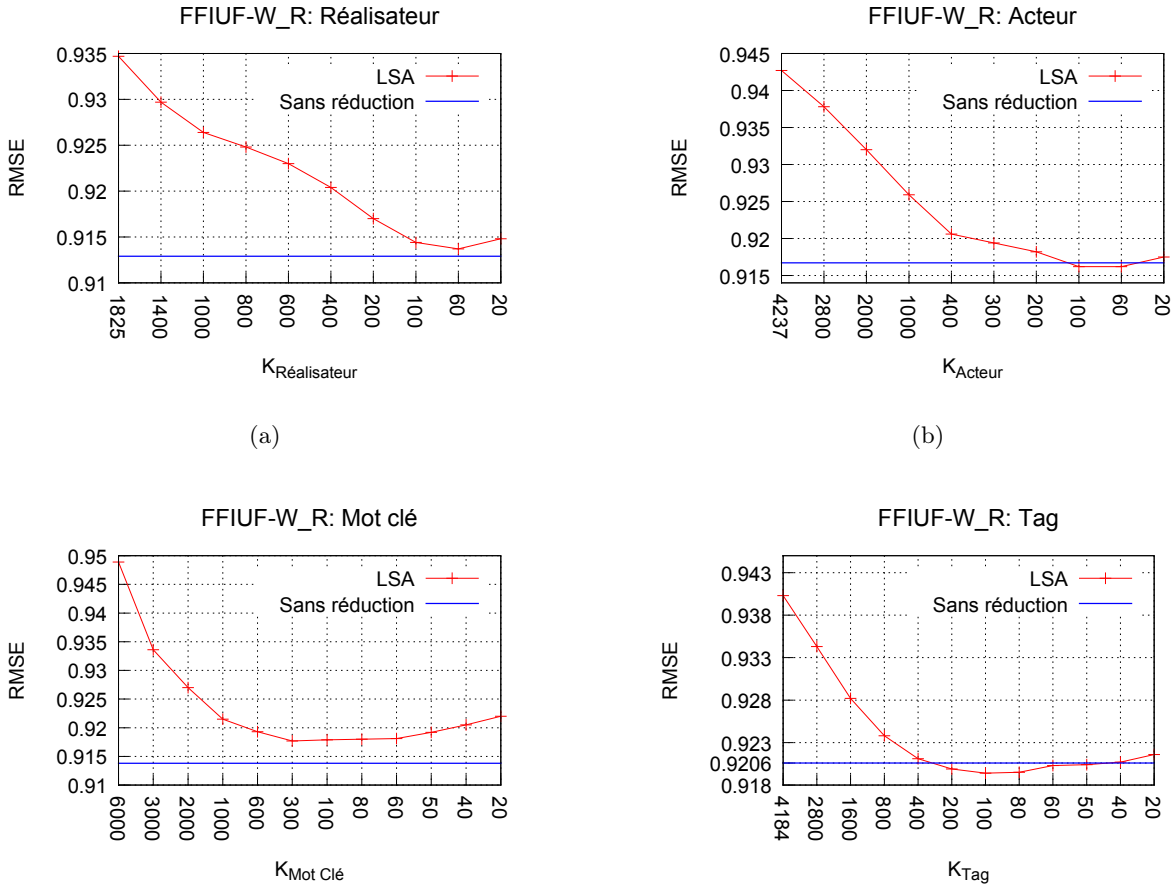


FIGURE 5.3 – USCF-FFIUF avec application d'une LSA pour réduire la dimension du MSUA

que la factorisation d'une matrice $M(l, c)$ est l'application d'une SVD, ainsi le nombre de variables latentes est égal à $K_A = min(l, c)$. En effet, pour l'attribut *Réalisateur*, le nombre initial de descripteurs est égal à 1825. La factorisation de la matrice $Q_{A_{(6552,1825)}}$ a abouti à une dégradation de la précision des recommandations qui a atteint la valeur de $RMSE = 0.9347$ pour $K_A = 1825$ contre $RMSE = 0.9129$ sans factorisation. Ce constat reste le même pour les trois autres attributs.

Par ailleurs, on constate que les courbes représentant la précision de la réduction de la dimension des quatre attributs affichent la même tendance. La réduction de la dimension augmente la précision jusqu'à atteindre une valeur seuil de K_A à partir de laquelle la précision commence à diminuer. Pour l'attribut *Tag*, par exemple (figure 5.3(d)), l'optimum est atteint pour K_{Tag} égal à 100 avec une $RMSE = 0.9194$ légèrement meilleure que celle obtenue sans la réduction de la dimension ($RMSE = 0.9206$). Pour l'attribut *Tag*, la dimension du MSUA est passée de 4184 à 100 tout en maintenant les mêmes performances en terme de précision. Ce constat est le même pour les autres attributs. Toutefois, la précision de l'attribut *Mot Clé* s'est vue légèrement réduite en passant de 0.9179 pour $K_{MotClé} = 100$ à 0.9138 sans réduction de la dimension. Le tableau 5.8 donne, pour la meilleure précision obtenue par chaque attribut, le taux de réduction

de la dimension et le gain (ou la perte) en précision par rapport à l'approche sans réduction de la dimension.

TABLE 5.8 – USCF-FFIUF avec application d'une LSA pour réduire Q_A

A	$\lvert F_A \rvert$	$RMSE_{\lvert F_A \rvert}$	K_A	$RMSE_{K_A}$	Gain RMSE	% Réduction
Réalisateur	1825	0.9130	60	0.9137	−0.0007	96.71%
Acteur	4237	0.9167	60	0.9162	+0.0005	98.58%
Mot Clé	6840	0.9138	300	0.9177	−0.0039	95.61%
Tag	4184	0.9206	100	0.9194	+0.0012	97.61%

5.1.3 Synthèse des résultats

Évaluation des deux méthodes de réduction de la dimension

Après avoir présenté les différents résultats obtenus par chacune des deux méthodes de réduction, il s'avère intéressant de se poser les questions suivantes. Quelle méthode est la plus performante ? Existe-il un comportement différent selon les attributs ? La figure 5.4 compare les meilleures performances (précisions) obtenues par les deux méthodes pour les quatre attributs en précisant pour chaque performance la dimension obtenue du MSUA.

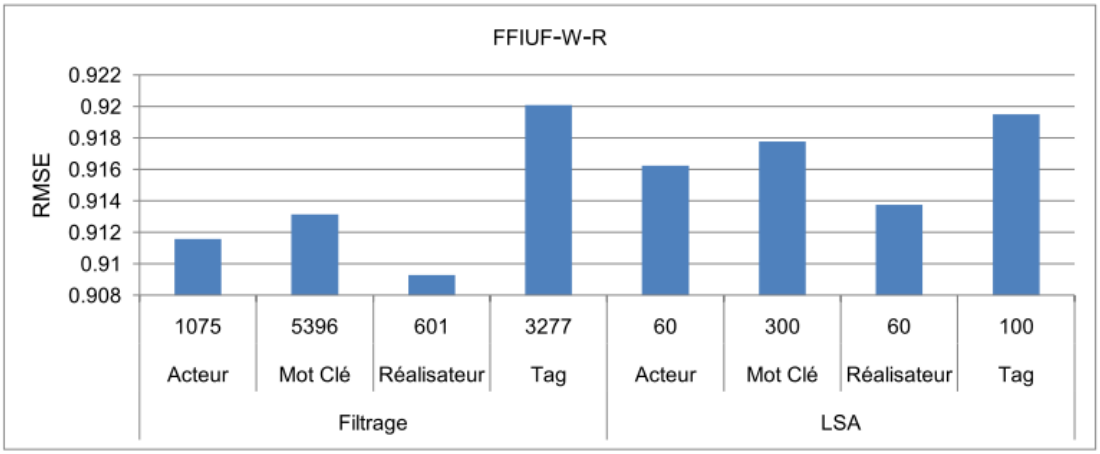


FIGURE 5.4 – Comparaison des meilleures performances de USCF-FFIUF avec réduction de la dimension.

Les attributs *Acteur* et *Réalisateur* affichent les meilleures performances en terme de précision pour les deux méthodes avec une avance pour la réduction par filtrage des descripteurs mais une réduction plus importante pour la méthode LSA. Pour les attributs *Tag* et *Mot Clé*, la réduction de la dimension par filtrage des descripteurs reste très faible (de l'ordre de 21%) comparée à celle obtenue par la LSA, qui en plus, affiche des performances très proches de celles obtenues par le filtrage par descripteur.

L'analyse de ce graphique nous permet de dire que la méthode LSA est plus performante en terme de réduction de la dimension et de précision pour les attributs *Tag* et *Mot Clé*. Toutefois, les affirmations sont moins évidentes quant aux attributs *Acteur* et *Réalisateur*.

Évaluation de la précision des prédictions de votes

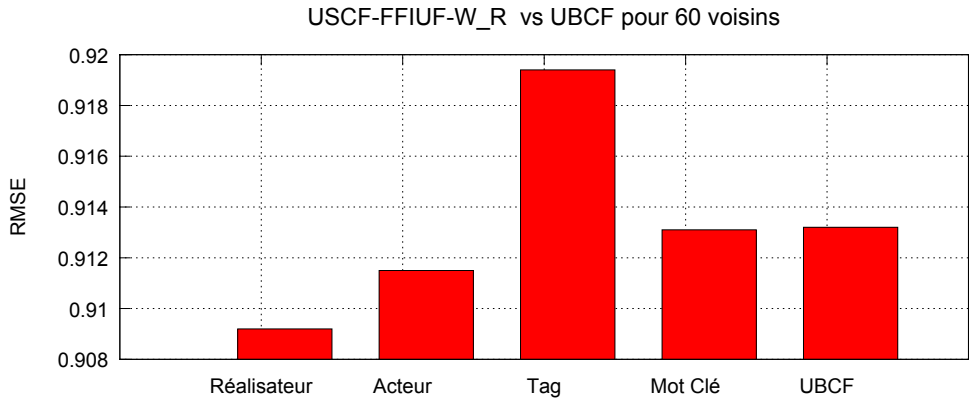


FIGURE 5.5 – Évaluation de la précision des prédictions de USCF-FFIUF par rapport à une approche collaborative

La figure 5.5 compare les performances de notre approche USCF-FFIUF appliquée aux quatre attributs par rapport à l'algorithme purement basé sur l'usage UBCF. Les performances correspondent aux meilleures précisions enregistrées par chaque attribut. L'attribut *Réalisateur* affiche les meilleures performances suivi de l'attribut *Acteur*, tous deux devançant légèrement les performances de l'algorithme de filtrage collaboratif basé sur les voisins, UBCF sur le graphique.

Évaluation de la précision d'une Top-N liste

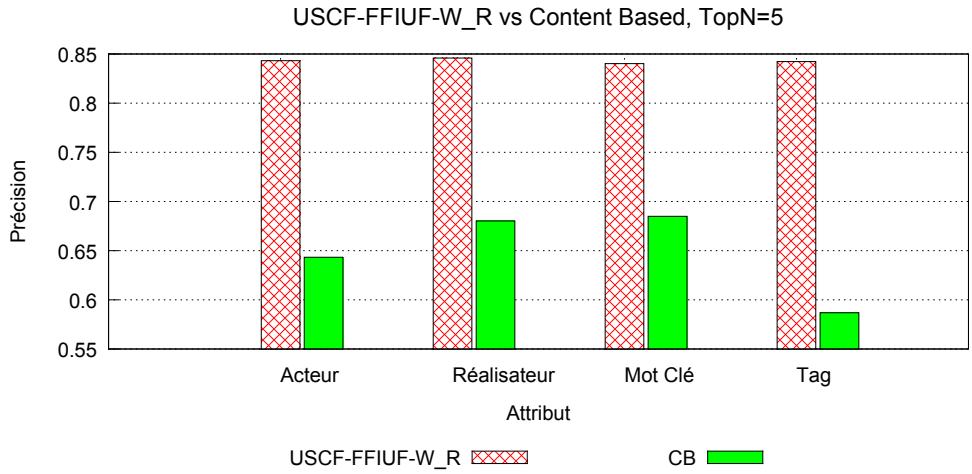


FIGURE 5.6 – Évaluation de la précision d'une Top-N liste de USCF-FFIUF vs CB

La figure 5.6 compare notre approche à une approche purement basée sur le contenu (CB sur le graphique). Nous avons utilisé les mesures permettant d'évaluer une TopN liste vu que l'algorithme basé sur le contenu ne calcule pas de prédiction de votes. Pour ce faire, nous avons utilisé la Précision (voir formule 1.5) qui estime le taux de recommandations pertinentes pour

une liste Top-N.

Le graphique représente la précision d'une Top-N liste égale à 5 pour les quatre attributs et l'algorithme basé sur le contenu (CB sur les graphique) que nous avons présenté à la section 3.7.2. On voit que notre approche est plus performante et ce pour les quatre attributs.

Évaluation de la diversité

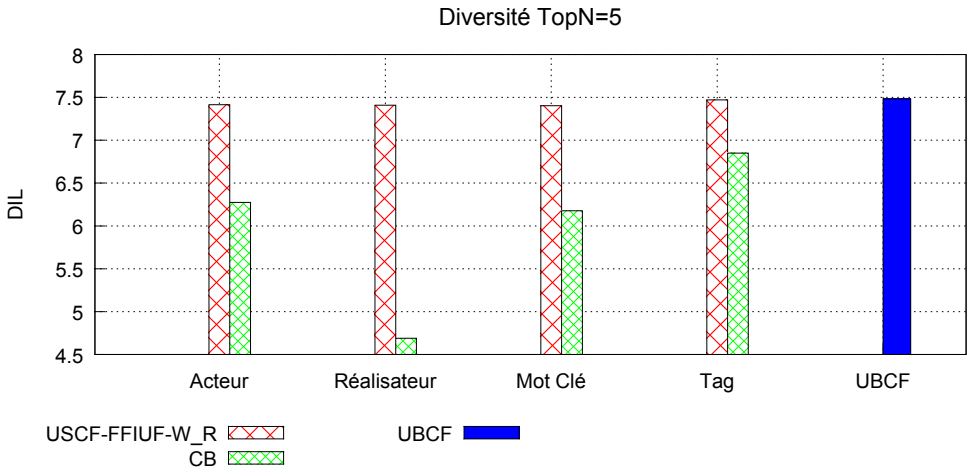


FIGURE 5.7 – Évaluation de la diversité d'une Top-N liste de USCF-FFIUF par rapport à CB et UBCF

On rappelle que la diversité des recommandations est très souvent un critère d'évaluation de la qualité d'un système de recommandation [Shani and Gunawardana, 2011]. La diversité est le point fort des algorithmes de filtrage collaboratif basés sur la mémoire et plus particulièrement ceux basés sur les utilisateurs (UBCF). Elle est généralement évaluée pour les algorithmes basés sur le contenu. Dans notre cas, nous évaluons la diversité de notre approche par rapport à l'algorithme CB et l'algorithme UBCF. On rappelle par ailleurs que la diversité a été calculée sur la base des attributs *genre*, *origine* et *réalisateur*. Nous avons utilisé la Dissimilarité Intra Liste (DIL) (voir formule 1.13) qui somme les dissimilarités entre les paires d'items de la Top-N liste par utilisateur et calcule par la suite la moyenne sur tous les utilisateurs.

La figure 5.7 compare la diversité de notre approche USCF-FFIUF par rapport à celles obtenues par les algorithmes de filtrage collaboratif UBCF et de filtrage basé sur le contenu CB pour une TopN liste égale à 5. La diversité affichée par notre approche avoisine celle obtenue par l'algorithme de filtrage collaboratif (UBCF) tout en étant largement supérieure à celle enregistrée par l'approche basée sur le contenu, particulièrement pour l'attribut *Réalisateur* qui, rappelons le, affiche également les meilleures performances en terme de précision des prédictions (voir 5.5).

5.2 Algorithme de Rocchio pour l'apprentissage du MSUA

Nous présentons dans cette section une deuxième contribution pour la construction du modèle sémantique des utilisateurs par attribut pour les attributs dépendants. L'idée est d'utiliser

l'algorithme de Rocchio [Rocchio, 1971; Buckley and Salton, 1995b] pour inférer les préférences des utilisateurs pour les descripteurs de l'attribut à partir du profil sémantique des items (MSIA) et de la matrice des votes Mv. L'algorithme de Rocchio a été utilisé dans les systèmes de recommandations basés sur le contenu [Adomavicius and Tuzhilin, 2005; Desrosiers and Karypis, 2011]. Nous l'utilisons pour construire le profil sémantique des utilisateurs par attribut.

Cette section est composée de trois parties. Dans la première partie (voir section 5.2.1), nous présentons notre approche pour la construction du MSUA en utilisant l'algorithme de Rocchio. Dans la seconde partie (voir section 5.2.2), nous nous intéressons à la réduction de la dimension du MSUA défini par le nombre de descripteurs de l'attribut correspondant. Pour ce faire nous proposons trois méthodes différentes pour la réduction de la dimension. La première réduit la dimension du MSIA en filtrant les descripteurs de l'attribut en question (voir section 5.2.2.1) pour ne conserver que les plus pertinents. La seconde (voir section 5.2.2.2) en réduisant la dimension de la matrice sémantique des items par attribut, MSIA, en lui appliquant une analyse sémantique latente (LSA). La troisième méthode (voir section 5.2.2.3) en appliquant une LSA à la matrice Q_A modélisant le MSUA. Pour chaque méthode de réduction, une évaluation de la précision des recommandations est présentée pour les quatre attributs dépendants. La troisième partie (voir section 5.2.3) évalue d'une part, les trois méthodes de réduction de la dimension et compare d'autre part, les performances de notre approche par rapport à une approche purement basée sur les usages et une approche purement basée sur le contenu. Les performances sont évaluées en terme de précision et de diversité des recommandations.

5.2.1 Apprentissage du MSUA

L'algorithme de Rocchio est utilisé pour déterminer l'importance d'un descripteur pour un utilisateur. Nous débutons cette section par une brève description du principe de l'algorithme de Rocchio (voir section 5.2.1.1). L'application de l'algorithme de Rocchio pour la construction du MSUA est présentée à la section 5.2.1.2. Nous terminons cette section en comparant les résultats obtenus par cette approche avec ceux obtenus par notre approche précédente FFIUF (voir section 5.1), et ce pour les quatre attributs dépendants.

5.2.1.1 Algorithme de Rocchio

L'algorithme de Rocchio est une procédure de retour d'expérience utilisée dans le domaine de la recherche d'information [Salton, 1989]. Il vise à optimiser la formulation d'une requête par apprentissage. Dans un système de recherche d'information utilisant une représentation en modèle vectoriel, tout document D et toute requête R peuvent être modélisés par un vecteur défini dans un espace de dimension t. $D = (d_1, d_2, \ldots, d_t)$ et $R = (r_1, r_2, \ldots, r_t)$. d_i et r_i représentent respectivement les poids du terme d'indice i dans D et R. La détermination des documents répondant à une requête R donnée se fait en mesurant la similarité entre les vecteurs modélisant l'ensemble des documents et le vecteur modélisant la requête R.

Rocchio a montré dans [Rocchio, 1971], que dans un système de recherche d'informations utilisant le cosinus comme mesure de similarité entre les documents et les requêtes, la formulation optimale pour une requête aboutissant au filtrage du plus grand nombre de documents pertinents parmi une collection de documents est définie par la formule 5.14.

$$R_{opt} = \frac{1}{|D_{pertinent}|} \sum_{D_{pertinent}} \frac{D_l}{\|D_l\|} - \frac{1}{|D_{NonPertinent}|} \sum_{D_{Nonpertinent}} \frac{D_l}{\|D_l\|} \tag{5.14}$$

Où $\|D_l\|$ est la norme Euclidienne du vecteur document D_l ; $D_{Pertinent}$ est l'ensemble des documents pertinents et $D_{NonPertinent}$ est l'ensemble des documents non pertinents.

Nous avons appliqué la formule de Rocchio (5.14) pour déterminer le profil sémantique par attribut de l'utilisateur u. Pour ce faire, nous avons remplacé les documents par les items de I, chaque item i étant décrit par un modèle vectoriel i_{SVM}. Le profil sémantique par attribut de l'utilisateur u selon Rocchio est donné par le vecteur $Rocchio(u)$ défini par la formule 5.15.

$$Rocchio(u) = \frac{1}{\left|I_{u_{pertinent}}\right|} \sum_{I_{u_{pertinent}}} \frac{i_{SVM}}{\|i_{SVM}\|} - \frac{1}{|I_{u_{NonPertinent}}|} \sum_{I_{u_{NonPertinent}}} \frac{i_{SVM}}{\|i_{SVM}\|} \tag{5.15}$$

Où $I_{u_{pertinent}}$ est l'ensemble des items pertinents pour l'utilisateur u que nous avons déjà défini par la formule (5.4). $I_{u_{NonPertinent}} = I_u \setminus I_{u_{pertinent}}$ est l'ensemble des items qui ne sont pas pertinents pour l'utilisateur u.

5.2.1.2 Construction du modèle sémantique des utilisateurs par Attribut MSUA

La construction du modèle sémantique des utilisateurs par attribut MSUA est décrite par l'algorithme 5.4. L'algorithme est constitué de deux étapes :

Application de la mesure TFIDF

La première étape de l'algorithme consiste à calculer le vecteur i_{SVM} représentant l'item i par une modélisation vectorielle SVM (lignes 2). Nous rappelons que chaque item i de I est représenté par un vecteur représentant son profil sémantique par attribut PSA_{i_A}. La représentation vectorielle i_{SVM} (ligne 4) est alors obtenue à partir du PSA_{i_A} en appliquant la mesure TFIDF. La mesure TFIDF est la même que celle définie dans l'approche FFIUF (section 5.1.1.2) mais en remplaçant la matrice des fréquences des votes MFV par la matrice sémantique des items par attribut (MSIA).

$$i_{SVM} = (i_{SVM}(f_d))_{(d=1..|F_A|)} = (TFIDF(PSA_{i_A}, f_d))_{(d=1..|F_A|)} \tag{5.16}$$

$TFIDF(PSA_{i_A}, f_d)$ donne le poids du descripteur f_d dans l'item i représenté par son profil sémantique par attribut PSA_{i_A}.

Application de la formule de Rocchio

La deuxième étape consiste à appliquer la formule de Rocchio (voir la formule (5.15)) à tous les utilisateurs u de U (ligne 7 de l'algorithme 5.4).

Algorithme 5.4 : Rocchio

Données :

A : attribut,

Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,

$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.

Résultat :

Q_A : matrice modélisant MSUA, les utilisateurs en ligne et les descripteurs de A en colonne.

K_A : nombre de colonnes de Q_A.

début
  `/* Appliquer la mesure TFIDF sur` $MSIA_A$ `*/`
  pour chaque $(i \in I)$ **faire**
    `/* Construction du modèle vectoriel de l'item` i `:`i_{SVM} `*/`
    pour chaque $(f_d \in F_A)$ **faire**
      $i_{SVM}(f_d) = TFIDF(PSA_{i_A}, f_d)$
    fin
  fin
  `/* Appliquer la formule de Rocchio */`
  pour chaque $(u \in U)$ **faire**
    $Q_A(u) = Rocchio(u)$`/* appliquer la formule` (5.15) `*/`
  fin
  $K_A = |F_A|$
fin

5.2.1.3 Évaluation

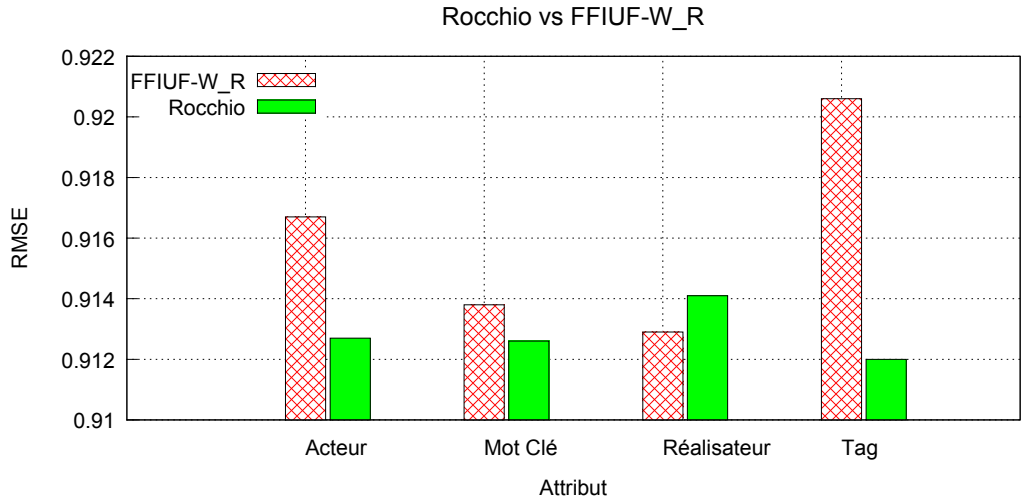


FIGURE 5.8 – Évaluation de la précision des prédictions de USCF-FFIUF vs USCF-Rocchio sans réduction de la dimension

La figure 5.8 compare la précision des prédictions obtenue par les deux méthodes FFIUF et Rocchio pour la construction du MSUA pour les quatre attributs dépendants. L'algorithme USCF-Rocchio affiche une performance supérieure à celle de l'algorithme USCF-FFIUF sauf pour l'attribut *Réalisateur* où les résultats sont assez équivalents. Ceci peut s'expliquer par le fait que Rocchio utilise aussi bien les items pertinents que les items non-pertinents pour la construction du profil sémantique des utilisateurs alors que FFIUF utilise uniquement les items pertinents.

