THESE POUR OBTENIR LE GRADE DE DOCTEUR
ÉCOLE NATIONALE SUPÉRIEURE DES MINES D'ALÈS (IMT MINES ALÈS)

En Informatique

École doctorale I2S – Information, Structures, Systèmes
Portée par l'Université de Montpellier

Unité de recherche EuroMov Digital Health in Motion

Des données aux connaissances : vers des recommandations plus pertinentes, diversifiées et transparentes

Présentée par Yu DU
le 3 Décembre 2021

Sous la direction de Sylvie RANWEZ et Vincent RANWEZ

Devant le jury composé de

Catherine FARON ZUCKER, Maître de Conférences (HDR), Université Côte d'Azur	Rapporteur
Marie-Hélène ABEL, Professeur des Universités, Université de Technologie de Compiègne (UTC)	Rapporteur
Sylvie CALABRETTO, Professeur des Universités, Institut National des Sciences Appliquées (INSA Lyon)	Examinatrice
Sandra BRINGAY, Professeur des Universités, Université Paul Valéry Montpellier	Présidente du jury
Raphaël TRONCY, Maître de conférences, EURECOM, Sophia Antipolis	Examinateur
Sylvie RANWEZ, Professeur, IMT Mines Alès	Co-directrice de thèse
Vincent RANWEZ, Professeur, Institut Agro (Montpellier SupAgro)	Co-directeur de thèse
Nicolas SUTTON-CHARANI, Maître-assistant, IMT Mines Alès	Encadrant de proximité

COLLÈGE DOCTORAL
UNIVERSITÉ DE MONTPELLIER

IMT Mines Alès
École Mines-Télécom

Résumé

Dans le contexte actuel de surcharge causée par l'important volume de données numériques accessibles, les systèmes de recommandation permettent de guider l'utilisateur dans ses activités d'apprentissages, d'achats, de loisir, d'écoute musicale, de lectures..., en lui suggérant des *items* personnalisés. Pour cela, ils prédisent ses préférences relativement aux items qu'il n'a pas encore évalués. Des approches classiques de recommandation, comme le filtrage collaboratif par exemple, reposent sur les *données* collectées par le biais de retours d'utilisateurs, généralement sous la forme d'une matrice de notes, et tentent d'y découvrir les informations pertinentes pour caractériser et prédire les goûts des utilisateurs. En complément de ces *données*, les *connaissances* liées aux items eux-mêmes représentent également un atout majeur pour l'amélioration des performances des systèmes de recommandation. L'ingénierie des connaissances, plus spécifiquement le Web sémantique et les graphes de connaissances, peuvent y jouer un rôle central. Tel est le cadre de notre travail de recherche qui propose différentes voies d'amélioration des systèmes de recommandation, adoptant une vision transversale « des *données* aux *connaissances* », et ce sur trois aspects différents : la *pertinence (accuracy)*, la *diversification* et l'*explicabilité* des recommandations.

Notre première contribution est principalement axée sur les *données*. Elle concerne l'*accuracy des recommandations* en termes de prédiction des goûts des utilisateurs. Nous proposons EBCR (*Empirical Bayes Concordance Ratio*), une méthode simple et générique inspirée de l'inférence bayésienne, qui permet d'ajuster les calculs de similarité entre utilisateurs (ou entre items) mis en œuvre dans le filtrage collaboratif. Cet ajustement est réalisé en fonction du nombre d'items co-notés (ou du nombre