5.2.2 Réduction de la dimension du MSUA

Comme nous l'avons déjà dit, il existe deux approches pour réduire la dimension du MSUA, la première en réduisant le nombre de colonnes de la MSIA, la seconde en réduisant la dimension de la matrice Q_A (voir section 5.2.2.3). Pour la réduction de la MSIA, nous proposons en plus de l'approche basée sur le filtrage des descripteurs (voir section 5.2.2.1), une deuxième méthode consistant à appliquer une LSA sur la MSIA (voir section 5.2.2.2). On constate par ailleurs, que les performances affichées par la méthodes Rocchio pour les quatre attributs sont assez proches avec une légère supériorité pour l'attribut *Tag*. Les résultats affichés par la méthode FFIUF sont quant à eux plus distants.

5.2.2.1 Réduction de la dimension de la MSIA par filtrage des descripteurs

Le principe de la méthode a été présenté à la section 5.1.2.1. Nous rappelons qu'il s'agit de filtrer les descripteurs de l'attribut A en ne conservant que les plus pertinents. Un descripteur est jugé pertinent s'il renferme un nombre de votes supérieur à un seuil donné μ. Une étape supplémentaire est ajoutée par rapport à l'algorithme de Rocchio 5.4, elle consiste à construire l'ensemble des descripteurs pertinents (voir ligne 3 de l'algorithme 5.5).

Algorithme 5.5 : Rocchio avec filtrage des descripteurs de A

Données :

A : attribut,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
μ : seuil déterminant les descripteurs pertinents.

Résultat :

Q_A : matrice modélisant MSUA, les $|U|$ utilisateurs en ligne et K_A descripteurs pertinents de A en colonne.
K_A : nombre de colonnes de Q_A.

```
1  début
       /* Filtrage des descripteurs de A                                  */
2      F_{A_μ} = ∅
3      pour chaque (f_d ∈ F_A) faire
           /* Construire l'ensemble F_{A_μ} des descripteurs pertinents selon la
              formule (5.12)                                              */
4          . si η(f_d) ≥ μ alors
5              ajouter f_d à F_{A_μ}
6          fin
7      fin
       /* Appliquer la mesure du TFIDF sur la MSIA_A construite à partir des
          descripteurs de F_{A_μ}                                         */
8      pour chaque (i ∈ I) faire
           /* Construction du modèle vectoriel de l'item i :i_SVM         */
9          pour chaque (f_d ∈ F_{A_μ}) faire
10             i_SVM(f_d) = TFIDF(PSA_{i_A}, f_d)
11         fin
12     fin
       /* Appliquer la formule de Rocchio                                 */
13     pour chaque (u ∈ u) faire
14         Q_A(u) = Rocchio(u)/* appliquer la formule (5.15)              */
15
16     fin
17     K_A = |F_{A_μ}|
18 fin
```

Evaluation

La figure 5.9 illustre la performance de l'algorithme USCF-Rocchio en appliquant le filtrage des descripteurs comme méthode de réduction de la dimension par rapport au même algorithme sans réduction de la dimension, et ce pour les quatre attributs dépendants *Réalisateur*, *Acteur*, *mot clé* et *Tag*. En observant les courbes de la réduction de la dimension, on constate qu'elles affichent la même tendance pour les quatre attributs, la réduction de la dimension n'augmente pas la précision des prédictions. La perte de la précision reste néanmoins faible par rapport au taux de réduction de la dimension.

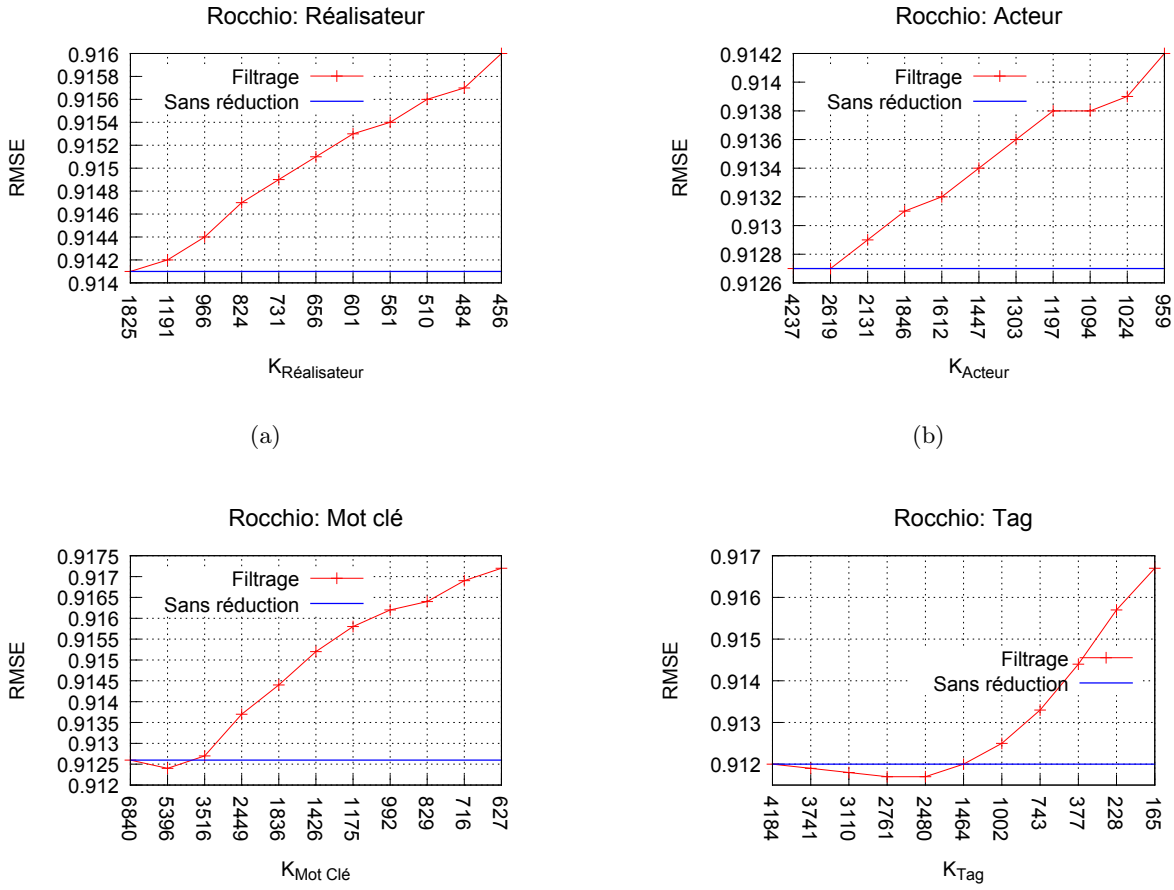


FIGURE 5.9 – USCF-Rocchio : filtrage des descripteurs pour la réduction de la dimension du MSIA

5.2.2.2 Réduction de la dimension de la MSIA par application d'une LSA

Les travaux présentés dans cette section ont fait l'objet des deux publications [Ben Ticha *et al.*, 2014; Ben Ticha *et al.*, 2015]. Il s'agit d'appliquer une Analyse Sémantique Latente (LSA) de rang k à la matrice obtenue en appliquant la mesure TFIDF sur la MSIA. On rappelle que la LSA utilise une Décomposition en Valeurs Singulières (Singular value Decomposition SVD) tronquée (voir formule 2.15), conservant uniquement les k plus grandes valeurs singulières et leurs vecteurs associés.

Algorithme

Le principe de la construction du MSUA est décrit par l'algorithme 5.6, il est constitué de trois étapes :

1. Construction de la matrice $MSIA_{TFIDF}$ en appliquant la mesure TFIDF sur la MSIA (ligne 2 de l'algorithme 5.6).

2. Application d'une LSA de rang k à la matrice $MSIA_{TFIDF}$ (ligne (3)) . La formule (5.17) donne la matrice d'approximation de rang k de la matrice $MSIA_{TFIDF}$ après lui avoir appliqué une LSA de rang k.

$$MSIA_{TFIDF} \approx LSA_k(MSIA_{TFIDF}) = B_{|I|,k} * \Sigma_{k,k} * V^t_{k,|F_A|} \tag{5.17}$$

Les lignes dans $B_{|I|,k}$ représentent les vecteurs items définis dans l'espace LSA. Les lignes dans $V_{|F_A|,k}$, représentent les vecteurs descripteurs définis dans l'espace LSA. Ainsi, chaque item est défini dans l'espace LSA par un vecteur i_{LSA} composé de k variables latentes.

3. Application de la formule de Rocchio aux items décrits par des vecteurs définis dans l'espace LSA (ligne (5.18)).

$$Rocchio(u)_{LSA} = \frac{1}{|I_{u_{pertinent}}|} \sum_{I_{u_{pertinent}}} \frac{i_{LSA}}{\|i_{LSA}\|} - \frac{1}{|I_{u_{NonPertinent}}|} \sum_{I_{u_{NonPertinent}}} \frac{i_{LSA}}{\|i_{LSA}\|} \tag{5.18}$$

Algorithme 5.6 : Rocchio en appliquant une LSA sur MSIA

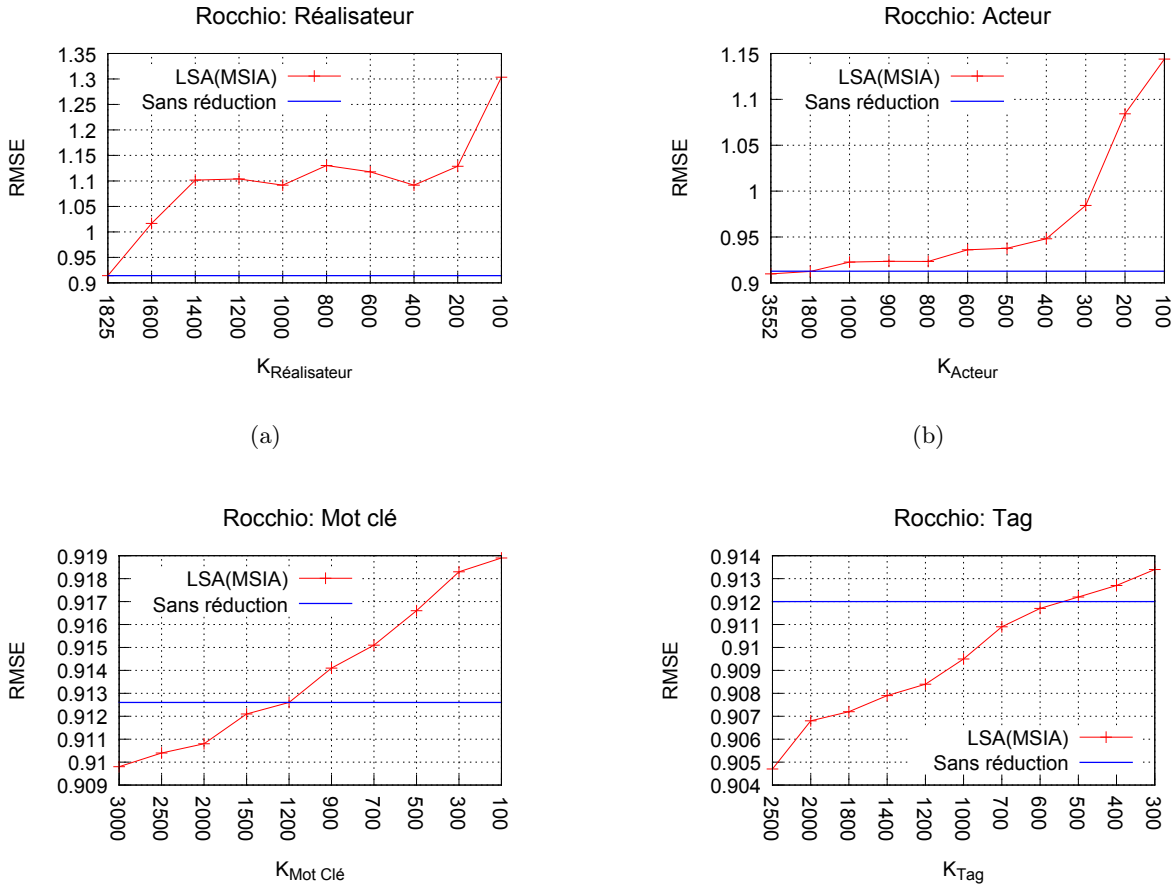
Données :
A : attribut,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
k : le rang de la LSA.

Résultat :
Q_A : matrice modélisant MSUA, les $|U|$ utilisateurs en ligne et K_A descripteurs latents de A en colonne.
K_A : nombre de colonnes de Q_A.

```
1  début
       /* Appliquer la mesure TFIDF sur la MSIA_A                          */
2      MSIA_TFIDF = (TFIDF(PSA_{i_A}, f_d))_(i = 1...|I| et d = 1..|F_A|)
       /* Appliquer une SVD tronquée de rang k sur MSIA_TFIDF              */
3      MSIA_TFIDF = LSA_k(MSIA_TFIDF)        /* appliquer la formule 5.17 */
4
       /* Appliquer la formule de Rocchio                                  */
5      pour chaque (u ∈ u) faire
6          Q_A(u) = Rocchio(u)_LSA/* appliquer la formule (5.18)          */
7
8      fin
9      K_A = k
10 fin
```

Évaluation

La figure 5.10 illustre les performances de USCF-Rocchio en fonction du rang K_A de la LSA appliquée à la matrice MSIA (courbes en rouge avec des points). On constate en premier lieu que les attributs *Réalisateur* (figure 5.10(a)) et *Acteur* (figure 5.10(b)) ont un comportement à l'opposé de celui des attributs *Mot Clé* (figure 5.10(c)) et *Tag* (figure 5.10(d)).



(a) (b)

(c) (d)

FIGURE 5.10 – USCF-Rocchio : avec application d'une LSA sur la MSIA

Attribut *Réalisateur* : la factorisation n'a aucun effet sur cet attribut. En effet, pour $K_{réalisateur}$ égal à 1825 (factorisation) la précision $RMSE = 0.9141$ est identique à celle obtenue sans réduction de la dimension $RMSE = 0.9142$. Cependant, la LSA dégrade la performance quelle que soit la valeur du rang K_A appliqué. En effet, en observant la courbe du graphique 5.10(a), on constate que la RMSE augmente considérablement lorsque le rang $K_{réalisateur}$ diminue. A titre d'exemple, pour $K_A = 1600$, la $RMSE = 1.016$ soit une perte en précision très élevée de l'ordre de 0.1018 pour un taux de réduction de l'ordre de 12.33%. On peut en conclure que la réduction de la dimension de la MSIA par application d'une LSA n'est pas adaptée à l'attribut *Réalisateur*.

Attribut *Acteur* : la tendance de la courbe est assez semblable à celle de l'attribut *Réalisateur*, toutefois il y a quelques différences. La factorisation améliore légèrement la précision de la prédiction. En effet, pour $K_{Acteur} = 3552$ (factorisation), la précision $RMSE = 0.9098$ est assez proche de celle obtenue sans réduction de la dimension $RMSE = 0.9127$. La LSA maintient les performances de l'algorithme USCF-Rocchio jusqu'à une valeur seuil de K_{Acteur} au delà de laquelle les performances en terme de précision de la prédiction se dégradent assez rapidement comme le montrent le tableau 5.9 et la figure 5.10(b).

TABLE 5.9 – USCF-Rocchio avec application d'une LSA sur MSIA (attribut Acteur)

K_{Acteur}	$RMSE_{K_Acteur}$	Gain RMSE	% Réduction
3552	0.9098	+0.0029	16.17%
2000	0.9121	+0.0006	52.8%
1800	0.9126	+0.0001	57.52%
1400	0.9137	−0.0010	66.96%
1100	0.9153	−0.0026	74.04%
900	0.9236	−0.0108	78.76%
600	0.9360	−0.0233	85.84%

Attributs *Tag* et *Mot Clé* : les courbes associées à ces deux attributs affichent la même tendance. La LSA (y compris la factorisation) améliore la précision de la prédiction par rapport à la même approche sans réduction de la dimension. A partir d'une valeur seuil du rang K_A de la LSA la précision diminue comme l'illustrent les courbes des figures 5.10(d) et 5.10(c). Toutefois, la perte en précision est très faible et négligeable par rapport au taux de la réduction de la dimension. En effet, en analysant les valeurs du tableau 5.10, on voit que pour l'attribut *Tag*, le taux de réduction de la dimension peut atteindre 90.44% pour précision $RMSE = 0.9127$ équivalente à celle obtenue sans réduction de la dimension $RMSE = 0.9120$. Le même constat s'applique à l'attribut *Mot Clé* puisqu' on garde la même précision tout en réduisant la dimension en la divisant par sept (82.46%).

TABLE 5.10 – USCF-Rocchio avec application d'une LSA sur MSIA

Attribut	K_A	$RMSE_{K_A}$	Gain RMSE	% Réduction
Tag	2593	0.9047	+0.0073	38.03%
	2000	0.9068	+0.0052	52.2%
	1200	0.9084	+0.0036	71.32%
	500	0.9122	−0.0002	88.05%
	400	0.9127	−0.0007	90.44%
Mot Clé	3544	0.9098	+0.0027	48.07%
	2000	0.9108	+0.0018	70.76%
	1200	0.9126	+0.000	82.46%
	800	0.9145	−0.019	88.3%
	500	0.9166	−0.0040	92.69%

Après analyse des résultats, on peut conclure que l'application d'une LSA pour réduire la dimension de la MSIA n'est pas adaptée à l'attribut *Réalisateur*, elle donne des résultats satisfaisants pour l'attribut *Acteur* et aboutit à de bons résultats pour les attributs *Tag* et *Mot Clé*. Par quoi peut-on expliquer ces résultats ?

Tout d'abord, on doit rappeler que les variables latentes générées par une LSA représentent un groupe de descripteurs fortement corrélés dans les données initiales. La réduction de la dimension par LSA a pour effet de réduire potentiellement la quantité de bruit associée aux données sémantiques. Ainsi, les performances de la réduction de la dimension par LSA dépendent de la corrélation existante entre les descripteurs de l'attribut, plus les descripteurs sont corrélés,

meilleure est la réduction.

En analysant les différents attributs, on constate que l'attribut *Mot clé* est un attribut dont les descripteurs sont fortement corrélés, puisqu'ils représentent des termes décrivant les items issus d'un processus d'indexation [Salton, 1989]. Les descripteurs de l'attribut *Tag* peuvent également être fortement corrélés puisqu'ils décrivent des annotions ajoutées par les utilisateurs pour décrire les items. Ce qui n'est pas le cas de l'attribut *Réalisateur* dont chaque descripteur représente le nom du réalisateur d'un film. Quant aux descripteurs de l'attribut *Acteur*, ils représentent les principaux acteurs (rang 1 et 2) des films qui peuvent être corrélés mais pas autant que les attributs *Mot clé* et *Tag*.

L'application d'une LSA pour réduire la dimension de la MSIA augmente les performances de l'algorithme USCF-Rocchio lorsque les descripteurs de l'attribut sont corrélés. C'est pour cette raison que les résultats obtenus par les attributs *Tag* et *Mot Clé* sont bien au dessus de ceux obtenus par l'attribut Acteur. La réduction par LSA ne peut pas s'appliquer à des attributs présentant des descripteurs non corrélés comme c'est le cas de l'attribut *Réalisateur*.

5.2.2.3 Réduction de la dimension de la matrice Q_A par application d'une LSA

La LSA est appliquée dans ce cas pour réduire la dimension de la matrice Q_A. Le principe de fonctionnement reste le même que celui présenté à la section 5.1.2.2 relatif à l'algorithme FFIUF. La dimension du MSUA est alors définie par le rang k de la LSA appliquée. La construction du MSUA est décrite par l'algorithme 5.7.

Algorithme 5.7 : Rocchio en appliquant une LSA sur Q_A

Données :

A : attribut,
Mv : matrice des votes, chaque colonne d'indice i correspond au profil usage PUI_i de l'item i,
$MSIA_A$: matrice sémantique des items par attribut pour l'attribut A.
k : rang de la LSA

Résultat :

Q_A : matrice modélisant MSUA, les utilisateurs en ligne et les descripteurs latents de A en colonne.
K_A : nombre de colonnes de Q_A (dimension du MSUA).

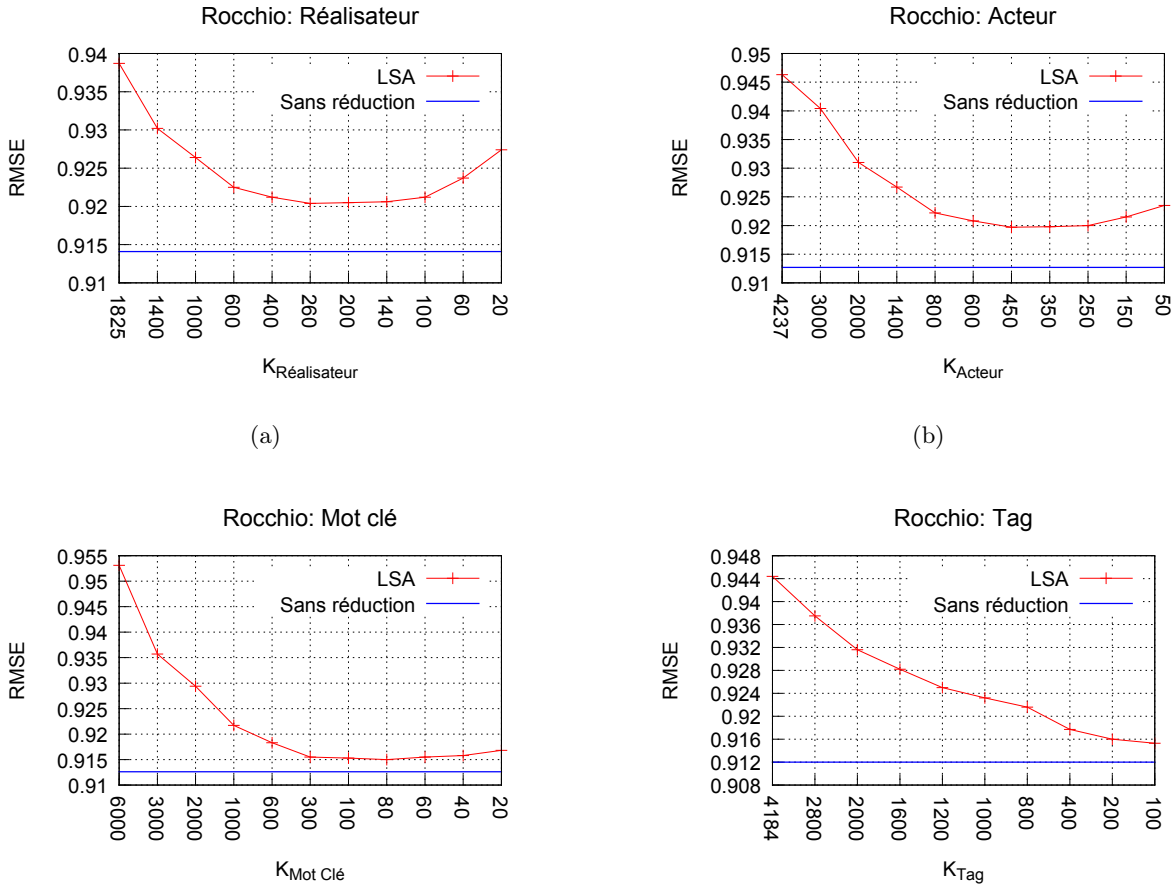
```
1  début
       /* Appliquer la mesure TFIDF sur MSIA_A                          */
2      pour chaque (i ∈ I) faire
           /* Construction du modèle vectoriel de l'item i :i_SVM       */
3          pour chaque (f_d ∈ F_A) faire
4              i_SVM(f_d) = TFIDF(PSA_{i_A}, f_d)
5          fin
6      fin
       /* Appliquer la formule de Rocchio                               */
7      pour chaque (u ∈ u) faire
8          Q_A(u) = Rocchio(u)/* appliquer la formule (5.15)            */
9
10     fin
       /* Réduction de la dimension par application d'une LSA de rang k */
11     Q_A = LSA_k(Q_A)                  /* voir formule 5.13 */
12     K_A = k
13 fin
```

Évaluation

La figure 5.11 illustre les performances de l'algorithme USCF-Rocchio en réduisant la dimension du MSUA par application d'une LSA de rang K_A pour les quatre attributs. Les performances sont comparées à celles obtenues sans réduction de la dimension (courbe en bleu).

Les résultats obtenus sont semblables à ceux observés sur l'algorithme FFIUF (voir figure 5.3). La factorisation dégrade les performances de l'algorithme USCF-Rocchio. En effet, pour l'attribut *Réalisateur*, le nombre initial de descripteurs est égal à 1825, la factorisation de la matrice $Q_{A_{(6552,1825)}}$ a abouti à une dégradation de la précision des recommandations qui a atteint la valeur de $RMSE = 0.9347$ pour $K_A = 1825$ contre $RMSE = 0.9129$ sans factorisation. Ce constat reste le même pour les trois autres attributs.

Par ailleurs, on constate que les courbes représentant la précision des prédictions après application d'une LSA affichent la même tendance pour les quatre attributs. La LSA augmente la précision jusqu'à ce que son rang atteigne une valeur seuil de K_A à partir de laquelle la précision commence à diminuer. Pour les quatre attributs, la LSA n'améliore pas la précision des recommandations. Le tableau 5.11, donne pour la meilleure précision obtenue par chaque attribut, le



(a) (b)

(c) (d)

FIGURE 5.11 – USCF-Rocchio : avec application d'une LSA sur le MSUA

taux de réduction de la dimension et le gain (ou la perte) en précision par rapport à l'approche sans réduction de la dimension. Il est à remarquer que la meilleure performance est obtenue par la plus petite dimension et ce pour les quatre attributs. En analysant les valeurs du tableau, on remarque que la perte en précision enregistrée par les attributs *Tag* et *Mot Clé* est moins élevée que celle enregistrée par les attributs *Acteur* et *Réalisateur*. Toutefois la dégradation des performances n'est pas aussi importante que celle obtenue pour l'application d'une LSA à la MSIA sur ces mêmes attributs (voir figure 5.10).

TABLE 5.11 – RMSE de USCF-Rocchio avec application d'une LSA pour réduire Q_A

A	$\|F_A\|$	$RMSE_{\|F_A\|}$	K_A	$RMSE_{K_A}$	Gain RMSE	% Réduction
Réalisateur	1825	0.9141	240	0.9202	−0.006	86.85%
Acteur	4237	0.9167	400	0.9195	−0.0069	90.56%
Mot Clé	6840	0.9138	200	0.9148	−0.0023	98.38%
Tag	4184	0.9206	100	0.9153	-0.0033	97.61%

5.2.3 Synthèse des résultats

Évaluation des méthodes de réduction de la dimension

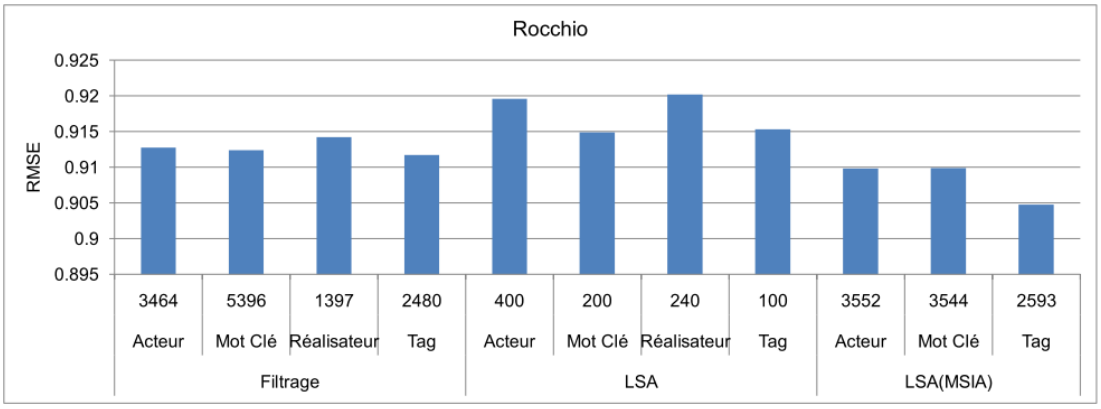


FIGURE 5.12 – Comparaison des meilleures performances de USCF-Rocchio avec réduction de la dimension.

La figure 5.12 compare les meilleures performances (précisions) obtenues par les trois méthodes de réduction de la dimension pour les quatre attributs en précisant pour chaque performance la dimension obtenue du MSUA. Pour la méthode de réduction par filtrage des descripteurs (Filtrage sur le graphique), les meilleures performances sont obtenues pour une réduction de dimension très faible. L'application d'une LSA à la matrice MSIA (LSA(MSIA) sur le graphique) donne les meilleurs résultats avec un faible taux de réduction. L'application d'une LSA à la matrice Q_A aboutit à un taux de réduction élevé avec une légère dégradation des performances.

Pour avoir une meilleure idée sur l'impact des différentes méthodes de réduction de la dimension, la figure 5.13 compare les performances des deux approches USCF-FFIUF et USCF-Rocchio avec réduction de la dimension appliquées aux quatre attributs pour une dimension égale à 500. En observant ce graphique on constate que :

— les attributs *Acteur* et *Réalisateur* ont le même comportement vis à vis des différentes méthodes de réduction. L'application d'une LSA pour réduire la dimension de la MSIA dégrade considérablement la précision des prédictions, beaucoup plus pour l'attribut *Réalisateur* (RMSE=1.08) que pour l'attribut Acteur (RMSE=0.935). Comme nous l'avons dit ci-dessus, ceci est dû au fait que les descripteurs ne sont pas ou peu corrélés. Les meilleures performances sont enregistrées pour la méthode de réduction par filtrage des descripteurs appliquée à l'algorithme FFIUF. Pour la méthode LSA, les performances sont assez proches pour les deux algorithmes Rocchio et FFIUF. Par ailleurs, pour les deux attributs, les performances de l'algorithme FFIUF sont supérieures à celles affichées par l'algorithme Rocchio pour la construction du MSUA. Ceci peut s'expliquer par le principe même de cet algorithme qui construit le profil sémantique des utilisateurs en se basant sur le nombre de votes associés à chaque descripteur de l'attribut. L'attribut *Réalisateur* est un attribut qui peut jouer un rôle dans la détermination des préférences de l'utilisateur, même chose pour les acteurs principaux d'un film. Ce qui n'est pas le cas de l'attribut *Mot Clé* dont les descripteurs sont le résultat d'une indexation du synopsis du film.

— les attributs *Tag* et *Mot Clé* affichent également le même comportement vis à vis des deux algorithmes de construction du MSUA. Pour ces attributs, l'algorithme Rocchio affiche

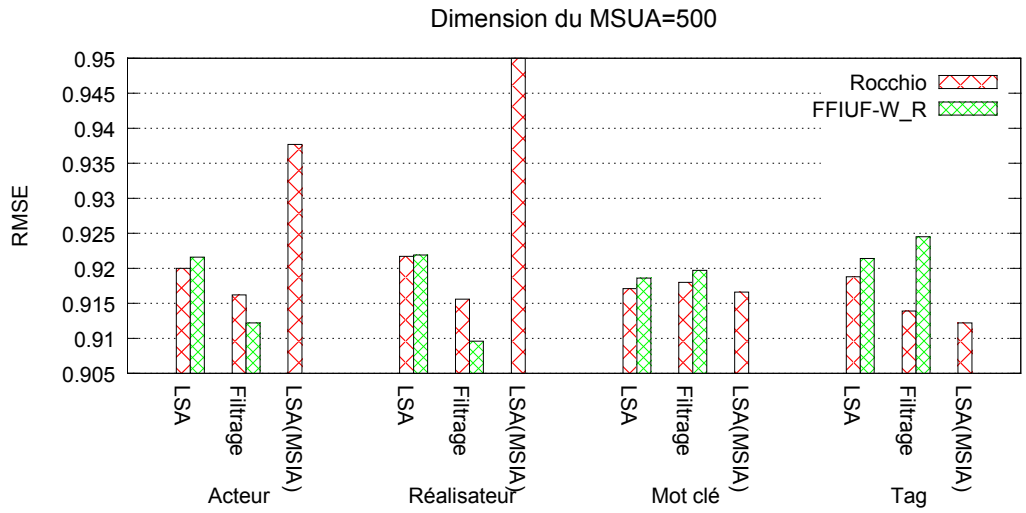


FIGURE 5.13 – Comparaison des performances de USCF-Rocchio vs USCF-FFIUF pour une dimension de MSUA égale à 500

des performances supérieures à celles enregistrées par l'algorithme FFIUF. La méthode de réduction de la dimension LSA(MSIA) affiche les meilleures performances, ceci peut s'expliquer par le fait que les descripteurs de ces attributs sont fortement corrélés.

Évaluation de la précision des prédictions de votes

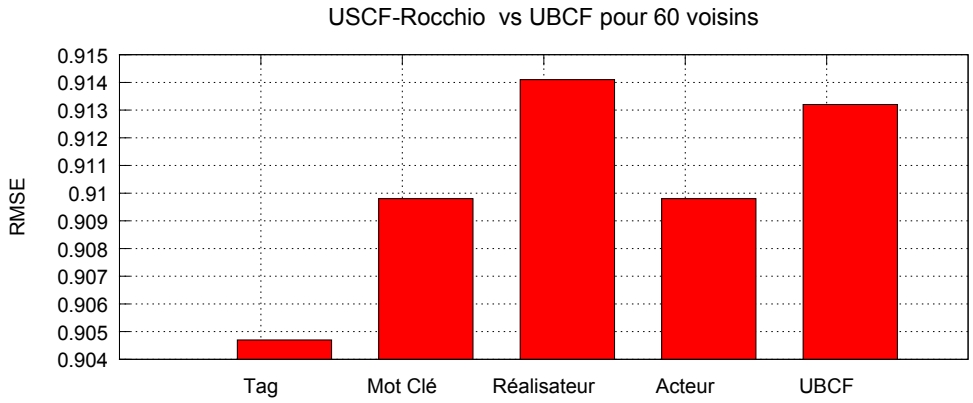


FIGURE 5.14 – Évaluation de la précision des pérdictions de USCF-Rocchio par rapport à une approche collaborative

La figure 5.14 compare les performances de notre approche USCF-Rocchio appliquée aux quatre attributs par rapport à l'algorithme de filtrage collaboratif UBCF. Les performances correspondent aux meilleures précisions enregistrées par chaque attribut. L'attribut *Tag* affiche les meilleures performances suivi des attributs *Acteur* et *Mot Clé*, tous les trois devançant les performances de l'algorithme UBCF.

Évaluation de la précision d'une Top-N liste

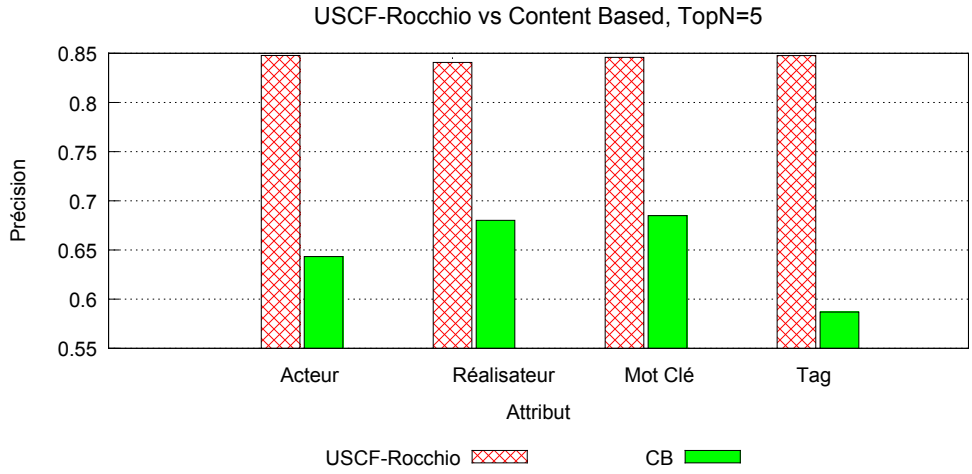


FIGURE 5.15 – Évaluation de la précision d'une Top-N liste de USCF-Rocchio vs CB

La figure 5.15 compare l'algorithme de recommandation hybride USCF-Rocchio à une approche purement basée sur le contenu (CB sur le graphique). Comme nous l'avons déjà dit, nous avons évalué les deux approches en calculant la Précision (voir formule 1.5) d'une liste Top-N.

Le graphique représente la précision d'une Top-N liste égale à 5 pour les quatre attributs et l'algorithme basé sur le contenu (CB sur les graphiques) que nous avons présenté à la section 3.7.2. Pour les quatre attributs, la précision de la Top5 liste de notre approche est largement supérieure à celle obtenue par une approche purement basée sur le contenu.

Évaluation de la diversité

La figure 5.16 compare la diversité de notre approche USCF-Rocchio par rapport à celles obtenues par les algorithmes de filtrage collaboratif UBCF et de filtrage basé sur le contenu CB pour une TopN liste égale à 5. Nous rappelons que la diversité a été mesurée en calculant la DIL (Dissimilarité Intra Liste) (voir formule 1.13) des items de la liste représentés par les trois attributs *genre*, *origine* et *réalisateur*. La diversité affichée par l'algorithme de recommandation USCF-Rocchio avoisine celle obtenue par l'algorithme de filtrage collaboratif (UBCF) tout en étant largement supérieure à celle enregistrée par l'approche basée sur le contenu (CB). L'attribut *Tag* affiche la meilleure diversité qui, rappelons le, affiche également les meilleures performances en terme de précision des prédictions comme le montre la figure 5.14.

5.3 Conclusion

Nous avons présenté dans ce chapitre deux approches, FFIUF et Rocchio, pour la construction du modèle sémantique des utilisateurs par attribut MSUA pour les attributs dépendants. Les deux approches sont basées sur les travaux issus des domaines de recherche et filtrage d'informations. Vu que les attributs dépendants peuvent avoir un nombre élevé de descripteurs, nous avons également proposé deux méthodes de réduction de la dimension. Une première méthode qui consiste à filtrer les descripteurs d'un attribut en ne conservant que les plus pertinents,

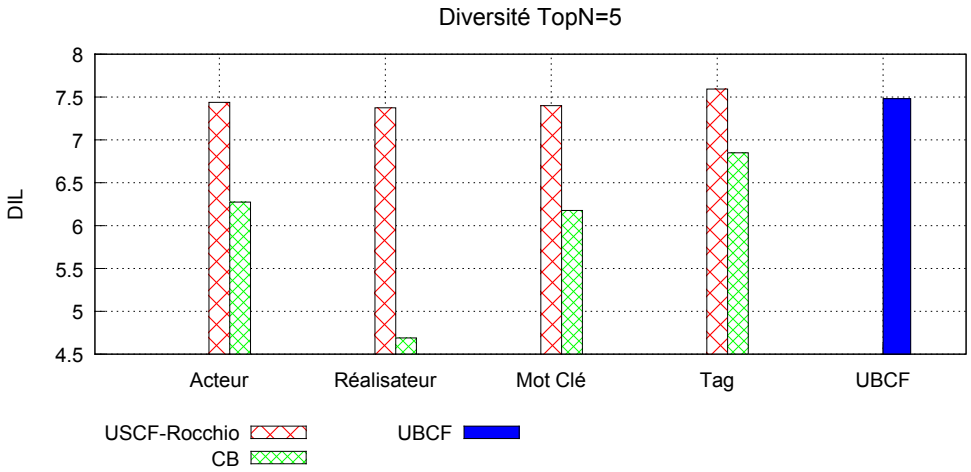


FIGURE 5.16 – Évaluation de la diversité d'une Top-N liste USCF-Rocchio par rapport à CB et UBCF

c'est-à-dire ceux ayant un nombre de votes au dessus d'un seuil donné. Une deuxième méthode issue des méthodes de factorisation des matrices, l'analyse sémantique latente (LSA). La LSA consiste à appliquer une décomposition en valeurs singulières (SVD) tronquée de rang K_A sur la matrice concernée par la réduction de la dimension. Deux algorithmes différents ont été proposés, le premier appliquant une LSA pour réduire la dimension de la matrice sémantique des items par attribut (MSIA), le second appliquant une LSA sur la matrice Q_A modélisant le MSUA.

Nous avons expérimenté nos deux approches avec et sans réduction de la dimension sur le jeu de données MovieLens 1M sur les attributs *Acteur*, *Réalisateur*, *Mot Clé* et *Tag*. Nous avons évalué les deux approches avec les différentes méthodes de réduction de la dimension en terme de précision des prédictions. Les résultats des deux approches sont différents selon les attributs. L'algorithme FFIUF avec filtrage des descripteurs est plus performant sur les attributs *Acteur* et *Réalisateur* que sur les attributs *Mot Clé* et *Tag*. L'algorithme Rocchio avec réduction de la dimension de la MSIA par application d'une LSA est plus performant pour les attributs *Tag* et *Mot Clé*. La réduction de la MSIA par une LSA est plus performante lorsque les descripteurs de l'attribut sont corrélés.

Nous avons également évalué la précision de nos deux approches par rapport à une approche purement collaborative et une approche basée sur le contenu. Nos deux approches hybrides, USCF-Rocchio et USCF-FFIUF, affichent une supériorité au niveau de la précision des recommandations par rapport aux approches non hybrides.

La diversité est un des points faibles des systèmes de recommandation basés sur le contenu (CB) en raison du principe de fonctionnement même de ces algorithmes. Cependant, les algorithmes de filtrage collaboratif, et particulièrement ceux basés sur les utilisateurs (UBCF), offrent une large diversité dans les recommandations qu'ils proposent. Nous avons comparé notre approche à UBCF et CB en terme de la diversité des recommandations. Les résultats obtenus sont largement supérieurs à ceux obtenus par un algorithme CB, et très proches des performances de l'algorithme UBCF.

Nous avons considéré dans ce chapitre ainsi que dans le chapitre précédent un seul attribut pour la construction du profil sémantique des utilisateurs, c'est-à-dire que le modèle sémantique des utilisateurs (MSU) a été confondu avec le MSUA. Or dans notre approche, USCF, le MSU est construit à partir d'un ensemble d'attributs pertinents. Pour chaque attribut est construit le MSUA correspondant en utilisant l'approche adaptée. Le modèle sémantique des utilisateurs est par la suite construit à partir des différents MSUA selon le principe que nous avons présenté à la section 3.4.

Chapitre 6

Synthèse des résultats, conclusion et perspectives

Nous clôturons ce manuscrit par une synthèse des résultats que nous avons obtenus (section 6.1). Un résumé de nos différentes contributions est présenté à la section 6.2. Enfin, la section 6.3 identifie les perspectives possibles de notre travail.

6.1 Synthèse des résultats

La section 6.1.1 compare les performances enregistrées par les différentes approches que nous avons proposées pour l'apprentissage du Modèle Sémantique des Utilisateurs par Attribut (MSUA) dans les chapitres 4 et 5. La recommandation à partir de la combinaison de différents attributs, c'est-à-dire l'apprentissage du Modèle Sémantique des Utilisateurs (MSU) à partir des différents MSUA fait l'objet de la section 6.1.2. Le comportement de notre approche USCF, et plus particulièrement le profil sémantique des utilisateurs, vis-à-vis de l'arrivée, en temps réel, de nouveaux votes est présenté et discuté à la section 6.1.3. Enfin, la section 6.1.4 compare les performances de notre approche, USCF, à celles obtenues par une approche purement collaborative et une approche purement basée sur le contenu.

6.1.1 Comparaison des différentes méthodes d'apprentissage du MSUA

Cette section analyse les meilleures performances obtenues par les différentes méthodes d'apprentissage du MSUA pour les items structurés. La figure 6.1 compare les meilleures performances enregistrées par les différentes méthodes de construction du MSUA sur les attributs indépendants (*Genre* et *Origine*) et les attributs dépendants (*Acteur*, *Réalisateur* et *Tag*). La performance est évaluée en mesurant la précision des prédictions de votes (RMSE) pour 60 voisins. Nous rappelons ci-après la signification des différentes méthodes présentées sur le graphique :

Moyenne : le $MSUA\_A$ est obtenu en calculant la moyenne des votes de chaque utilisateur sur les descripteurs de l'attribut A (voir page 74). Nous avons utilisé cette méthode comme approche hybride par rapport à laquelle nous avons comparé nos différentes méthodes.

***FuzzyCM* :** s'applique aux attributs indépendants multi-valués (tels que l'attribut *Genre*), elle utilise l'algorithme de clustering flou, Fuzzy C-Means, pour l'apprentissage du MSUA (voir section 4.3).

***KMeans* :** s'applique aux attributs indépendants monovalués (tels que l'attribut *Origine*), elle utilise l'algorithme de clustering non hiérarchique K-Means pour l'apprentissage du MSUA (voir section 4.4).

***FFIUF-WR* (Feature Frequency Inverse User Frequency) :** construit le MSUA pour les attributs dépendants à partir de la fréquence des votes des utilisateurs sur les descripteurs de l'attribut en appliquant la mesure TFIDF (Term Frequency Inverse Document Frequency) (voir section 5.1).

***Rocchio* :** construit le MSUA pour les attributs dépendants en appliquant l'algorithme de Rocchio (voir section 5.2).

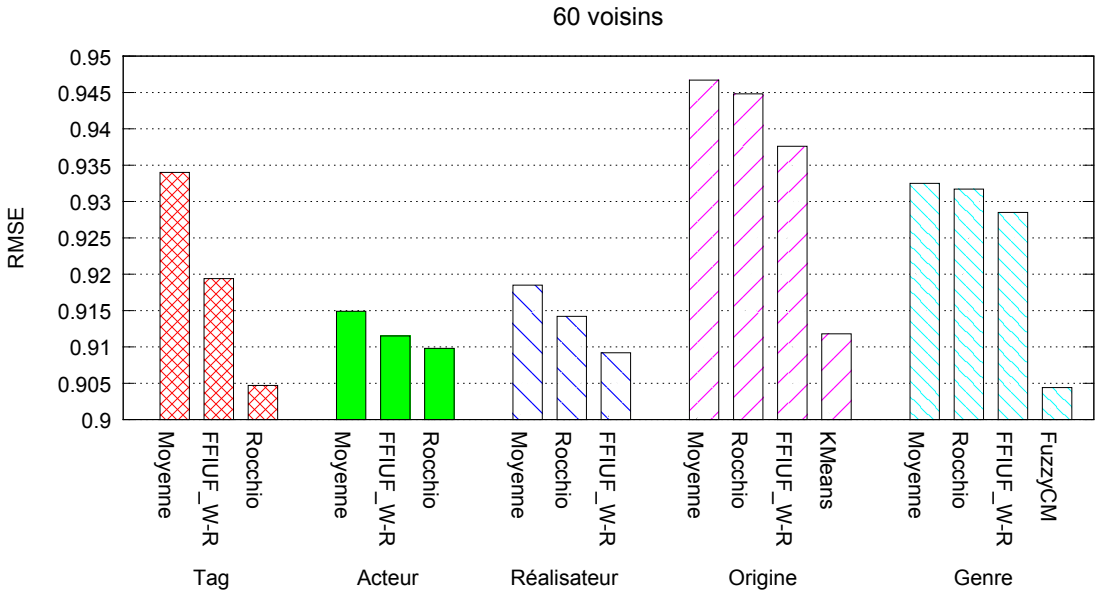


FIGURE 6.1 – Comparaison des méthodes d'apprentissages du MSUA pour les items structurés

En analysant ce graphique on peut en déduire les constatations suivantes :

Performance de l'algorithme Moyenne : le premier constat que nous pouvons faire en observant ce graphique est que les recommandations fournies par un MSUA construit à partir de la méthode *Moyenne* sont les moins pertinentes, et ce pour les différents attributs expérimentés. Par ailleurs, les précisions enregistrées par la méthode *Moyenne* sont largement en dessous de celles obtenues par un algorithme de filtrage collaboratif basé sur les utilisateurs (UBCF), et ce pour tous les attributs. En effet, la RMSE de UBCF pour 60 voisins est égale à 0.9132 alors qu'elle varie entre 0.9467 (*Origine*) et 0.9150 (*Acteur*) en passant par 0.9325 (*Genre*), 0.9340 (*Tag*) et 0.9185 (*Réalisateur*). Bien que la méthode *Moyenne* soit une approche hybride qui introduit l'aspect sémantique des items dans une approche collaborative, les recommandations qu'elle produit ne sont pas aussi pertinentes que celles obtenues par une approche purement collaborative. Ainsi, la prise en compte de l'aspect sémantique dans une approche collaborative ne suffit pas à lui seul à "booster" la précision des prédictions des votes.

Impact de la classification des attributs : nous avons appliqué les méthodes FFIUF-WR et Rocchio aux attributs indépendants, ici *Genre* et *Origine*. Nous rappelons que ces méthodes sont destinées à construire le MSUA des attributs dépendants. En observant les performances de ces algorithmes sur les attributs indépendants, on remarque aisément que la pertinence des recommandations enregistre une perte importante par rapport aux méthodes d'apprentissage destinées aux attributs indépendants. En effet, pour l'attribut *Genre*, la RMSE observée pour la méthode FuzzyCM est égale à 0.9044 alors qu'elle est de 0.9285 pour la méthode FFIUF-WR et de 0.9317 pour la méthode Rocchio, soit une différence de plus de 2 points dans le meilleur des cas. Le même constat peut être fait pour l'attribut *Origine* qui enregistre une RMSE de 0.9118 pour la méthode KMeans, alors qu'elle est de 0.9376 pour la méthode FFIUF-WR et de 0.9448 pour la méthode Rocchio. Par ailleurs, la pertinence des recommandations obtenues par les méthodes FFIUF-WR et Rocchio appliquées aux attributs indépendants *Genre* et *Origine* est largement en dessous de celle enregistrée par les trois attributs dépendants *Acteur*, *Réalisateur* et *Tag*. Les méthodes FFIUF-WR et Rocchio donnent de meilleurs résultats lorsque les attributs sont dépendants, c'est-à-dire disposant d'un nombre élevé de descripteurs, ce qui peut expliquer les mauvaises performances enregistrées par les attributs indépendants. On voit ainsi, l'intérêt d'appliquer des méthodes différentes pour l'apprentissage de MSUA en fonction de la classe d'appartenance de l'attribut correspondant.

Pertinence des attributs pour l'utilisateur : en observant de plus près ce graphique, nous constatons également que l'attribut *Origine* enregistre les plus mauvaises performances comparées aux autres attributs, et ce pour toutes les méthodes d'apprentissage du MSUA. Ceci peut s'expliquer par le fait que l'attribut *Origine* n'est pas aussi prédictif que les autres attributs. C'est-à-dire, qu'il représente, comparé aux autres attributs, le critère le moins important pris en considération par l'utilisateur lorsqu'il effectue son choix, ce qui est intuitivement compréhensible. Ceci confirme l'hypothèse que nous avons émise, à savoir que tous les attributs n'ont pas la même importance auprès des utilisateurs.

6.1.2 Modèle Sémantique des Utilisateurs à partir de la combinaison de plusieurs attributs

Nous rappelons que dans notre approche, le MSU est associé à un groupe d'attributs. Pour chaque attribut A du groupe, un $MSUA_A$ est construit en appliquant la méthode adéquate. Le MSU est alors construit par concaténation des différents $MSUA$ des attributs associés.

La figure 6.2 représente la précision des recommandations pour différents modèles sémantiques des utilisateurs (MSU). Chaque histogramme évalue la pertinence des recommandations correspondant à un MSU construit à partir de la concaténation de plusieurs MSUA. Les données sur l'axe des abscisses représentent les attributs ayant contribué à la construction du MSU. A titre d'exemple, la combinaison *Acteur-Réalisateur* signifie que le MSU est construit en concaténant le $MSUA_{Acteur}$ et $MSUA_{Réalisateur}$. Par soucis de lisibilité, nous ne précisons pas sur le graphique les noms des méthodes utilisées pour l'apprentissage du MSUA pour les attributs. Toutefois, pour chaque attribut A, la méthode associée à sa classe d'appartenance est utilisée, de même que le même $MSUA_A$ est utilisé dans toutes les expérimentations.

Nous avons représenté dans ce graphique trois groupes d'histogrammes. Le groupe du milieu représente les précisions des prédictions de votes obtenues pour un MSU construit à partir d'un seul attribut. Les deux autres groupes représentent les précisions des prédictions obtenues pour un

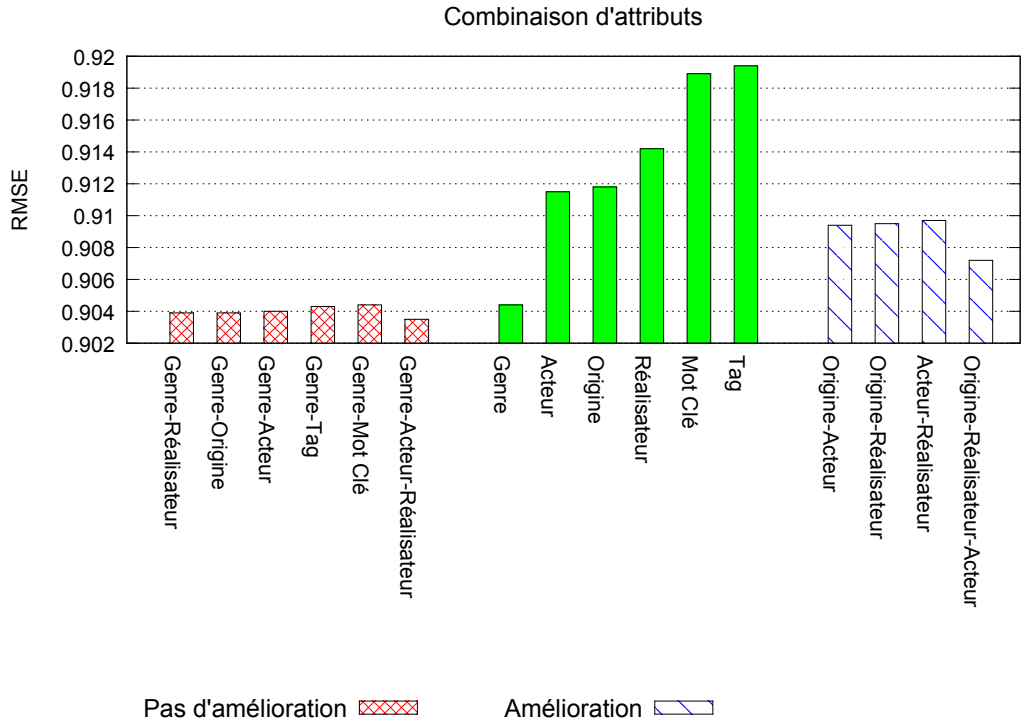


FIGURE 6.2 – Évaluation de la précision des prédictions de votes par combinaison d'attributs

MSU construit à partir de la combinaison de deux ou trois attributs. En analysant ce graphique, on constate que la combinaison d'attributs peut améliorer, dans certains cas, la pertinence des recommandations, c'est le cas des combinaisons figurant dans le groupe de droite que nous avons étiqueté "*amélioration*". Elle peut également n'apporter aucune amélioration par rapport à un des attributs, c'est le cas des combinaisons du groupe de gauche que nous avons étiqueté "*pas d'amélioration*". La précision observée par chacune des combinaisons du groupe "*amélioration*" est supérieure à celle enregistrée par chacun des attributs la composant. Ainsi, la précision observée par la combinaison *Origine-Acteur-Réalisateur* est supérieure à celle enregistrée par chacun des attributs *Origine*, *Réalisateur* et *Acteur*. C'est le cas également des combinaisons *Acteur-Réalisateur*, *Origine-Acteur* et *Origine-Réalisateur*. Bien que l'attribut *Origine* ait enregistré les plus mauvaises performances, le fait de le combiner à d'autres attributs a permis d'améliorer la pertinence des recommandations.

Le groupe "*pas d'amélioration*" représente les combinaisons des différents attributs avec l'attribut *Genre*. En analysant ce groupe, on constate qu'il n'y a pas eu gain en précision par rapport à l'attribut *Genre*, et ce pour toutes les combinaisons de ce groupe. En effet, la RMSE obtenue par l'attribut *Genre* est égale à 0.9044 alors qu'elle varie entre 0.9035 et 0.9040 pour toutes les combinaisons d'attributs de ce groupe. Ceci peut s'expliquer par le fait que l'attribut *Genre* a enregistré les meilleures performances, et le fait de le combiner avec tout autre attribut n'améliorera pas la pertinence des recommandations. Toutefois, un gain assez important est enregistré par rapport aux autres attributs. A titre d'exemple, la RMSE de la combinaison *Genre-Réalisateur* est égale à 0.9040 alors qu'elle est égale à 0.9142 pour l'attribut *Réalisateur* soit un gain d'un point.

La question qui se pose alors est : comment déterminer la ou les meilleures combinaisons d'attributs ? En poussant les expérimentations, nous avons constaté que la concaténation de deux MSUA correspondant à deux attributs différents ayant donné chacun une précision optimale, ne donne pas forcément la meilleure précision. Il est peut être intéressant d'étudier d'autres facteurs tels que la corrélation entre les attributs pour la détermination de la meilleure combinaison.

6.1.3 Adaptation du profil des utilisateurs

Les préférences de l'utilisateur évoluent et changent avec le temps. Nous avons présenté à la section 1.1.5 (page 20) quelques techniques de mise à jour du profil utilisateur. Parmi ces techniques, l'*adaptation par ajout d'information* est la plus utilisée dans les systèmes de recommandation [Montaner *et al.*, 2003]. L'adaptation du profil utilisateur est assurée, dans ce cas, par les nouvelles observations effectuées par l'utilisateur, et recueillies par le système par analyse des usages sous forme d'évaluations explicites ou implicites. Une nouvelle observation peut représenter, en fait, un retour d'expérience sur une recommandation proposée par le système. Étant donné notre hypothèse de départ qui stipule qu'aucune information n'est disponible sur les utilisateurs mis à part les évaluations des items qu'ils ont consultés, nous avons adopté cette technique pour la mise à jour du profil usage des utilisateurs. La mise à jour de ce profil consiste alors à actualiser la matrice des votes M_v en y intégrant les récentes évaluations de tous les utilisateurs, cette opération se fait en temps réel.

Nous rappelons que le profil sémantique des utilisateurs, représenté par le MSU, est utilisé pour calculer les similarités entre les différents utilisateurs du système. Pour chaque utilisateur, seules les similarités avec ses N plus proches voisins sont conservées. Elles sont par la suite utilisées avec la matrice des votes Mv lors de l'étape de recommandation pour calculer la prédiction des votes. La détermination du profil sémantique (MSU) ainsi que le calcul des similarités se fait en mode offline. L'étape de recommandation se fait, quant à elle, en temps réel, c'est-à-dire lors des différentes interactions entre l'utilisateur courant et le e-service.

Dans un algorithme de filtrage collaboratif classique basé sur les utilisateurs (UBCF), les nouvelles évaluations sont prises en compte en temps réel aussi bien pour déterminer les plus proches voisins que pour établir les recommandations. Dans notre approche, USCF, l'intégration des nouvelles évaluations ne se fait qu'au niveau de l'étape de recommandation, puisque la similarité entre les utilisateurs est calculée en mode offline. A travers cette expérimentation, nous cherchons à étudier le comportement de notre approche lors de l'intégration en temps réel de nouveaux votes dans le profil usage des utilisateurs.

L'expérimentation consiste à partager le jeu de données initial en deux jeux. Le premier jeu, que nous appellerons Jeu_{MSU}, est utilisé pour l'apprentissage du MSU. Le second que nous appellerons, $Jeu_{Nouveaux\ votes}$, est utilisé pour injecter de nouvelles évaluations et mettre à jour la matrice des votes Mv, c'est-à-dire le profil usage des utilisateurs. Les votes du $Jeu_{Nouveaux\ votes}$ sont plus récents que ceux composant le Jeu_{MSU}. Le jeu Jeu_{MSU} est horodaté et décomposé en jeu pour l'apprentissage et jeu pour le test selon la proportion 80/20, le jeu apprentissage est utilisé pour construire le MSU et le jeu test pour l'évaluation de la prédiction des votes.

La figure 6.3 représente les résultats de trois découpages différents du jeu initial, le graphique 6.3(a) concerne un découpage 70/30, c'est-à-dire que $Jeu_{Nouveaux\ votes}$ contient 30% et Jeu_{MSU}

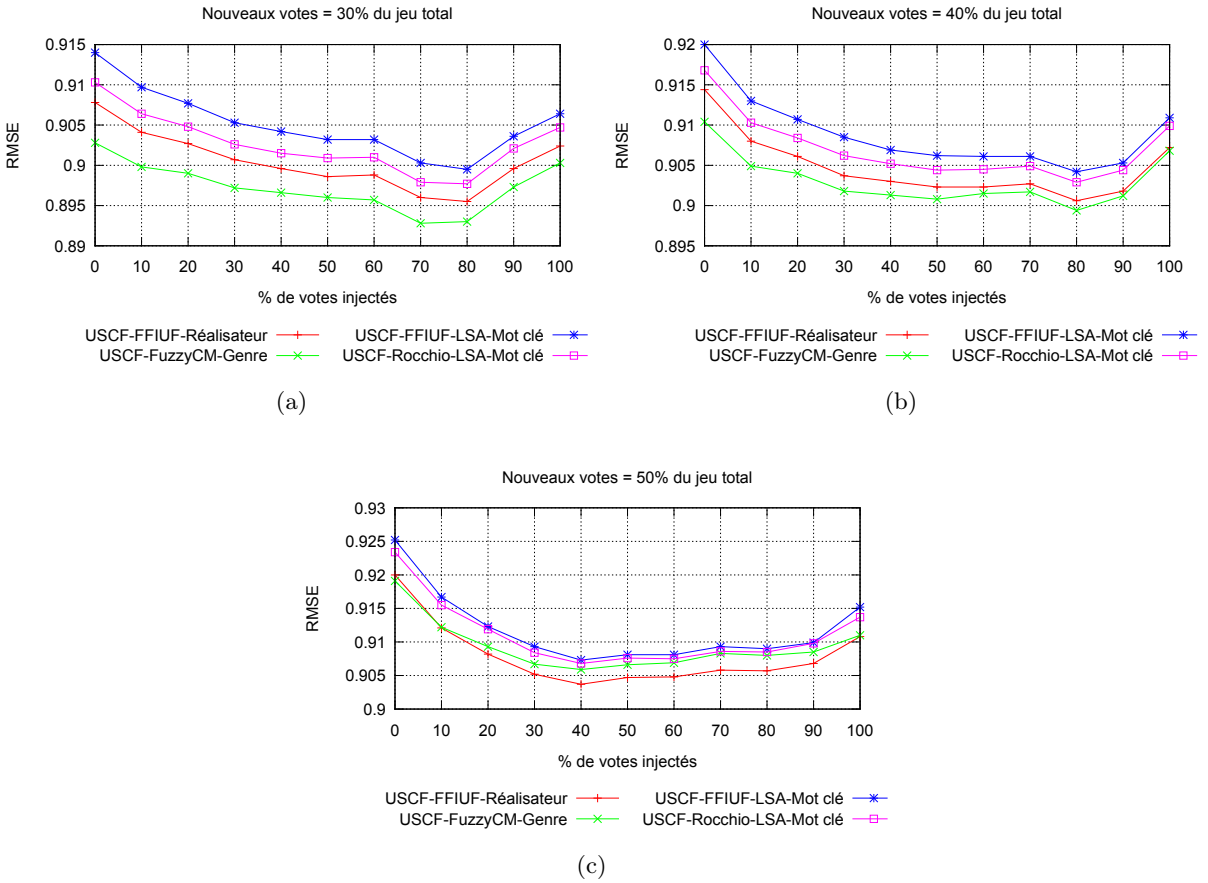


FIGURE 6.3 – Évaluation de la stabilité du modèle sémantique des utilisateurs

70% des votes du jeu initial. Dans les graphiques 6.3(b) et 6.3(c) les découpages correspondent respectivement à 60/40 et 50/50. L'axe des abscisses de chaque graphique représente le pourcentage de nouveaux votes injectés dans la matrice des votes M_v à chaque expérimentation. Ainsi, la valeur 10 indique que 10% des votes les moins récents du jeu $Jeu_{Nouveaux\ votes}$ ont été ajoutés à la matrice des votes M_v, la valeur 100 indique que tous les votes du jeu $Jeu_{Nouveaux\ votes}$ ont été ajoutés à la matrice des votes et la valeur 0 indique que le même jeu (Jeu_{MSU}) est utilisé pour la construction du MSU et pour la recommandation. Les prédictions sont calculées pour 60 voisins, en utilisant la matrice des votes M_v et les similarités entre les utilisateurs calculées à partir du MSU précédemment construit à partir du jeu Jeu_{MSU}. Il est à noter que pour évaluer les recommandations, les votes injectés sont également découpés selon la proportion 80/20, 80% sont ajoutés à la matrice des votes, et 20% au jeu test. L'expérimentation a été menée pour les attributs :

Genre : la méthode FuzzyCM a été utilisée pour la construction du MSU, elle est notée par USCF-FuzzyCM-Genre sur les graphiques.

Réalisateur : la méthode FFIUF-WR avec filtrage des descripteurs a été utilisée pour la construction du MSU, elle est notée USCF-FFIUF-Réalisateur sur les graphiques.

Mot Clé : nous avons utilisé deux méthodes. La première FFIUF avec réduction de la dimension par application d'une LSA, notée USCF-FFIUF-LSA-MotClé sur les graphiques.

La deuxième en appliquant la méthode Rocchio avec réduction de la dimension par LSA, notée USCF-Rocchio-LSA-MotClé sur les graphiques.

En analysant les courbes du graphique 6.3(a), on remarque qu'elles présentent toutes la même allure. La RMSE diminue jusqu'à atteindre un seuil égal à 70% du taux de votes injectés puis augmente. Le premier constat à faire est que l'ajout de nouveaux votes augmente la précision des recommandations. Ceci implique que les communautés d'utilisateurs identifiées à partir du profil sémantique des utilisateurs (MSU) restent valables et résistent à l'ajout de nouvelles évaluations. Nous rappelons que le filtrage collaboratif se base essentiellement sur les votes des plus proches voisins de l'utilisateur courant, donc l'augmentation du nombre de votes des voisins d'un utilisateur, améliore la pertinence des recommandations qu'il reçoit. Ainsi, si le système identifie deux utilisateurs comme étant assez proches, et si le premier attribue une évaluation positive à un item, alors il y a de fortes chances que ce dernier soit recommandé au deuxième. Le deuxième constat est qu'à partir de 70% de votes injectés, ce qui correspond à un taux de nouveaux votes de l'ordre de 23% du jeu total utilisé pour la recommandation, la précision commence à diminuer. La diminution de la précision peut indiquer le début de la divergence entre le profil usage et le profil sémantique. Toutefois, la diminution de la précision reste assez faible sans atteindre sa valeur initiale même après l'injection de tous les votes, soit un taux de nouveaux votes de l'ordre de 30% du total des votes utilisés pour la recommandation. Les mêmes constats peuvent être faits pour les deux autres graphiques avec des valeurs de seuils différents, 80% pour le découpage 60/40 soit 32% de nouveaux votes (graphique 6.3(a)), 40% pour le découpage 50/50, soit un taux de nouveaux votes de l'ordre de 28% du total des votes utilisés pour la recommandation. Ce dernier constat signifie que le profil sémantique des utilisateurs résiste à un taux de nouvelles évaluations au delà de 50% de la totalité des votes. Ce qui signifie que les similarités établies à partir du profil sémantique des utilisateurs sont assez robustes.

Ceci nous amène à conclure que, le fait que l'adaptation du profil sémantique des utilisateurs soit exécutée en mode offline ne dégrade pas la pertinence des recommandations, mais permet au contraire de les améliorer. Toutefois, la détermination de la fréquence d'adaptation du profil sémantique des utilisateurs reste à étudier et peut être étroitement liée aux propriétés du e-service concerné.

6.1.4 USCF vs filtrage basé sur le contenu et filtrage collaboratif

Comparaison de la précision des prédictions de USCF par rapport à une approche collaborative

La figure 6.4 compare les performances de notre approche USCF à une approche purement collaborative. Les performances sont évaluées en mesurant la précision de la prédiction des votes (RMSE) en faisant varier les N plus proches voisins de 10 à 100 (axe des abscisses). Pour notre approche hybride, nous avons représenté les performances de 4 expérimentations : la combinaison *Origine-Acteur-Réalisateur* représentée par la courbe intitulée *USCF-Origine-Acteur-Réalisateur*, la combinaison *Genre-Acteur-Réalisateur* représentée par la courbe intitulée *USCF-Genre-Acteur-Réalisateur*, l'attribut *Tag* représenté par la courbe *USCF-Tag* et l'attribut *mot-clé* représenté par la courbe *USCF-Mot Clé*. Les trois premières expérimentations concernent les items structurés, la quatrième (les mots clés) concerne les items non structurés. UBCF (User Based Collaborative Filtering, page 73) et IBCF (Item Based Collaborative Filtering, page 73) sont les algorithmes collaboratifs par rapport auxquels nous avons comparé notre approche.

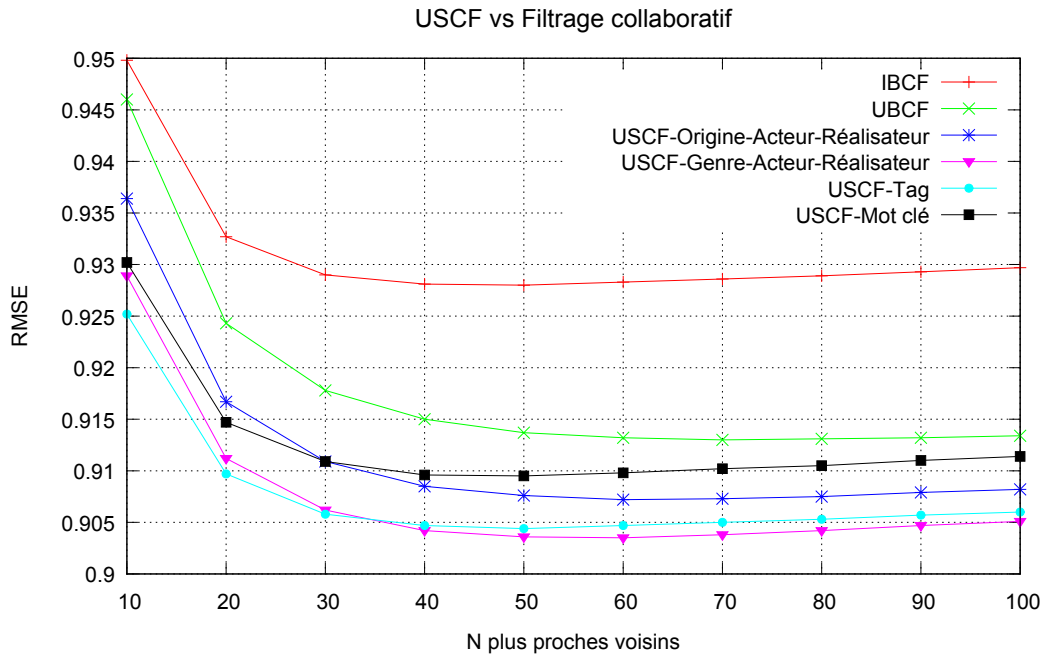


FIGURE 6.4 – Évaluation de la précision des prédictions de USCF par rapport à une approche collaborative (UBCF et IBCF)

En analysant les courbes du graphique, on constate que toutes les courbes ont la même allure, la RMSE diminue jusqu'à une valeur donnée de N (plus proches voisins) puis augmente. Toutes les courbes convergent pour N entre 60 et 70 voisins. Les précisions des prédictions de votes de USCF sont supérieures à celles observées par UBCF, qui elles mêmes sont supérieures à celles enregistrées par IBCF et ce pour tous les voisins. La meilleure performance est obtenue par *USCF-Genre-Acteur-Réalisateur* dont la valeur du RMSE est égale à 0.9035 pour 60 voisins, soit un gain de l'ordre de 1 point par rapport à UBCF dont la RMSE est égale à 0.9132 pour le même nombre de voisins.

En conclusion, on peut dire qu'en plus du traitement du problème de passage à l'échelle, dont souffrent les approches collaboratives, à travers non seulement la réduction de la dimension du profil sémantique des utilisateurs mais également l'exécution d'une partie du traitement en mode offline, les recommandations proposées par notre approche sont plus pertinentes que celles effectuées par une approche purement collaborative.

Comparaison de la précision d'une Top-N liste recommandée par USCF par rapport à une approche basée sur le contenu

La figure 6.5 compare les performances de notre approche USCF par rapport à une approche basée sur le contenu CB sur le graphique (voir page 74). Les performances sont évaluées en calculant la précision d'une Top-N liste d'items recommandés (voir page 74) avec N égal à 5. L'axe des abscisses représente les différentes combinaisons d'attributs expérimentées, aussi bien par notre approche USCF que par CB. Pour notre approche, USCF, nous avons conservé les mêmes algorithmes représentés dans 6.4 en y ajoutant la combinaison *Acteur-Réalisateur*.

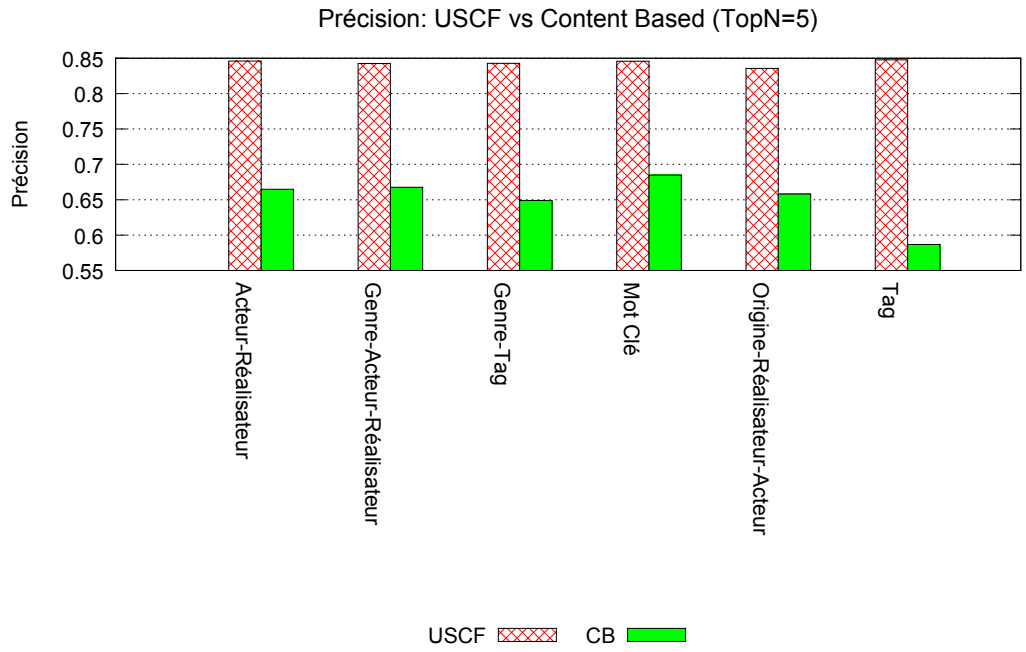


FIGURE 6.5 – Évaluation de la précision d'une Top-5 liste recommandée par USCF vs CB

Pour les différentes combinaisons d'attributs, on constate que la précision de USCF est supérieure à celle observée par CB. Les précisions de CB varient entre 58% et 69%, la meilleure performance de CB est enregistrée par *Mot Clé*. Pour USCF, les performances enregistrées par les différentes expérimentations sont assez proches tout en étant supérieures à 80%. Une précision de 80% signifie que dans une liste composée de 5 recommandations, 4 sont pertinentes pour l'utilisateur, alors que pour CB, seulement 2 à 3 sont pertinentes.

En plus de sa supériorité en terme de pertinence des recommandations, USCF produit des recommandations pertinentes avec très de peu de données sémantiques. Ainsi, nous avons vu qu'avec uniquement l'attribut *Genre*, nous obtenons des performances supérieures à celles observées par UBCF, ce qui n'est pas le cas de l'approche CB qui exige de disposer d'une quantité variée de données sémantiques pour garantir des recommandations pertinentes. Par ailleurs, USCF est capable de fournir des recommandations pertinentes pour des items structurés et des items non structurés.

Comparaison de la diversité des recommandations d'une Top-N liste

La figure 6.6 représente la diversité des recommandations d'une TOP-N liste produites par notre approche USCF, l'approche collaborative UBCF et l'approche basée sur le contenu CB, pour N égale à 5. Nous rappelons que la diversité est le point fort des approches collaboratives et le point faible des approches basées sur le contenu, et ce en raison du principe même de fonctionnement de chaque approche. En effet, pour CB, l'algorithme recommande les items ayant un contenu semblable à ceux appréciés par le passé par l'utilisateur. Pour UBCF, les items recommandés sont ceux appréciés par les utilisateurs semblables à l'utilisateur courant, ce qui explique que les recommandations peuvent être très diversifiées, un même utilisateur peut apprécier des livres de cuisine et des livres policiers.

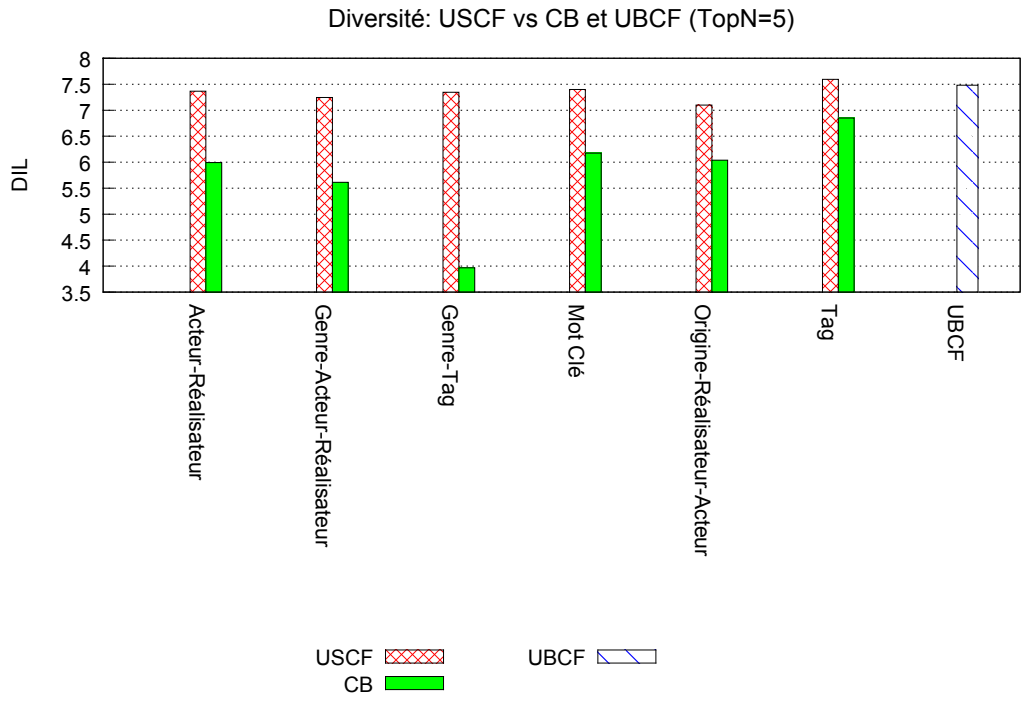


FIGURE 6.6 – Évaluation de la diversité d'une Top-5 liste recommandée par USCF vs CB et UBCF

La diversité est calculée en mesurant la dissimilarité des items composant la top-N liste (voir page 75). L'axe des abscisses reprend les mêmes expérimentations figurant dans la figure 6.5 en y ajoutant UBCF. En analysant ce graphique, on y voit que la diversité des recommandations produites par USCF est équivalente à celle enregistrée par UBCF, et est largement supérieure à celle obtenue par CB et ce pour les différentes combinaisons d'attributs. La diversité des recommandations obtenues par notre approche est due à l'application du principe du filtrage collaboratif pour la détermination des items à recommander. L'information sémantique n'étant prise en compte que pour la détermination du profil sémantique des utilisateurs. Ainsi, le système recommande à l'utilisateur courant les items appréciés par les utilisateurs avec lesquels il partage les mêmes préférences quant au contenu des items.

6.1.5 Évaluation de la complexité temporelle

Nous déterminons dans cette section la complexité temporelle de notre approche, User Semantic Collaborative Filtering (USCF). USCF est constituée de deux composants. L'apprentissage du Modèle Sémantique des Utilisateurs (MSU) qui est exécuté en mode offline, et le calcul de la prédiction des votes pour l'utilisateur courant qui est exécuté en temps réel. La détermination des plus proches voisins de chaque utilisateur est également effectuée en mode offline en calculant la similarité entre tous les couples d'utilisateurs à partir de leur profil sémantique.

Complexité de l'apprentissage du MSU

L'apprentissage du modèle sémantique des utilisateurs (MSU) est constitué de $|L|$ processus exécutés de façon parallèle. Chaque processus est chargé de l'apprentissage du Modèle Sémantique des Utilisateurs par Attribut (MSUA) correspondant à un attribut parmi les attributs pertinents constituant la liste L. La complexité de la construction du MSU correspond alors à la complexité du processus ayant le coût le plus élevé en temps de calcul comme le donne la formule (6.1)

$$comp(MSU) = max\left\{comp(MSUA_A), A \in L\right\} \tag{6.1}$$

Or, la complexité de l'apprentissage du $MSUA_A$ dépend de l'algorithme appliqué. Ainsi, la complexité des algorithmes Fuzzy-C-Means est de l'ordre $O(nc^2dn_{it})$ [Almeida and Sousa, 2006], et K-Means est de l'ordre $O(ncdn_{it})$ [Xu and Wunsch, 2005] avec n le nombre d'objets à classifier, c le nombre de clusters, d la dimension d'un objet et n_{it} le nombre d'itérations. De plus, la complexité d'une décomposition en valeurs singulières (SVD) appliquée à une matrice $m \times n$ est de l'ordre de $O(min\left\{mn^2, m^2n\right\})$ [Golub and F., 1996; Holmes *et al.*, 2007]. Le tableau 6.1 donne la complexité de chacune des méthodes que nous avons proposées pour l'apprentissage du MSUA. Nous rappelons, par ailleurs, la signification des symboles utilisés, à savoir, $|I|$ le nombre d'items, $|U|$ le nombre d'utilisateurs, K_A le nombre de clusters (mais également la dimension du Modèle sémantique des utilisateurs pour l'attribut A, $MSUA_A$), $|F_A|$ le nombre de descripteurs de l'attribut A et $MSIA_A$ la matrice sémantique des items pour l'attribut A constituée de $|I|$ lignes et $|F_A|$ colonnes.

Méthode		Complexité : $comp(MSUA_A)$
FuzzyCM		$O(\|I\| K_A^2 \|U\| n_{it})$
KMeans		$O(\|I\| K_A \|U\| n_{it})$
FFIUF	Filtrage	$O(\|U\|)$
	LSA(SVD $\|U\| \times \|F_A\|$)	$O(min(\|U\|^2 \|F_A\|, \|U\| \|F_A\|^2)$
Rocchio	LSA(MSIA)(SVD $\|I\| \times \|F_A\|$)	$O(min(\|I\|^2 \|F_A\|, \|I\| \|F_A\|^2)$
	Filtrage	$O(\|U\| \|F_A\|)$
	LSA(SVD $\|U\| \times \|F_A\|$)	$O(min(\|U\|^2 \|F_A\|, \|U\| \|F_A\|^2)$

Table 6.1 – Complexité temporelle de l'apprentissage du MSUA

Complexité de USCF vs Filtrage collaboratif

La complexité de notre approche, USCF, est découpée en deux parties. La complexité des composants exécutés en mode offline (apprentissage du MSU et calcul des similarités entre tous les couples d'utilisateurs), et celle du composant exécuté en temps réel chargé du calcul de la prédiction des votes, $pred(u_a, i)_{i \in I \setminus I_{u_a}}$, de l'ensemble des items non observés par l'utilisateur courant u_a. Le tableau 6.2 donne la complexité de notre approche (USCF) et la compare avec celle de l'algorithme de filtrage collaboratif basé sur les utilisateurs (UBCF). En analysant le tableau, on voit que la complexité de la partie temps réel de notre approche est linéaire alors que celle du filtrage collaboratif est de l'ordre de $O(|U||I|)$. Ceci est dû au fait que la détermination des plus proches voisins de l'utilisateur courant u_a (calcul des similarités entre u_a et les autres utilisateurs de U) se fait en temps réel, en plus du fait que les utilisateurs sont définis dans la dimension des items I. Dans notre approche, $O(|U|^2K)$ représente le coût de calcul des similarités entre tous les couples d'utilisateurs. Or, comme K représente la dimension du profil sémantique

Algorithme			Complexité
USCF	Offline	MSU	$max\{comp(MSUA_A), A \in L\}$
		Similarités de tous les utilisateurs : $sim(u,v)_{u\in U\, et\, v\in U}$	$O(\|U\|^2)$
	Temps réel	Prédiction pour u_a : $pred(u_a,i)_{i\in I\setminus I_{u_a}}$	$O(\|I\setminus I_{u_a}\|)$
UBCF	Temps Réel	Similarités avec u_a : $sim(u_a,v)_{v\in U}$	$O(\|U\|\|I\|)$
		Prédictions pour u_a :$pred(u_a,i)_{i\in I\setminus I_{u_a}}$	$O(\|I\setminus I_{u_a}\|)$

TABLE 6.2 – Complexité temporelle de USCF vs UBCF

des utilisateurs, et que ce dernier a une valeur réduite très inférieure à $|U|$, la complexité peut être ramenée à $O(|U|^2)$.

6.2 Conclusion

Nous nous sommes intéressés dans cette thèse aux systèmes de recommandation personnalisée. Nous avons présenté dans un premier temps, leurs principe de fonctionnement, et les critères permettant leur évaluation. Nous avons présenté, dans un deuxième temps, les deux principales techniques de recommandation personnalisée : la recommandation collaborative et la recommandation basée sur le contenu ainsi que les différentes façons de les hybrider. Plusieurs méthodes d'hybridation ont été ainsi développées [Balabanović and Shoham, 1997; Burke, 2002; Burke, 2007]. La plupart de ces systèmes sont orientés processus, il s'agit d'exécuter le filtrage collaboratif sur les résultats issus d'un filtrage sur le contenu ou vice versa, soit pour résoudre le problème des données manquantes, soit pour résoudre le problème du démarrage à froid.

Partant de ce constat, l'objectif de cette thèse a été de proposer une autre technique d'hybridation en étudiant les bénéfices de l'exploitation combinée d'une part, des informations sémantiques des items à recommander, avec d'autre part, le filtrage collaboratif qui tire profit du comportement de la communauté des usagers du e-service. L'aspect sémantique des items est défini par la description textuelle disponible sur les items à recommander. Les informations issues de l'analyse des usages sont constituées des évaluations émises par les utilisateurs sur les items qu'il ont consultés. Les évaluations sont représentées sous forme de votes explicites ou implicites définis sur une échelle de valeurs. Seules les applications évoluant dans un contexte à faible risque sont ciblées. Dans un contexte à faible risque, les items se caractérisent par une faible utilité et une faible complexité. Une faible utilité signifie que le coût enduré par l'utilisateur suite à la sélection d'une mauvaise recommandation est faible. La complexité se mesure par les propriétés de l'item à prendre en considération lors de la conception d'un système de recommandation. C'est le cas par exemple de la location de DVDs, de l'achat de livres, de la visualisation de film ou du choix d'un restaurant. Ce n'est pas le cas par exemple, des applications dédiées à la vente d'automobiles, aux transactions financières, à l'achat d'ordinateur, à l'achat d'appareil photo ou à l'achat de voyages. Dans ce contexte, nous avons supposé ne disposer d'aucune information sur les usagers à part un numéro permettant de les identifier. La méthodologie que nous avons adoptée est délibérément empirique. Elle repose sur une série d'expérimentations élaborée à partir de nos différentes propositions, sur la base d'un jeu de données largement utilisé par la communauté scientifique, le MovieLens Dataset [JeuMovieLens, 2014].

Le système que nous proposons, User Semantic Collaboratif Filtering (USCF), est constitué d'un seul système de recommandation intégrant les données sémantiques et les données issues de l'analyse des usages dans un algorithme de recommandation personnalisée hybride. Selon la classification des types d'hybridations effectuée par Burke [Burke, 2007], notre approche d'hybridation peut être classée comme étant une technique Méta-level. L'apprentissage d'un premier modèle est effectué en utilisant le contenu des items, le Modèle Sémantique des Utilisateurs (MSU), il est par la suite exploité dans une approche collaborative basée sur les utilisateurs.

Nous avons tout d'abord identifié les différentes sources d'informations sémantiques décrivant les items (structurées, semi structurées ou non structurées) que nous avons ramenées à une structure du type attribut-descripteurs. Ces attributs permettent de construire un modèle sémantique des utilisateurs. Seuls les attributs pertinents sont sélectionnés, c'est-à-dire les attributs ayant une importance capitale dans le choix des utilisateurs. Étant donnée que l'identification de tels attributs est liée au domaine d'application du e-service, nous avons supposé, pour simplifier, que la sélection est effectuée par des experts du domaine. Pour chaque attribut pertinent, nous avons

construit un Modèle Sémantique des Utilisateurs par Attribut (MSUA). Le modèle sémantique des utilisateurs est alors la concaténation des différents modèles sémantiques des utilisateurs par attribut.

Pour l'apprentissage du modèle sémantique des utilisateurs par attribut (MSUA), nous avons tout d'abord identifié deux catégories d'attributs, les attributs indépendants et les attributs dépendants. Cette catégorisation a été effectuée sur la base du lien existant entre le nombre de valeurs (descripteurs) d'un attribut et le nombre d'items. Les attributs indépendants se caractérisent par un nombre stable de valeurs indépendamment du nombre d'items. Les attributs dépendants se caractérisent, au contraire, par un nombre de valeurs variant selon le nombre d'items. Pour chaque catégorie d'attribut, nous avons présenté des approches différentes pour l'apprentissage du modèle sémantique des utilisateurs. Dans nos premières contributions, nous avons utilisé des approches de classification non supervisée pour construire le modèle sémantique des utilisateurs pour les attributs indépendants. Pour ce faire, nous avons appliqué deux algorithmes de clustering pour partitionner les items définis selon leur profil usage, le Fuzzy C-Means et le K-Means. Le Fuzzy C-Means pour faire un clustering flou appliqué à des attributs multivalués, alors que le K-Means a été utilisé sur les attributs monovalués. Les contributions suivantes ont concerné les attributs dépendants pour lesquels nous avons appliqué des approches issues du filtrage d'informations. Comme première proposition, nous avons défini une méthode basée sur le nombre de votes associé à chaque descripteur pour l'apprentissage du modèle sémantique des utilisateurs par attribut. Une deuxième proposition a consisté à utiliser l'algorithme de Rocchio pour la construction du MSUA. Partant du constat que les attributs dépendants ont généralement un nombre de valeurs élevé pouvant dépasser dans certains cas le nombre d'items, nous avons proposé deux techniques pour réduire le nombre de descripteurs. La première se base sur le filtrage des descripteurs en ne conservant que les plus pertinentes d'entre eux, la pertinence d'un descripteur étant guidée par le nombre de votes qui lui est associé. La deuxième applique une analyse sémantique latente (LSA) d'un rang très inférieur au nombre de descripteurs.

Le modèle sémantique des utilisateurs constitue le nouveau profil utilisateur, il est utilisé pour calculer les similarités entres les différents couples utilisateurs afin de définir, pour chaque usager, la communauté avec laquelle il partage des goûts similaires. Cette communauté est en fait constituée de ses N plus proches voisins, N étant défini de façon empirique. Le système recommande alors à l'utilisateur courant les items appréciés par les utilisateurs de sa communauté. Pour ce faire, il prédit pour chaque item non observé par l'utilisateur courant, la valeur de son vote à partir des votes attribués par les utilisateurs de sa communauté en appliquant un algorithme de filtrage collaboratif basé sur les usagers. Les items ayant une valeur de vote prédite supérieure à un seuil donné sont identifiés comme pertinents pour l'utilisateur et lui sont recommandés.

Les bénéfices de notre approche hybride par rapport à une approche collaborative ou par rapport à d'autres approches hybrides peuvent être résumés par les points suivants :

Sensibilité aux données manquantes :

Les données manquantes sont un problème commun à la majorité des systèmes de recommandation en raison du fait que les utilisateurs notent très peu d'items. Ils sont à l'origine de nombreux problèmes auxquels doivent faire face les algorithmes de filtrage collaboratif basés sur les voisins vu que la similarité entre deux utilisateurs est basée sur les items qu'ils ont co-évalués.

Dans notre approche, le modèle sémantique des utilisateurs est utilisé pour calculer les similarités entre les utilisateurs. En augmentant la densité des données de ce modèle par la réduction de la dimension représentée par les descripteurs des attributs, nous avons été en mesure de calculer la similarité entre les utilisateurs n'ayant aucun ou très peu d'items en commun. Ce qui a permis pour un utilisateur donné d'élargir ou d'améliorer la communauté d'utilisateurs auquel il appartient.

Passage à l'échelle :

La massivité des données est l'un des problèmes récurrents des systèmes de recommandation vu l'augmentation constante du nombre d'items et d'utilisateurs. L'algorithme de filtrage collaboratif connu pour ses bonnes performances en terme de précision des prédictions de votes, ne passe plus l'échelle en raison, entre autres, du coût élevé nécessaire au calcul des similarités entre les utilisateurs. Notre approche a traité ce problème à trois niveaux :

Parallélisation de l'apprentissage du profil sémantique des utilisateurs : l'apprentissage du modèle sémantique d'utilisateur est un traitement totalement parallélisable. En effet, les apprentissages des différents modèles sémantiques des utilisateurs par attribut le constituant sont totalement indépendants, ils peuvent par conséquent s'exécuter en parallèle.

Traitement en mode offline : l'apprentissage du profil sémantique des utilisateurs et le calcul des similarités entre les utilisateurs sont exécutés en mode offline. Pour chaque utilisateur, seuls ses similarités avec ses plus proches voisins sont stockées. L'étape de prédiction se fait, quant à elle, en temps réel à partir des similarités pré-calculées. Nous avons vu dans les expérimentations que nous avons menées, la robustesse du profil sémantique face à l'ajout de nouveaux votes, ce qui permet de réduire la fréquence de mise à jour du modèle sémantique des utilisateurs et des similarités entre les utilisateurs.

Réduction de la dimension : la dimension réduite du modèle sémantique des utilisateurs permet de réduire le coût nécessaire au calcul des similarités entre les utilisateurs.

Traitement des différentes sources d'information sémantique :

L'information sémantique disponible sur les items peut être soit structurée, soit non structurée (décrite par un texte libre) soit semi-structurée (combinaison des deux). L'approche que nous proposons traite ces trois types d'informations. Lorsque l'information est non structurée, et après une étape d'indexation, le profil sémantique relatif à cette information peut être construit en appliquant l'une des méthodes associées aux attributs dépendants. Si l'information est structurée, pour chaque attribut sélectionné par les experts du domaine, et en fonction de la catégorie de l'attribut indépendant ou dépendant, la méthode correspondante est appliquée pour l'apprentissage du modèle sémantique des utilisateurs. Toutefois, nous avons vu qu'il n'est pas nécessaire de disposer d'une variété d'information pour obtenir des recommandations pertinentes comme c'est le cas de la recommandation basée sur le contenu.

Amélioration de la précision :

L'apprentissage d'un nouveau profil utilisateur, *le profil sémantique des utilisateurs*, inférant les préférences des utilisateurs pour une sélection des données sémantiques des items, et son

utilisation dans un algorithme de filtrage collaboratif basé sur les voisins, a permis de maintenir voire améliorer la précision des prédictions des votes par rapport à un algorithme de filtrage collaboratif exploitant uniquement le profil usage. L'amélioration est toutefois variable en fonction de la combinaison d'attributs et des algorithmes d'apprentissage du modèle sémantique des utilisateurs par attribut appliqués. Par ailleurs, dans toutes les expérimentations que nous avons menées, notre approche hybride a affiché une supériorité au niveau de la pertinence des recommandations par rapport à une approche basée sur le contenu.

6.3 Perspectives

Extension du domaine d'application

Nous avons expérimenté notre approche et les différents modèles que nous avons proposés sur le domaine de la recommandation des films et plus particulièrement les jeux de données de MovieLens [JeuMovieLens, 2014]. Cependant, les performances d'un algorithme de recommandation peuvent varier en fonction des données utilisées ou du domaine d'application [Shani and Gunawardana, 2011; Adomavicius and Tuzhilin, 2005]. C'est pour cette raison qu'il serait intéressant de confirmer nos conclusions en expérimentant nos modèles sur d'autres domaines d'applications tels que la recommandation de livres, ou la recommandation d'articles de recherches par exemple.

Détermination des poids initiaux des items dans l'algorithme Fuzzy C means

Dans l'algorithme d'initialisation de la méthode Fuzzy C Means, nous avons émis l'hypothèse que le poids initial d'appartenance d'un item i à un cluster est égal à $1/n$, n étant le nombre de descripteurs décrivant l'item i. Cette hypothèse peut être étendue en explorant d'autres heuristiques issues du domaine de filtrage d'information.

Détermination des attributs pertinents

Une première extension importante de notre travail est l'automatisation de la sélection des attributs pertinents. Dans l'état présent, nous supposons que les attributs pertinents sont sélectionnés par les experts du domaine. Nous avons montré de façon empirique que tous les attributs n'ont pas la même importance chez l'utilisateur. Les expérimentations ont montré également que les attributs, qui intuitivement peuvent être discriminatifs chez l'utilisateur, fournissent des recommandations plus pertinentes. Ce constat peut être un point de départ pour explorer des méthodes d'apprentissage automatique pour déterminer les attributs les plus prédictifs pour les applications présentant un large choix d'attributs candidats.

Sélection de la meilleure approche par attribut

Il a été démontré que la pertinence des recommandations d'un algorithme de recommandation est étroitement lié aux données [Shardanand and Maes, 1995]. Nous avons par ailleurs présenté différentes méthodes pour l'apprentissage du modèle sémantique des utilisateurs pouvant s'appliquer à la même catégorie d'attributs. La question qui se pose alors est quelle méthode appliquer pour un jeu de données donné et un attribut donné ? Est-il possible de définir une hybridation des approches que nous avons présentées ? Une technique qu'il serait intéressant d'étudier est le switching [Burke, 2007]. Le système serait amené à sélectionner, en fonction des données dont il dispose, le modèle sémantique des utilisateurs aboutissant à la meilleure précision.

Combinaison linéaire des attributs

Le modèle sémantique des utilisateurs (MSU) est construit à partir de la fusion des modèles sémantiques des utilisateurs par attribut (MSUA) associés. Tous les attributs sont ainsi traités de la même façon. Or, nous avons vu qu'il peut y avoir des attributs plus prédictifs que d'autres. L'idée serait alors d'associer un poids à chaque attribut représentant son importance dans le domaine. La similarité entre deux utilisateurs est alors obtenue en calculant une combinaison linéaire des similarités calculées à partir de chaque MSUA individuellement comme le définit la formule (6.2).

$$sim(u,w) = \alpha_{A_1} sim_{A_1}(u,w) + \ldots + \alpha_{A_n} sim_{A_n}(u,w) \tag{6.2}$$

avec A_k un attribut, $sim_{A_k}(u,v)$ la similarité entre les utilisateurs u et v calculée à partir du $MSUA_{A_k}$ et α_{A_k} le poids de l'attribut A_i. La somme des poids étant égale à 1. La détermination des poids des attributs peut se faire par apprentissage.

Une autre idée serait de calculer la prédiction de vote d'un couple, (utilisateur item), pour chaque attribut en utilisant le MSUA associé. La combinaison linéaire se fera alors au niveau du calcul de la prédiction des votes comme le définit la formule (6.3), au lieu d'être au niveau du calcul des similarités.

$$pred(u,i) = \alpha_{A_1} pred_{A_1}(u,i) + \ldots + \alpha_{A_n} pred_{A_n}(u,i) \tag{6.3}$$

$pred_{A_k}(u,i)$ étant la prédiction du vote de l'utilisateur u pour l'item i déterminée à partir du modèle sémantique des utilisateur par attribut pour l'attribut A_k, $MSUA_{A_k}$. Par ailleurs, pour certaines applications, il est possible de disposer de l'importance d'un attribut de manière personnalisée par utilisateur. Il est possible alors de calculer la prédiction en utilisant les pondérations personnalisées comme le définit la formule 6.4.

$$pred(u,i) = \alpha_{u,A_1} pred_{A_1}(u,i) + \ldots + \alpha_{u,A_n} pred_{A_n}(u,i) \tag{6.4}$$

α_{u,A_1} est le poids représentant l'importance de l'attribut A_1 pour l'utilisateur u. Une étude approfondie des différentes façons de combiner les modèles sémantiques des utilisateurs par attribut (MSUA) pour faire de la prédiction de votes et/ou de la recommandation serait intéressante.

Traitement du problème du démarrage à froid (cold start)

Le problème du démarrage à froid concerne les items et les utilisateurs lorsqu'ils sont nouvellement introduits dans le système. Un nouvel utilisateur qui n'a noté aucun ou très peu d'items ne peut pas recevoir de recommandation puisque le système ne connaît pas ses préférences, on parle dans ce cas du démarrage à froid pour les utilisateurs. Le même problème se pose également lorsqu'un nouvel item est ajouté dans le système, il ne peut être recommandé avant de recevoir un nombre suffisant d'évaluations, on parle alors de démarrage à froid pour les items.

Démarrage à froid pour les items : le filtrage sur le contenu permet de recommander les nouveaux items n'ayant aucune évaluation en raison du principe même de son fonctionnement. Il serait intéressant d'appliquer le même principe en exploitant le profil sémantique des utilisateurs. Deux propositions peuvent être étudiées. La première concerne uniquement les attributs dépendants, il s'agit de calculer la similarité entre l'utilisateur courant et l'ensemble des nouveaux items. Le Top-N des items les plus similaires lui seront alors recommandés. Les items doivent au préalable être définis dans la même dimension que celle du modèle sémantique des utilisateurs.

La deuxième proposition, consiste à déterminer à l'aide de notre système, USCF, les items à recommander à l'utilisateur courant, puis à calculer la similarité entre ces items et l'ensemble des nouveaux items. Le Top-N des nouveaux items les plus similaires lui seront également recommandés. La similarité entre les items est calculée, dans ce cas, sur la base de leurs profils sémantiques.

Démarrage à froid pour les utilisateurs : étant donné que la similarité entre les utilisateurs est établie à partir du profil sémantique des utilisateurs et non pas à partir de la corrélation entre les votes des items qu'ils ont évalués en commun, il serait intéressant d'expérimenter les différentes méthodes d'apprentissage du profil sémantique des utilisateurs et d'étudier leur comportement vis à vis des utilisateurs ayant très peu de votes. Une solution consisterait à recommander à un nouvel utilisateur, le Top-N des items appréciés par ses plus proches voisins.

Annexe A

L'ontologie de l'attribut Origine du film

Nous présentons dans cette annexe l'ontologie décrivant l'origine des films. Nous rappelons que nous l'avons utilisée pour l'initialisation de l'algorithme KMeans lors de l'apprentissage du modèle sémantique des utilisateurs pour l'attribut *Origine* (voir section 4.4 page 96). Elle est extraite à partir l'ontologie The Movie Ontology (MO), initiée par Amancio Bouza du département informatique de l'université de Zurich [Bouza, 2010]. Pour des raisons de lisibilité, nous la présentons sur six graphiques. La figure A.1 décrit uniquement les concepts généraux sans y inclure les pays. On y voit que les pays sont classés par continent : Afrique (voir figure A.2), Asie (voir figure A.3), Amérique (voir figure A.4), Europe (voir figure A.5), et Océanie (voir figure A.6).

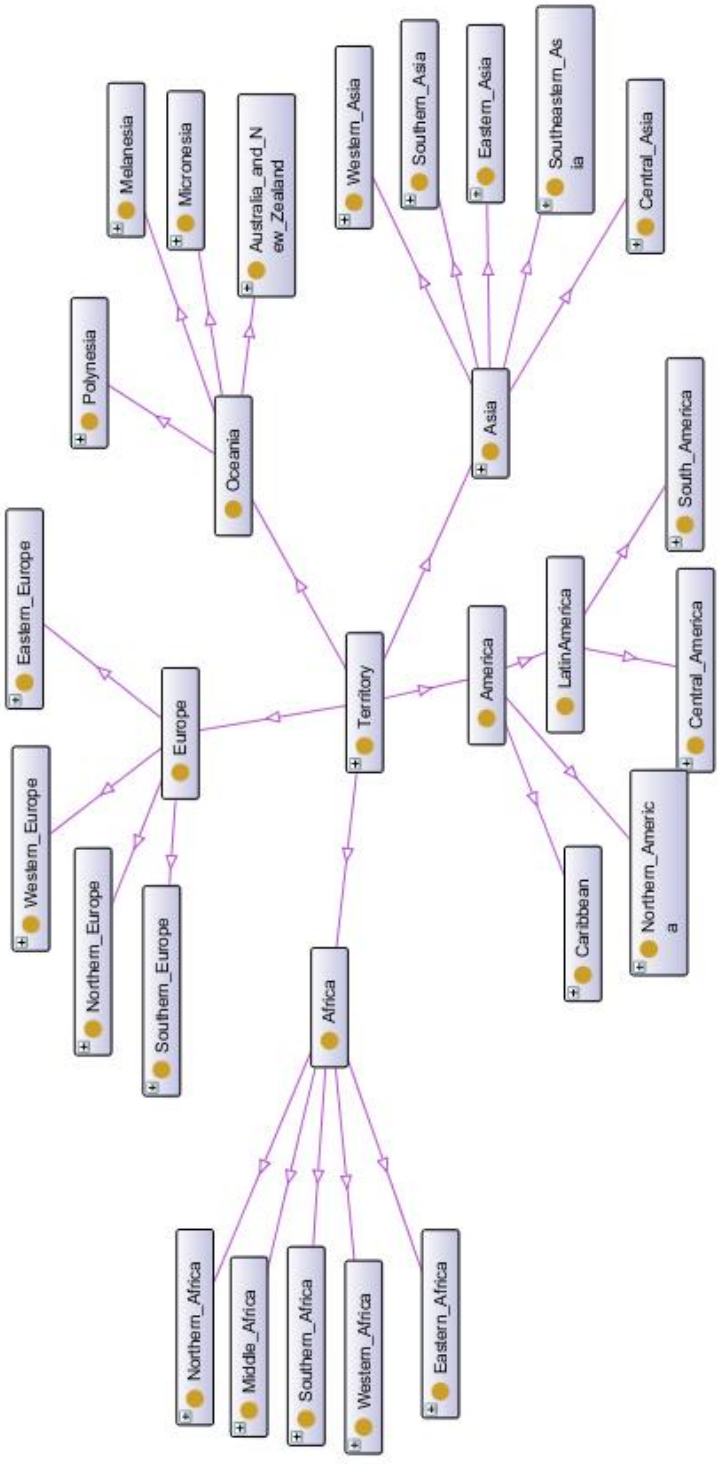


FIGURE A.1 – Ontologie de l'attribut Origine du film

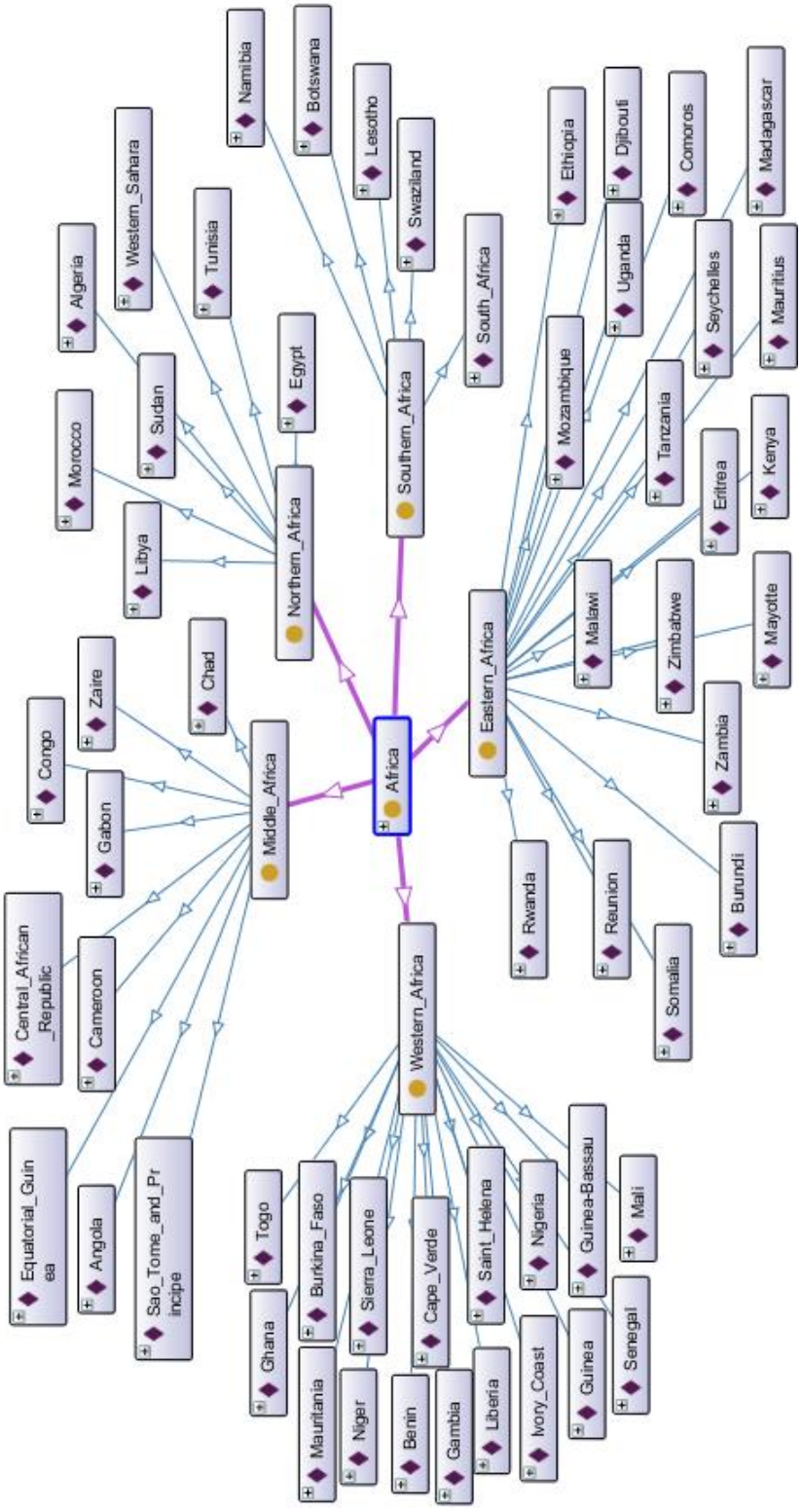


FIGURE A.2 – Ontologie de l'attribut Origine du film pour le continent : Afrique

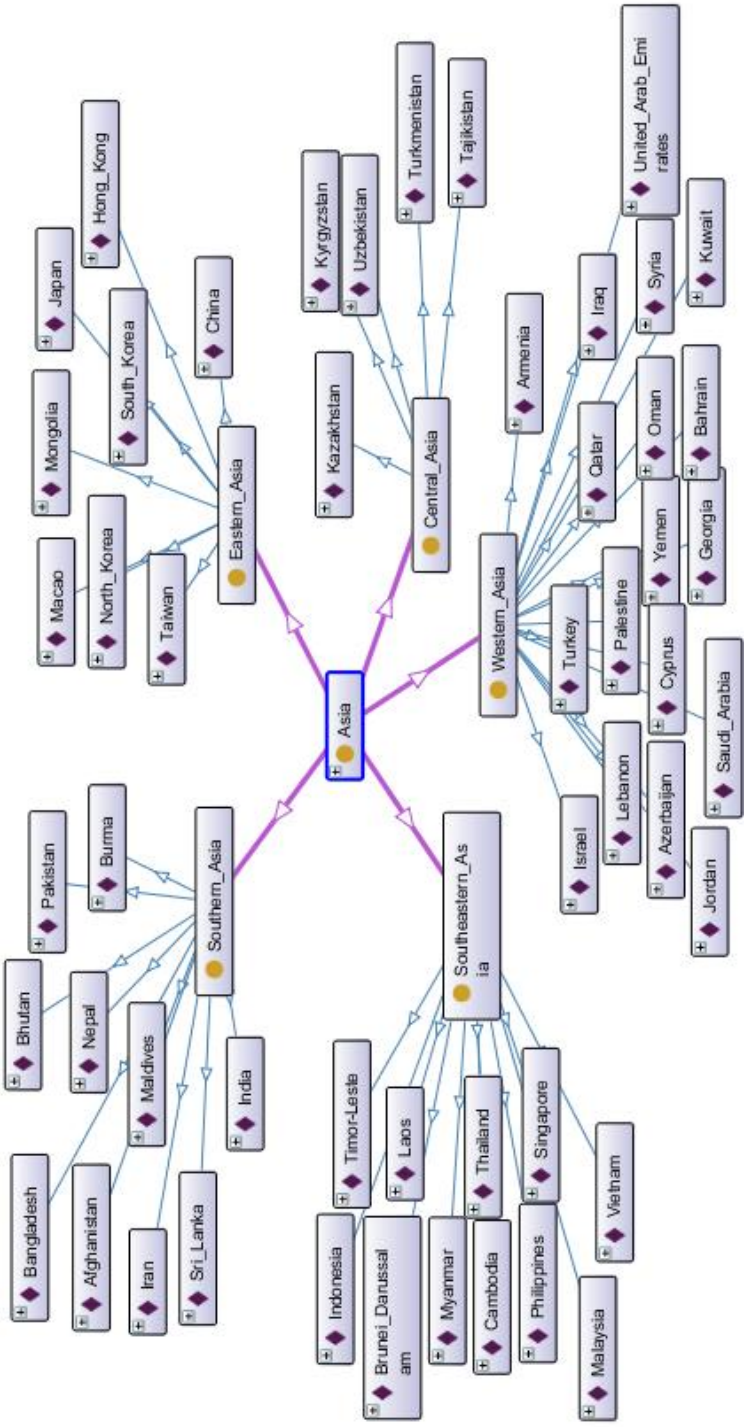


FIGURE A.3 – Ontologie de l'attribut Origine du film pour continent : Asie

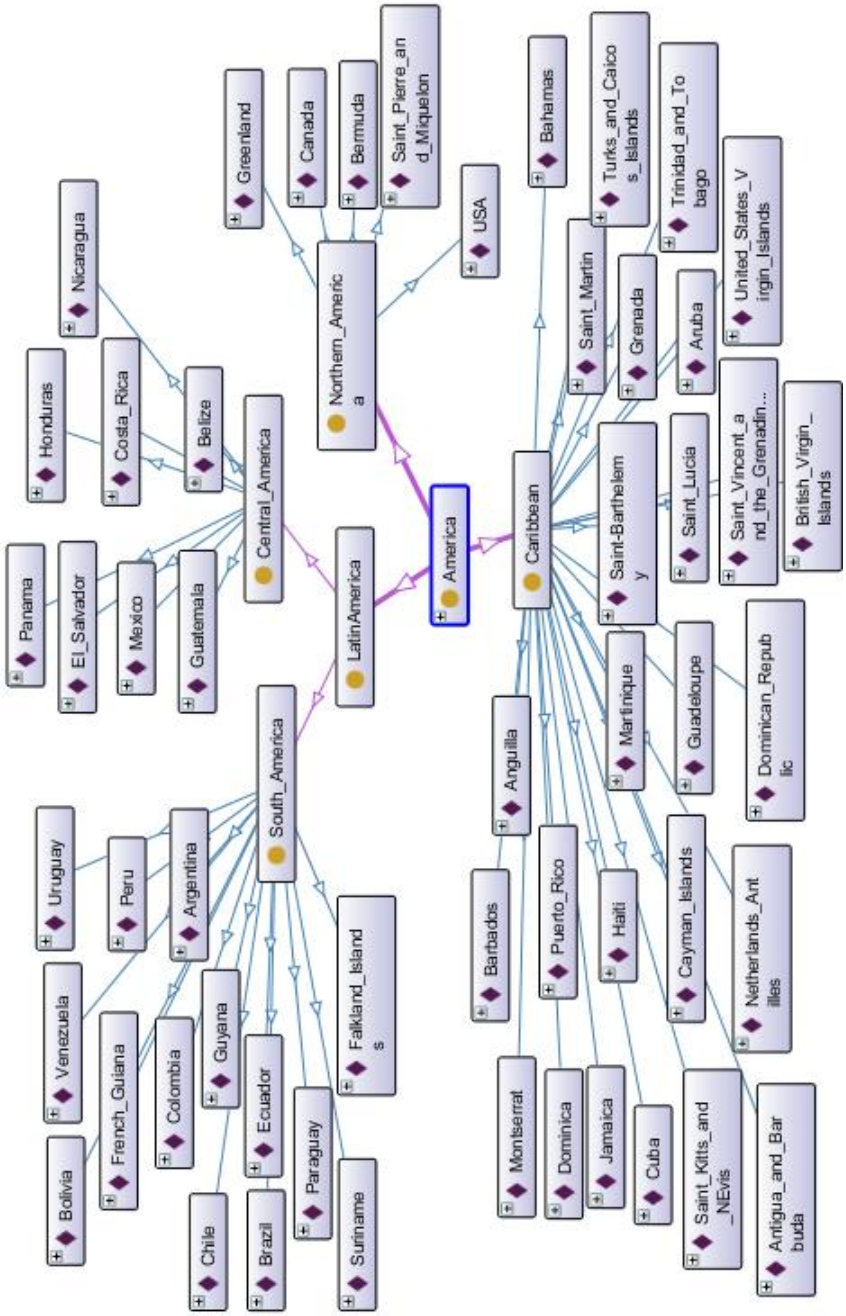


FIGURE A.4 – Ontologie de l'attribut Origine du film pour continent : Amérique

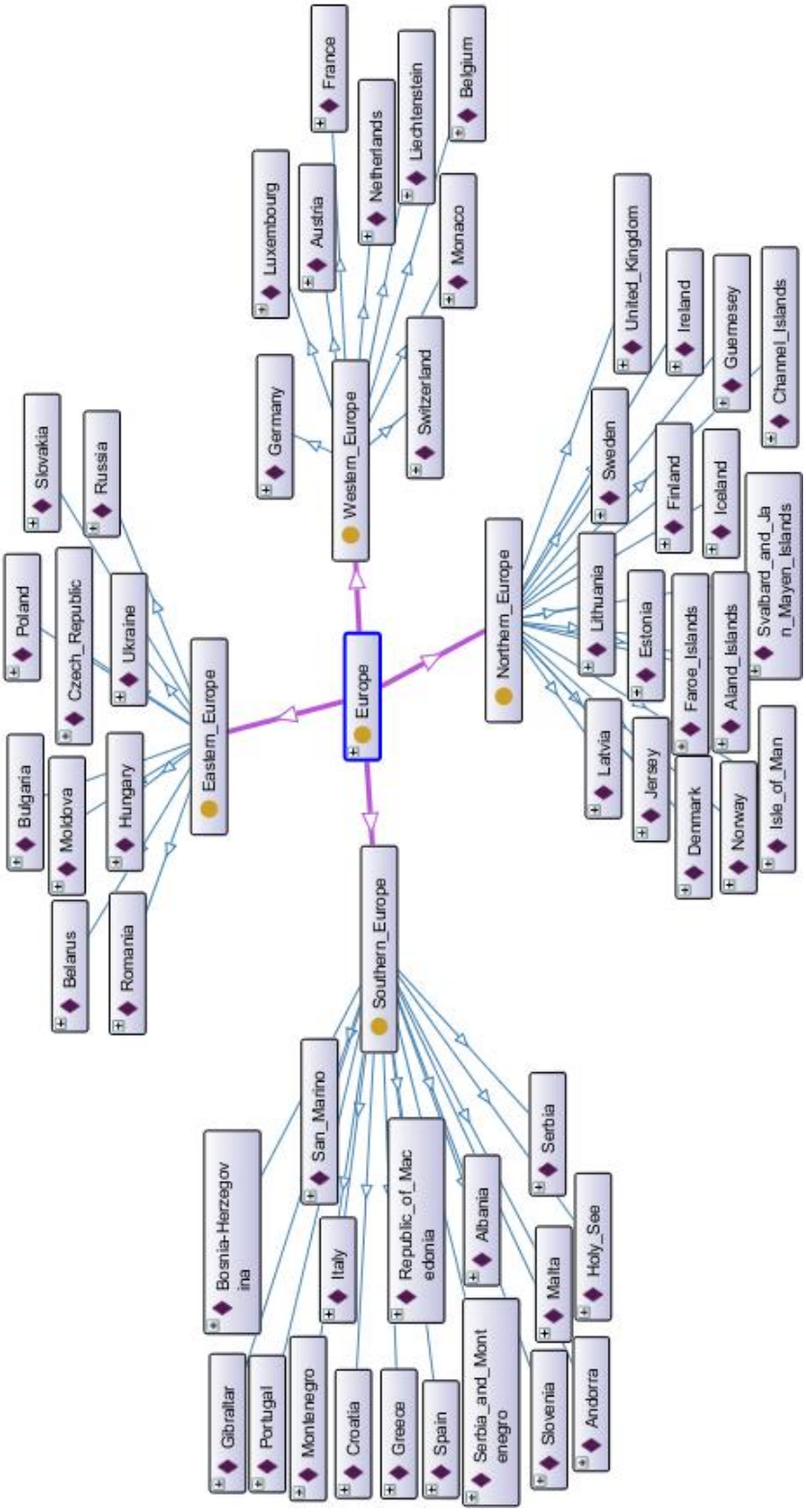


FIGURE A.5 – Ontologie de l'attribut Origine du film pour continent : Europe

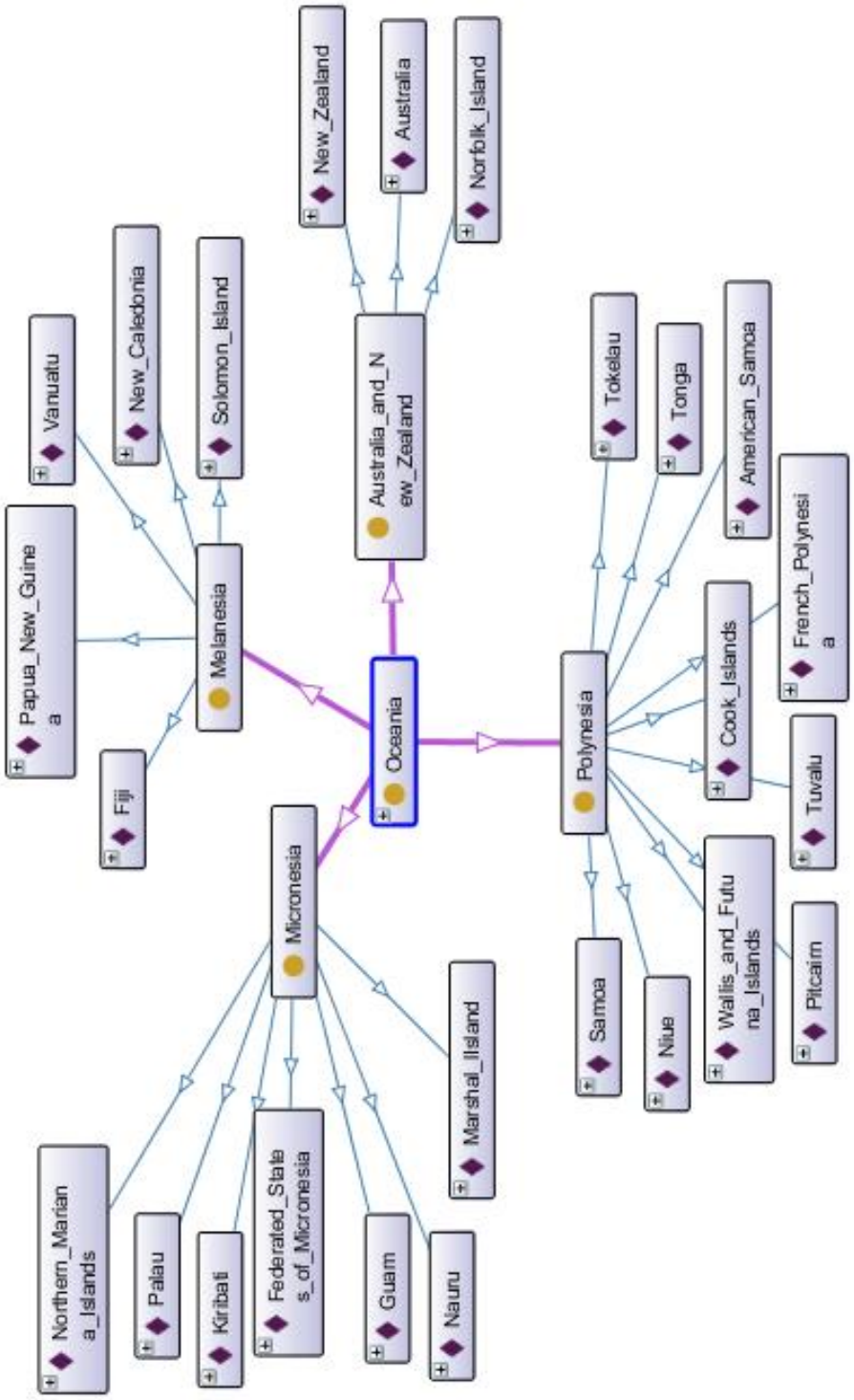


FIGURE A.6 – Ontologie de l'attribut Origine du film pour continent : Océanie

Bibliographie

[Aciar *et al.*, 2007] Silvana Aciar, Debbie Zhang, Simeon Simoff, and John Debenham, (2007). Informed recommender : Basing recommendations on consumer product reviews. Intelligent Systems, IEEE, 22(3), pp.39–47.

[Adomavicius and Tuzhilin, 2005] Gediminas Adomavicius and Alexander Tuzhilin, (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng., 17(6), pp.734–749.

[Aggarwal *et al.*, 1999] Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu, and Philip S. Yu. Horting hatches an egg : A new graph-theoretic approach to collaborative filtering. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '99, pages 201–212, New York, NY, USA, 1999. ACM.

[Ahn *et al.*, 2007] Jae-wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Yeon Syn. Open user profiles for adaptive news systems : Help or harm ? In Proceedings of the 16th International Conference on World Wide Web, WWW '07, pages 11–20, New York, NY, USA, 2007. ACM.

[Al-Shamri and Bharadwaj, 2008] Mohammad Yahya H. Al-Shamri and Kamal K. Bharadwaj, (2008). Fuzzy-genetic approach to recommender systems based on a novel hybrid user model. Expert Systems with Applications, 35(3), pp.1386 – 1399.

[Almazro *et al.*, 2010] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, and William Nzoukou, (2010). A survey paper on recommender systems. CoRR, abs/1006.5278.

[Almeida and Sousa, 2006] R. J. Almeida and J. M. C. Sousa. Comparison of fuzzy clustering algorithms for classification. In Evolving Fuzzy Systems, 2006 International Symposium on, pages 112–117. IEEE, 2006.

[Amazon, 2014] Amazon. www.amazon.com, 2014. consulté, février 2014.

[Anderberg, 1973] Michael R. Anderberg. Cluster analysis for applications. Academic Press, 1973.

[Baeza-Yates and Ribeiro-Neto, 1999] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, 1999.

[Balabanović and Shoham, 1997] Marko Balabanović and Yoav Shoham, (March 1997). Fab : Content-based, collaborative recommendation. Commun. ACM, 40(3), pp.66–72.

[Balabanović, 1998] Marko Balabanović, (January 1998). Exploring versus exploiting when learning user models for text recommendation. User Modeling and User-Adapted Interaction, 8(1-2), pp.71–102.

[Barragáns-Martínez *et al.*, 2010] Ana Belén Barragáns-Martínez, Enrique Costa-Montenegro, Juan C. Burguillo, Marta Rey-López, Fernando A. Mikic-Fonte, and Ana Peleteiro, (2010).

A hybrid content-based and item-based collaborative filtering approach to recommend {TV} programs enhanced with singular value decomposition. Information Sciences, 180(22), pp.4290 – 4311.

[Basilico and Hofmann, 2004] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, pages 9–, New York, NY, USA, 2004. ACM.

[Basu *et al.*, 1998] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification : Using social and content-based information in recommendation. In In Proceedings of the Fifteenth National Conference on Artificial Intelligence, pages 714–720. AAAI Press, 1998.

[Belkin and Croft, 1992] Nicholas J. Belkin and W. Bruce Croft, (December 1992). Information filtering and information retrieval : Two sides of the same coin ? Commun. ACM, 35(12), pp.29–38.

[Bell *et al.*, 2007] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07, pages 95–104, New York, NY, USA, 2007. ACM.

[Ben Ticha *et al.*, 2011a] Sonia Ben Ticha, Azim Roussanaly, and Anne Boyer. User semantic model for hybrid recommender systems. In The 1st Int. Conf. on Social Eco-Informatics - SOTICS, pages 95–101, Barcelona, Espagne, October 2011. IARIA.

[Ben Ticha *et al.*, 2011b] Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. User-feature model for hybrid recommender system. In 4th International Conference on Information Systems and Economic Intelligence-SIIE'2011, Marrakech, Maroc, 02 2011.

[Ben Ticha *et al.*, 2012] Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. User semantic preferences for collaborative recommendations. In Christian Huemer and Pasquale Lops, editors, E-Commerce and Web Technologies, volume 123 of Lecture Notes in Business Information Processing, pages 203–211. Springer Berlin Heidelberg, 2012.

[Ben Ticha *et al.*, 2013] Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. Feature frequency inverse user frequency for dependant attribute to enhance recommendations. In The Third Int. Conf. on Social Eco-Informatics - SOTICS, Lisbon, Portugal, November 2013. IARIA.

[Ben Ticha *et al.*, 2014] Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. User semantic model for dependent attributes to enhance collaborative filtering. In Proceedings of the 10th International Conference on Web Information Systems and Technologies, volume 2, pages 205–212, Barcelona, Spain, April 2014.

[Ben Ticha *et al.*, 2015] Sonia Ben Ticha, Azim Roussanaly, Anne Boyer, and Khaled Bsaïes. Rocchio algorithm to enhance semantically collaborative filtering. In Valérie Monfort and Karl-Heinz Krempels, editors, Web Information Systems and Technologies, Lecture Notes in Business Information Processing (LNBIP), chapter 19. Springer International Publishing Switzerland, 2015.

[Bennett and Lanning, 2007] James Bennett and Stan Lanning. The netflix prize. In Proceedings of KDD Cup and Workshop, 2007.

[Berkhin, 2006] P. Berkhin. A survey of clustering data mining techniques. In Jacob Kogan, Charles Nicholas, and Marc Teboulle, editors, Grouping Multidimensional Data, pages 25–71. Springer Berlin Heidelberg, 2006.

[Bezdek, 1981] James C. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[Billsus and Pazzani, 1998] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In Proceedings of the Fifteenth International Conference on Machine Learning, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[Billsus and Pazzani, 1999] Daniel Billsus and MichaelJ. Pazzani. A hybrid user model for news story classification. In Judy Kay, editor, UM99 User Modeling, volume 407 of CISM International Centre for Mechanical Sciences, pages 99–108. Springer Vienna, 1999.

[Billsus and Pazzani, 2000] Daniel Billsus and MichaelJ. Pazzani, (2000). User modeling for adaptive news access. User Modeling and User-Adapted Interaction, 10(2-3), pp.147–180.

[Billsus *et al.*, 2002] Daniel Billsus, Clifford A. Brunk, Craig Evans, Brian Gladish, and Michael Pazzani, (May 2002). Adaptive interfaces for ubiquitous web access. Commun. ACM, 45(5), pp.34–38.

[Bobadilla and Serradilla, 2009] Jesus Bobadilla and Francisco Serradilla. The effect of sparsity on collaborative filtering metrics. In Proceedings of the Twentieth Australasian Conference on Australasian Database - Volume 92, ADC '09, pages 9–18, Darlinghurst, Australia, Australia, 2009. Australian Computer Society, Inc.

[Bobadilla *et al.*, 2009] J. Bobadilla, F. Serradilla, and A. Hernando, (2009). Collaborative filtering adapted to recommender systems of e-learning. Knowledge-Based Systems, 22(4), pp.261 – 265. Artificial Intelligence (AI) in Blended Learning (AI) in Blended Learning.

[Bobadilla *et al.*, 2013] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez, (July 2013). Recommender systems survey. Know.-Based Syst., 46, pp.109–132.

[Boone, 1998] Gary Boone. Concept features in re :agent, an intelligent email agent. In Proceedings of the Second International Conference on Autonomous Agents, AGENTS '98, pages 141–148, New York, NY, USA, 1998. ACM.

[Bouza, 2010] Amancio Bouza. Mo - the movie ontology, 2010. [Online ; 26. Jan. 2010].

[Boyer *et al.*, 2008] Anne Boyer, Armel Brun, and Azim Roussanaly. Projet d'équipe kiwi (knowledge, information and web intelligence). Technical report, http ://kiwi.loria.fr/, 2008.

[Breese *et al.*, 1998] J.S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Proceeding of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI), pages 43–52, Morgan Kaufmann, San Francisco, 1998. Madison, Wisconsin.

[Bruno PRADEL, 2013] Bruno PRADEL. Évaluation des systèmes de recommandation à partir d'historiques de données. phdthesis, Université Pierre et Marie Curie, 2013.

[Brusilovsky, 2007] Peter Brusilovsky. Adaptive navigation support. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, The Adaptive Web, volume 4321 of Lecture Notes in Computer Science, pages 263–290. Springer Berlin Heidelberg, 2007.

[Buckley and Salton, 1995a] Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, SIGIR, pages 351–357. ACM Press, 1995.

[Buckley and Salton, 1995b] Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '95, pages 351–357, New York, NY, USA, 1995. ACM.

[Bunt *et al.*, 2007] Andrea Bunt, Giuseppe Carenini, and Cristina Conati. Adaptive content presentation for the web. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, The Adaptive Web, volume 4321 of Lecture Notes in Computer Science, pages 409–432. Springer Berlin Heidelberg, 2007.

[Burke, 1999] Robin Burke. The wasabi personal shopper : A case-based recommender system. In Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence, AAAI '99/IAAI '99, pages 844–849, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[Burke, 2000] Robin Burke, (2000). Knowledge-based recommender systems. Encyclopedia of Library and Information Systems, 69.

[Burke, 2002] Robin Burke, (2002). Hybrid recommender systems : Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), pp.331–370.

[Burke, 2007] Robin Burke. Hybrid web recommender systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, The Adaptive Web, volume 4321 of Lecture Notes in Computer Science, pages 377–408. Springer Berlin Heidelberg, 2007.

[Canny, 2002] John Canny. Collaborative filtering with privacy via factor analysis. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '02, pages 238–245, New York, NY, USA, 2002. ACM.

[Castagnos and Boyer, 2007] Sylvain Castagnos and Anne Boyer. Modeling preferences in a distributed recommender system. In Cristina Conati, Kathleen F. McCoy, and Georgios Paliouras, editors, User Modeling, volume 4511 of Lecture Notes in Computer Science, pages 400–404. Springer, 2007.

[Castagnos, 2008] Sylvain Castagnos. Mododélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information. PhD thesis, Université Nancy II, 2008.

[Chan, 2000] Philip K. Chan. Constructing web user profiles : A non-invasive learning approach. In Brij Masand and Myra Spiliopoulou, editors, Web Usage Analysis and User Profiling, number 1836 in Lecture Notes in Computer Science, pages 39–55. Springer Berlin Heidelberg, 2000.

[Chee *et al.*, 2001] SonnyHanSeng Chee, Jiawei Han, and Ke Wang. Rectree : An efficient collaborative filtering method. In Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa, editors, Data Warehousing and Knowledge Discovery, volume 2114 of Lecture Notes in Computer Science, pages 141–151. Springer Berlin Heidelberg, 2001.

[Chien and George, 1999] Y. H. Chien and E. I. George. A bayesian model for collaborative filtering. In In Proceedings of 7th International Workshop on Artificial Intelligence and Statistics, 1999.

[Choi *et al.*, 2012] Keunho Choi, Donghee Yoo, Gunwoo Kim, and Yongmoo Suh, (2012). A hybrid online-product recommendation system : Combining implicit rating-based collaborative filtering and sequential pattern analysis. Electronic Commerce Research and Applications, 11(4), pp.309 – 317.

[ChoiceStream, 2006] ChoiceStream. choicestream personalization survey. Technical report, 02 2006.

[Christakou and Stafylopatis, 2005] Christina Christakou and Andreas Stafylopatis. A hybrid movie recommender system based on neural networks. In Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, ISDA '05, pages 500–505, Washington, DC, USA, 2005. IEEE Computer Society.

[Claypool *et al.*, 1999] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In Proceedings of ACM SIGIR workshop on recommender systems, volume 60, 1999.

[Cleverdon and Kean, 1968] C. Cleverdon and M. Kean. Factors determining the performance of indexing systems. Aslib Cranfield Research Project, Cranfield, England, 1968.

[Cremonesi *et al.*, 2010] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[Cunningham *et al.*, 2001] Pádraig Cunningham, Ralph Bergmann, Sascha Schmitt, Ralph Traphöner, Sean Breen, and Barry Smyth, (2001). Websell : Intelligent sales assistants for the world wide web. KI, 15(1), pp.28–32.

[de Campos *et al.*, 2010] Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, and Miguel A. Rueda-Morales, (2010). Combining content-based and collaborative recommendations : A hybrid approach based on bayesian networks. International Journal of Approximate Reasoning, 51(7), pp.785–799.

[Degemmis *et al.*, 2007] Marco Degemmis, Pasquale Lops, and Giovanni Semeraro, (2007). A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. User Modeling and User-Adapted Interaction, 17(3), pp.217–255.

[Delgado and Ishii, 1999] J. Delgado and N. Ishii. Memory-based weighted majority prediction for recommender systems. In Proc. of the ACM SIGIR'99 Workshop on Recommender Systems, 1999.

[Deshpande and Karypis, 2004] Mukund Deshpande and George Karypis, (January 2004). Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst., 22(1), pp.143–177.

[Desrosiers and Karypis, 2011] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, Recommender Systems Handbook, pages 107–144. Springer US, 2011.

[Diday *et al.*, 1982] Edwin Diday, Jacques Lemaire, Jean Pouget, and Francoise. Eléments d'analyse de données. Dunod Informatique, Paris, France, 1982.

[Diederich and Iofciu, 2006] Jörg Diederich and Tereza Iofciu. Finding communities of practice from user profiles based on folksonomies. In In Proceedings of the 1st International Workshop on Building Technology Enhanced Learning solutions for Communities of Practice (TEL-CoPs'06), co-located with the First European Conference on Technology-Enhanced Learning, 2006.

[Dooms, 2013] Simon Dooms. Dynamic generation of personalized hybrid recommender systems. In Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pages 443–446, New York, NY, USA, 2013. ACM.

[Dumais, 2004] Susan T. Dumais, (2004). Latent semantic analysis. Annual Review of Information Science and Technology, 38(1), pp.188–230.

[Ekstrand *et al.*, 2011] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan, (February 2011). Collaborative filtering recommender systems. Found. Trends Hum.-Comput. Interact., 4(2), pp.81–173.

[ESWC, 2014] ESWC. Linked open data-enabled recommender system challenge., 2014. consulté, juillet 2014.

[Felfernig *et al.*, 2011] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker. Developing constraint-based recommenders. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, Recommender Systems Handbook, pages 187–215. Springer US, 2011.

[Frakes and Baeza-Yates, 1992] William B. Frakes and Ricardo Baeza-Yates, editors. Information Retrieval : Data Structures and Algorithms. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[Funk., 2006] S. Funk. Netfix update : Try this at home, sifter.org/simon/journal/20061211.html, 2006.

[Gajos *et al.*, 2008] Krzysztof Z. Gajos, Katherine Everitt, Desney S. Tan, Mary Czerwinski, and Daniel S. Weld. Predictability and accuracy in adaptive user interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08, pages 1271–1274, New York, NY, USA, 2008. ACM.

[Ghazanfar and Prugel-Bennett, 2010] M.A. Ghazanfar and A. Prugel-Bennett. A scalable, accurate hybrid recommender system. In Third International Conference on Knowledge Discovery and Data Mining, 2010. WKDD '10, pages 94–98, 2010.

[Ghosh and Dubey, 2013] Soumi Ghosh and Sanjay Kumar Dubey, (2013). Comparative analysis of k-means and fuzzy c-means algorithms. International Journal of Advanced Computer Science and Applications (IJACSA), 4(4), pp.35–38.

[Goldberg *et al.*, 1992] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry, (December 1992). Using collaborative filtering to weave an information tapestry. Commun. ACM, 35(12), pp.61–70.

[Goldberg *et al.*, 2001] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins, (2001). Eigentaste : A constant time collaborative filtering algorithm. Information Retrieval, 4(2), pp.133–151.

[Golub and F., 1996] Gene H. Golub and Van Loan Charles F. Matrix Computations. The Johns Hopkins University Press, 1996.

[Good *et al.*, 1999] Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. Combining collaborative filtering with personal agents for better recommendations. In Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence, AAAI '99/IAAI '99, pages 439–446, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[Gruber, 1995] Thomas R. Gruber, (1995). Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human Computer Studies, 43(5-6), pp.907–928.

[Heckerman *et al.*, 2001] David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie, (September 2001). Dependency networks for inference, collaborative filtering, and data visualization. J. Mach. Learn. Res., 1, pp.49–75.

[Herlocker *et al.*, 1999] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[Herlocker *et al.*, 2004] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl, (January 2004). Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst., 22(1), pp.5–53.

[HetRec2011, 2011] HetRec2011. In 2nd Int Workshop on Information Heterogeneity and Fusion in Recommender Systems. The 5th ACM Conf. RecSys, 2011.

[Heyer *et al.*, 1999] Laurie J. Heyer, Semyon Kruglyak, and Shibu Yooseph, (November 1999). Exploring expression data : Identification and analysis of coexpressed genes. Genome Research, 9(11), pp.1106–1115.

[Hill *et al.*, 1995] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[Hofmann, 1999] Thomas Hofmann. Probabilistic latent semantic indexing. In Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[Hofmann, 2003] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR '03, pages 259–266, New York, NY, USA, 2003. ACM.

[Hofmann, 2004] Thomas Hofmann, (January 2004). Latent semantic models for collaborative filtering. ACM Trans. Inf. Syst., 22(1), pp.89–115.

[Holmes *et al.*, 2007] Michael Holmes, Alexander Gray, and Charles Isbell. Fast SVD for large-scale matrices. In Workshop on Efficient Machine Learning at NIPS, 2007.

[IMDB, 2014] IMDB. www.imdb.com, 2014. consulté, février 2014.

[Jacquet and Drouot, 2007] H. Jacquet and L. Drouot. Rapport de synthèse du projet e-veille. Technical report, Erdyn consultants, 02 2007.

[Jain *et al.*, 1999] A. K. Jain, M. N. Murty, and P. J. Flynn, (September 1999). Data clustering : A review. ACM Comput. Surv., 31(3), pp.264–323.

[JeuMovieLens, 2014] JeuMovieLens. http ://grouplens.org/datasets/movielens/, 2014. consulté, février 2014.

[Jin *et al.*, 2004] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. Web usage mining based on probabilistic latent semantic analysis. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04, pages 197–205, New York, NY, USA, 2004. ACM.

[Johnson, 1967] StephenC. Johnson, (1967). Hierarchical clustering schemes. Psychometrika, 32(3), pp.241–254.

[Kamba *et al.*, 1995] Tomonari Kamba, Krishna A. Bharat, and Michael C. Albers. The krakatoa chronicle - an interactive, personalized NewsPaper on the web. In Proceedings of the Fourth International World Wide Web Conference, pages 159–170, 1995.

[Kamber, 2006] M. Kamber. Data Mining :Concepts and Techniques. Elsevier, 2006.

[Karypis, 2001] George Karypis. Evaluation of item-based top-n recommendation algorithms. In Proceedings of the Tenth International Conference on Information and Knowledge Management, CIKM '01, pages 247–254, New York, NY, USA, 2001. ACM.

[Kassab, 2009] Randa Kassab. Analyse des propriétés stationnaires et des propriétés émergentes dans les flux d'information changeant au cours du temps. PhD thesis, Université Nancy I, 2009.

[Kaufman and Rousseeuw, 2009] L Kaufman and PJ Rousseeuw. Finding groups in data : An introduction to cluster analysis. Wiley, 2009.

[Kim *et al.*, 2010] Heung-Nam Kim, Ae-Ttie Ji, Inay Ha, and Geun-Sik Jo, (2010). Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. Electronic Commerce Research and Applications, 9(1), pp.73 – 83. Special Issue : Social Networks and Web 2.0.

[Kitts *et al.*, 2000] Brendan Kitts, David Freed, and Martin Vrieze. Cross-sell : A fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '00, pages 437–446, New York, NY, USA, 2000. ACM.

[Kobsa *et al.*, 2001] Alfred Kobsa, Jürgen Koenemann, and Wolfgang Pohl, (March 2001). Personalised hypermedia presentation techniques for improving online customer relationships. Knowl. Eng. Rev., 16(2), pp.111–155.

[Konstan *et al.*, 1997] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl, (March 1997). Grouplens : Applying collaborative filtering to usenet news. Commun. ACM, 40(3), pp.77–87.

[Koren and Bell, 2011] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, Recommender Systems Handbook, pages 145–186. Springer US, 2011.

[Koren *et al.*, 2009] Y. Koren, R. Bell, and C. Volinsky, (Aug 2009). Matrix factorization techniques for recommender systems. Computer, 42(8), pp.30–37.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood : A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[Koren, 2009] Yehuda Koren. 1 the bellkor solution to the netflix grand prize, 2009.

[Koychev, 2000] Ivan Koychev. Gradual forgetting for adaptation to concept drift. In Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning, 2000.

[Krulwich, 1997] Bruce Krulwich, (1997). Lifestyle finder : Intelligent user profiling using large-scale demographic data. AI magazine, 18(2), pp.37.

[Lamprier, 2008] Sylvain Lamprier. Vers la conception de documents composites : extraction et organisation de l'information pertinente. PhD thesis, Université d'Angers, December 2008.

[Lang, 1995] Ken Lang. Newsweeder : Learning to filter netnews. In in Proceedings of the 12th International Machine Learning Conference (ML95, 1995.

[Lee *et al.*, 2002] Meehee Lee, Pyungseok Choi, and Yongtae Woo. A hybrid recommender system combining collaborative filtering with neural network. In Paul De Bra, Peter Brusilovsky, and Ricardo Conejo, editors, Adaptive Hypermedia and Adaptive Web-Based Systems, volume 2347 of Lecture Notes in Computer Science, pages 531–534. Springer Berlin Heidelberg, 2002.

[Lekakos and Caravelas, 2008] George Lekakos and Petros Caravelas, (2008). A hybrid approach for movie recommendation. Multimedia Tools and Applications, 36(1-2), pp.55–70.

[Lieberman, 1995] Henry Lieberman. Letizia : An agent that assists web browsing. In Proceedings of the International Joint Conference on Artificial Intelligence, page 924–929. Morgan Kaufmann Publishers Inc., 1995.

[Linden *et al.*, 2003] G. Linden, B. Smith, and J. York, (Jan 2003). Amazon.com recommendations : item-to-item collaborative filtering. Internet Computing, IEEE, 7(1), pp.76–80.

[Linqi and Li, 2008] Gao Linqi and Congdong Li. Hybrid personalized recommended model based on genetic algorithm. In Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on, pages 1–4, Oct 2008.

[Liu *et al.*, 2010] Zhaobin Liu, Wenyu Qu, Haitao Li, and Changsheng Xie, (2010). A hybrid collaborative filtering recommendation mechanism for {P2P} networks. Future Generation Computer Systems, 26(8), pp.1409 – 1417.

[Lops *et al.*, 2011] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems : State of the art and trends. In Recommender Systems Handbook, pages 73–105. Springer US, 2011.

[Lousame and Sánchez, 2009] Fabián P. Lousame and Eduardo Sánchez. A taxonomy of collaborative-based recommender systems. In Web Personalization in Intelligent Environments, page 81–117. Springer, 2009.

[Lü *et al.*, 2012] Linyuan Lü, Matus Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou, (10 2012). Recommender systems. Physics Reports, 519(1), pp.1–49.

[Luo *et al.*, 2012] Xin Luo, Yunni Xia, and Qingsheng Zhu, (2012). Incremental collaborative filtering recommender based on regularized matrix factorization. Knowledge-Based Systems, 27(0), pp.271 – 280.

[Luo *et al.*, 2013] Xin Luo, Yunni Xia, and Qingsheng Zhu, (2013). Applying the learning rate adaptation to the matrix factorization based collaborative filtering. Knowledge-Based Systems, 37(0), pp.154 – 164.

[MacQueen, 1967] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281–297. University of California Press, 1967.

[Maltz and Ehrlich, 1995] David Maltz and Kate Ehrlich. Pointing the way : Active collaborative filtering. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95, pages 202–209, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[Maneeroj and Takasu, 2009] S. Maneeroj and A. Takasu. Hybrid recommender system using latent features. In Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on, pages 661–666, May 2009.

[Manning *et al.*, 2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA, 2008.

[Manzato, 2012] Marcelo G. Manzato. Discovering latent factors from movies genres for enhanced recommendation. In Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, pages 249–252, New York, NY, USA, 2012. ACM.

[Marinho *et al.*, 2011] LeandroBalby Marinho, Alexandros Nanopoulos, Lars Schmidt-Thieme, Robert Jäschke, Andreas Hotho, Gerd Stumme, and Panagiotis Symeonidis. Social tagging recommender systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, Recommender Systems Handbook, pages 615–644. Springer US, 2011.

[Masthoff, 2011] Judith Masthoff. Group recommender systems : Combining individual models. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, Recommender Systems Handbook, pages 677–702. Springer US, 2011.

[McNally *et al.*, 2011] Kevin McNally, Michael P. O'Mahony, Maurice Coyle, Peter Briggs, and Barry Smyth, (October 2011). A case study of collaboration and reputation in social web search. ACM Trans. Intell. Syst. Technol., 3(1), pp.4 :1–4 :29.

[Melville *et al.*, 2002] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In Eighteenth National Conference on Artificial Intelligence, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[Micarelli *et al.*, 2007] Alessandro Micarelli, Fabio Gasparetti, Filippo Sciarrone, and Susan Gauch. Personalized search on the world wide web. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, The Adaptive Web, volume 4321 of Lecture Notes in Computer Science, pages 195–230. Springer Berlin Heidelberg, 2007.

[Middleton *et al.*, 2004a] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure, (2004). Ontological user profiling in recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), pp.54–88.

[Middleton *et al.*, 2004b] Stuart E. Middleton, Nigel R. Shadbolt, and David C. De Roure, (January 2004). Ontological user profiling in recommender systems. ACM Trans. Inf. Syst., 22(1), pp.54–88.

[Millen *et al.*, 2006] David R. Millen, Jonathan Feinberg, and Bernard Kerr. Dogear : Social bookmarking in the enterprise. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '06, pages 111–120, New York, NY, USA, 2006. ACM.

[Miller *et al.*, 2003] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged : Experiences with an occasionally connected recommender system. In Proceedings of the 8th International Conference on Intelligent User Interfaces, IUI '03, pages 263–266, New York, NY, USA, 2003. ACM.

[Mingoti and Lima, 2006] Sueli A. Mingoti and Joab O. Lima, (2006). Comparing {SOM} neural network with fuzzy c-means, k-means and traditional hierarchical clustering algorithms. European Journal of Operational Research, 174(3), pp.1742 – 1759.

[Mladenic, 1999] Dunja Mladenic. Machine learning used by personal webwatcher. In Proceedings of ACAI-99 Workshop on Machine Learning and Intelligent Agents, 1999.

[Mobasher and Nakagawa, 2003] Bamshad Mobasher and Miki Nakagawa. A hybrid web personalization model based on site connectivity. In Proceedings of the WebKDD Workshop at the ACM SIGKDD Intl Conf. on Knowledge Discovery and Data Mining, Washington, DC, 2003.

[Mobasher *et al.*, 2002] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa, (2002). Discovery and evaluation of aggregate usage profiles for web personalization. Data Mining and Knowledge Discovery, 6(1), pp.61–82.

[Mobasher *et al.*, 2004] Bamshad Mobasher, Xin Jin, and Yanzan Zhou. Semantically enhanced collaborative filtering on the web. In Bettina Berendt, Andreas Hotho, Dunja Mladenic,

Maarten van Someren, Myra Spiliopoulou, and Gerd Stumme, editors, Web Mining : FromWeb to SemanticWeb, volume 3209 of Lecture Notes in Computer Science, pages 57–76. Springer Berlin / Heidelberg, 2004.

[Mobasher *et al.*, 2006] Bamshad Mobasher, Robin Burke, and J. J. Sandvig. Model-based collaborative filtering as a defense against profile injection attacks. In Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2, AAAI'06, pages 1388–1393. AAAI Press, 2006.

[Mobasher, 2007] Bamshad Mobasher. Data mining for web personalization. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, The Adaptive Web, volume 4321 of Lecture Notes in Computer Science, pages 90–135. Springer Berlin Heidelberg, 2007.

[Montaner *et al.*, 2003] Miquel Montaner, Beatriz Lòpez, and JosepLluís de la Rosa, (2003). A taxonomy of recommender agents on the internet. Artificial Intelligence Review, 19(4), pp.285–330.

[Mooney and Roy, 2000] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In Proceedings of the Fifth ACM Conference on Digital Libraries, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.

[Moreno *et al.*, 2014] Andrés Moreno, Christian Ariza-Porras, Paula Lago, ClaudiaLucía Jiménez-Guarín, Harold Castro, and Michel Riveill. Hybrid model rating prediction with linked open data for recommender systems. In Valentina Presutti, Milan Stankovic, Erik Cambria, Iván Cantador, Angelo Di Iorio, Tommaso Di Noia, Christoph Lange, Diego Reforgiato Recupero, and Anna Tordai, editors, Semantic Web Evaluation Challenge, volume 475 of Communications in Computer and Information Science, pages 193–198. Springer International Publishing, 2014.

[MovieLens, 2014] MovieLens. www.movielens.org, 2014. consulté, février 2014.

[Nakamura, 1998] N. Nakamura, A.and Abe. Collaborative filtering using weighted majority prediction algorithms. In ICML '98 : Proc. of the 15th Int. Conf. on Machine Learning, pages 395–403, 1998.

[Narayanan and Shmatikov, 2008] A Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In Security and Privacy, 2008. SP 2008. IEEE Symposium on, pages 111–125, May 2008.

[Núñez-Valdéz *et al.*, 2012] Edward Rolando Núñez-Valdéz, Juan Manuel Cueva Lovelle, Oscar Sanjuán Martínez, Vicente García-Díaz, Patricia Ordoñez de Pablos, and Carlos Enrique Montenegro Marín, (2012). Implicit feedback techniques on recommender systems applied to electronic books. Computers in Human Behavior, 28(4), pp.1186 – 1193.

[Niwa *et al.*, 2006] Satoshi Niwa, Takuo Doi, and Shinichi Honiden. Web page recommender system based on folksonomy mining for itng '06 submissions. In Proceedings of the Third International Conference on Information Technology : New Generations, ITNG '06, pages 388–393, Washington, DC, USA, 2006. IEEE Computer Society.

[Papagelis *et al.*, 2005] Manos Papagelis, Dimitris Plexousakis, and Themistoklis Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, Trust Management, volume 3477 of Lecture Notes in Computer Science, pages 224–239. Springer Berlin Heidelberg, 2005.

[Park and Chang, 2009] You-Jin Park and Kun-Nyeong Chang, (2009). Individual and group behavior-based customer profile model for personalized product recommendation. Expert Systems with Applications, 36(2, Part 1), pp.1932 – 1939.

[Pazzani and Billsus, 1997] Michael Pazzani and Daniel Billsus, (1997). Learning and revising user profiles : The identification of interesting web sites. Machine Learning, 27(3), pp.313–331.

[Pazzani and Billsus, 2007] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, The Adaptive Web, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007.

[Pazzani *et al.*, 1996] Michael Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill &#38 ; webert : Identifying interesting web sites. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, volume 1 of AAAI'96, pages 54–61. AAAI Press, 1996.

[Pazzani, 1999] MichaelJ. Pazzani, (1999). A framework for collaborative, content-based and demographic filtering. Artificial Intelligence Review, 13(5-6), pp.393–408.

[Porcel *et al.*, 2009] C. Porcel, J.M. Moreno, and E. Herrera-Viedma, (2009). A multi-disciplinar recommender system to advice research resources in university digital libraries. Expert Systems with Applications, 36(10), pp.12520 – 12528.

[Porcel *et al.*, 2012] C. Porcel, A. Tejeda-Lorente, M.A. Martínez, and E. Herrera-Viedma, (2012). A hybrid recommender system for the selective dissemination of research resources in a technology transfer office. Information Sciences, 184(1), pp.1 – 19.

[Rasmussen, 1992] Edie M. Rasmussen. Clustering algorithms, pages 419–442. In Frakes and Baeza-Yates [1992], 1992.

[Ren *et al.*, 2008] Lei Ren, Liang He, Junzhong Gu, Weiwei Xia, and Faqing Wu. A hybrid recommender approach based on widrow-hoff learning. In Future Generation Communication and Networking, 2008. FGCN '08. Second International Conference on, volume 1, pages 40–45, Dec 2008.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens : an open architecture for collaborative filtering of netnews. In The 1994 ACM conference on Computer supported cooperative work, page 175–186, 1994.

[Ricci *et al.*, 2011a] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, Recommender Systems Handbook, pages 1–35. Springer US, 2011.

[Ricci *et al.*, 2011b] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. Recommender Systems Handbook. Springer US, 2011.

[Riordan and Sorensen, 1995] A. Riordan and H. Sorensen. An intelligent agent for high-precision information filtering. In In Proceedings of CIKM '95 Intelligent Information Agents Workshop, 1995.

[Rocchio, 1966] J.J. Rocchio. Document retrieval systems - Optimization and evaluation. PhD thesis, Harvard Computation Laboratory, 1966.

[Rocchio, 1971] J.J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, The Smart Retrieval System - Experiments in Automatic Document Processing, chapter 14, pages 313–323. Prentice-Hall, Inc, 1971.

[Rorvig, 1999] Mark Rorvig, (1999). Images of similarity : A visual exploration of optimal similarity metrics and scaling properties of trec topic-document sets. Journal of the American Society for Information Science, 50(8), pp.639–651.

[Salakhutdinov *et al.*, 2007] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In Proceedings of the 24th International Conference on Machine Learning, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.

[Salton and McGill, 1983] G. Salton and M. McGill. Introduction to Modern Information Retrieval. McGraw-Hill Publishing Company, New York, NY, USA, 1983.

[Salton and Wong, 1978] G. Salton and A. Wong, (December 1978). Generation and search of clustered files. ACM Trans. Database Syst., 3(4), pp.321–346.

[Salton, 1989] G. Salton. Automatic Text Processing. Addison-Wesley, 1989.

[Sarwar *et al.*, 2000a] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In Proceedings of the 2Nd ACM Conference on Electronic Commerce, EC '00, pages 158–167, New York, NY, USA, 2000. ACM.

[Sarwar *et al.*, 2000b] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. In Proceedings of the ACM WebKDD 2000 Web Mining for E-Commerce Workshop, 2000.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[Sarwar *et al.*, 2002] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In Fifth International Conference on Computer and Information Science, pages 27–28, 2002.

[Schafer *et al.*, 2002] J. Ben Schafer, Joseph A. Konstan, and John Riedl. Meta-recommendation systems : User-controlled integration of diverse recommendations. In Proceedings of the Eleventh International Conference on Information and Knowledge Management, CIKM '02, pages 43–51, New York, NY, USA, 2002. ACM.

[Schafer *et al.*, 2007] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, The Adaptive Web, number 4321 in Lecture Notes in Computer Science, pages 291–324. Springer Berlin Heidelberg, 2007.

[Schein *et al.*, 2002] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[Schickel and Faltings, 2006] Vincent Schickel and Boi Faltings. Using an ontologcial a-priori score to infer user's preferences. In Workshop on Recommender Systems, in Conjunction with the 17th European Conference on Artificial Intelligence (ECAI 2006), page 102–106, 2006.

[Schickel-Zuber, 2007] Vincent Schickel-Zuber. Ontology filtering. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2007.

[Schmitt and Bergmann, 1999] S. Schmitt and R. Bergmann. Applying case-based reasoning technology for product selection and customization in electronic commerce environments. In Proceeding of the 12th Bled Electronic Commerce Conf, Bled, Slovenia, june 1999.

[Schütze and Silverstein, 1997] Hinrich Schütze and Craig Silverstein, (July 1997). Projections for efficient document clustering. SIGIR Forum, 31(SI), pp.74–81.

[Semeraro *et al.*, 2009] Giovanni Semeraro, Pierpaolo Basile, Marco de Gemmis, and Pasquale Lops. User profiles for personalizing digital libraries. In Y.L. Theng, S. Foo, D.G.H. Lian, and J.C. Na, editors, Handbook of Research on Digital Libraries : Design, Development and Impact, pages 149–158. IGI Global, 2009.

[Sen *et al.*, 2009] Shilad Sen, Jesse Vig, and John Riedl. Tagommenders : Connecting users to items through tags. In Proceedings of the 18th International Conference on World Wide Web, WWW '09, pages 671–680, New York, NY, USA, 2009. ACM.

[Serrano-Guerrero *et al.*, 2011] Jesus Serrano-Guerrero, Enrique Herrera-Viedma, Jose A. Olivas, Andres Cerezo, and Francisco P. Romero, (2011). A google wave-based fuzzy recommender system to disseminate information in university digital libraries 2.0. Information Sciences, 181(9), pp.1503 – 1516.

[Shani and Gunawardana, 2011] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, Recommender Systems Handbook, pages 257–297. Springer US, 2011.

[Shani *et al.*, 2005] Guy Shani, David Heckerman, and Ronen I. Brafman, (December 2005). An mdp-based recommender system. J. Mach. Learn. Res., 6, pp.1265–1295.

[Shardanand and Maes, 1995] Upendra Shardanand and Pattie Maes. Social information filtering : Algorithms for automating “word of mouth”. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[Shepitsen *et al.*, 2008] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys '08, pages 259–266, New York, NY, USA, 2008. ACM.

[Sheth and Maes, 1993] B. Sheth and P. Maes. Evolving agents for personalized information filtering. In Proceedings of the Ninth Conference on Artificial Intelligence for Applications, pages 345–352. IEEE, Computer Society Press, 1993.

[Shinde and Kulkarni, 2012] Subhash K. Shinde and Uday Kulkarni, (2012). Hybrid personalized recommender system using centering-bunching based clustering algorithm. Expert Systems with Applications, 39(1), pp.1381 – 1387.

[Smyth and Cotter, 2001] Barry Smyth and Paul Cotter, (2001). Personalized electronic program guides for digital tv. AI MAGAZINE, 22(1), pp.89–98.

[Smyth *et al.*, 2005] Barry Smyth, Evelyn Balfe, Oisin Boydell, Keith Bradley, Peter Briggs, Maurice Coyle, and Jill Freyne. A live-user evaluation of collaborative web search. In Proceedings of IJCAI 2005, the 19th International Joint Conference on Artificial Intelligence, volume 5, page 1419–1424, 2005.

[Smyth, 2007] Barry Smyth. Case-based recommendation. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, The Adaptive Web, volume 4321 of Lecture Notes in Computer Science, pages 342–376. Springer, 2007.

[Sorensen and McElligott, 1995] H. Sorensen and M. McElligott. Psun : A profiling system for usenet news. In In Proceedings of CIKM '95 Intelligent Information Agents Workshop, 1995.

[Sorensen and McElligott, 1998] H. Sorensen and M. McElligott. Clustering methods for collaborative filtering. In In Proceedings of the 1998 Workshop on Recommender Systems, Menlo Park, California, 1998. AAAI Press.

[Sorensen *et al.*, 1997] Humphrey Sorensen, Adrian O'Riordan, and Colm O'Riordan, (août 1997). Profiling with the informer text filtering agent. Journal of Universal Computer Science, 3(8), pp.988–1006.

[Stefani and Strapparava, 1998] Anna Stefani and Carlo Strapparava. Personalizing access to web sites : The siteif project. In In Proceedings of second Workshop on Adaptive Hypertext and Hypermedia, Pittsburgh, June 1998.

[Su and Khoshgoftaar, 2009] Xiaoyuan Su and Taghi M. Khoshgoftaar, (January 2009). A survey of collaborative filtering techniques. Adv. in Artif. Intell., 2009, pp.4 :2–4 :2.

[Symeonidis *et al.*, 2007] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Feature-weighted user model for recommender systems. In Cristina Conati, Kathleen McCoy, and Georgios Paliouras, editors, User Modeling 2007, volume 4511 of Lecture Notes in Computer Science, pages 97–106. Springer Berlin Heidelberg, 2007.

[Takács *et al.*, 2007] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk, (December 2007). Major components of the gravity recommendation system. SIGKDD Explor. Newsl., 9(2), pp.80–83.

[Takacs *et al.*, 2008] G. Takacs, I. Pilaszy, B. Nemeth, and Domonkos Tikk. Investigation of various matrix factorization methods for large recommender systems. In Data Mining Workshops, 2008. ICDMW '08. IEEE International Conference on, pages 553–562, Dec 2008.

[Takács *et al.*, 2009] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk, (June 2009). Scalable collaborative filtering approaches for large recommender systems. J. Mach. Learn. Res., 10, pp.623–656.

[Terveen *et al.*, 1997] Loren Terveen, Will Hill, Brian Amento, David McDonald, and Josh Creter, (March 1997). Phoaks : A system for sharing recommendations. Commun. ACM, 40(3), pp.59–62.

[Tintarev and Masthoff, 2011] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, Recommender Systems Handbook, pages 479–510. Springer US, 2011.

[Tufféry, 2005] Stephane Tufféry. Data mining et statistique décisionnelle : l'intelligence dans les bases de données. Technip, 2005.

[Verma *et al.*, 2013] S.K. Verma, N. Mittal, and B. Agarwal. Hybrid recommender system based on fuzzy clustering and collaborative filtering. In Computer and Communication Technology (ICCCT), 2013 4th International Conference on, pages 116–120, Sept 2013.

[Webb and Kuzmycz, 1995] GeoffreyI. Webb and Mark Kuzmycz, (1995). Feature based modelling : A methodology for producing coherent, consistent, dynamically changing models of agents' competencies. User Modeling and User-Adapted Interaction, 5(2), pp.117–150.

[Willett, 1983] Peter Willett, (1983). Similarity coefficients and weighting functions for automatic document classification : an empirical comparison. International Classification, (3), pp.138–142.

[Willett, 1988] Peter Willett, (1988). Recent trends in hierarchic document clustering : A critical review. Information Processing & Management, 24(5), pp.577 – 597.

[Xu and Wunsch, 2005] Rui Xu and Donald C. Wunsch, (May 2005). Survey of clustering algorithms. Neural Networks, IEEE Transactions on, 16(3), pp.645–678.

[Zaiane, 2002] O.R. Zaiane. Building a recommender agent for e-learning systems. In Computers in Education, 2002. Proceedings. International Conference on, pages 55–59 vol.1, Dec 2002.

[Zamir and Etzioni, 1998] Oren Zamir and Oren Etzioni. Web document clustering : A feasibility demonstration. In Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98, pages 46–54, New York, NY, USA, 1998. ACM.

[Zhang, 2009] Mi Zhang. Enhancing diversity in top-n recommendation. In Proceedings of the Third ACM Conference on Recommender Systems, RecSys '09, pages 397–400, New York, NY, USA, 2009. ACM.

[Ziegler *et al.*, 2005] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In Proceedings of the 14th International Conference on World Wide Web, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.

[Zigoris and Zhang, 2006] Philip Zigoris and Yi Zhang. Bayesian adaptive user profiling with explicit & implicit feedback. In Proceedings of the 15th ACM International Conference on Information and Knowledge Management, CIKM '06, pages 397–404, New York, NY, USA, 2006. ACM.

Résumé

Face à la surabondance des ressources et de l'information sur le net, l'accès aux ressources pertinentes devient une tâche fastidieuse pour les usagers de la toile. Les systèmes de recommandation personnalisée comptent parmi les principales solutions qui assistent l'utilisateur en filtrant les ressources, à priori à partir de l'observation de son comportement, pour ne lui proposer que celles susceptibles de l'intéresser. L'approche basée sur l'observation du comportement de l'utilisateur à partir de ses interactions avec le e-service est appelée analyse des usages. Le filtrage collaboratif et le filtrage basé sur le contenu sont les principales techniques de recommandations personnalisées. Le filtrage collaboratif exploite uniquement les données issues de l'analyse des usages pour construire le profil des utilisateurs alors que le filtrage basé sur le contenu utilise en plus les données décrivant le contenu des ressources. Un système de recommandation hybride combine les deux techniques de recommandation. L'objectif de cette thèse est de proposer une autre technique d'hybridation en étudiant les bénéfices de l'exploitation combinée d'une part, des informations sémantiques des ressources à recommander, avec d'autre part, le filtrage collaboratif qui tire profit du comportement de la communauté des usagers du e-service. Plusieurs approches ont été proposées pour l'apprentissage d'un nouveau profil utilisateur inférant ses préférences pour l'information sémantique décrivant les ressources. Pour chaque approche proposée, nous traitons le problème du manque de la densité des données et le problème de la massivité des données. Nous montrons également, de façon empirique, un gain au niveau de la précision des recommandations par rapport à des approches purement collaboratives ou basées sur le contenu.

Mots-clés: recommandation personnalisée, filtrage collaboratif, contenu des ressources, profil sémantique de l'utilisateur.

Abstract

Face to the ongoing rapid expansion of the Internet, user requires help to access to items that may interest her or him. A personalized recommender system filters relevant items from huge catalogue to particular user by observing his or her behavior. The approach based on observing user behavior from his interactions with the e-service is called usage analysis. Collaborative Filtering and Content-Based filtering are the most widely used techniques in personalized recommender system. Collaborative filtering uses only data from usage analysis to build user profile, while content-based filtering relies in addition on semantic information of items. Hybrid approach is another important technique, which combines collaborative and content-based methods to provide recommendations. The aim of this thesis is to present a new hybridization approach that takes into account the semantic information of items to enhance collaborative recommendations. Several approaches have been proposed for learning a new user profile inferring preferences for semantic information describing items. For each proposed approach, we address the sparsity and the scalability problems. We prove also, empirically, an improvement in recommendations accuracy against collaborative filtering and content-based filtering.

Keywords: personalized recommendation, collaborative filtering, item content, user semantic profile.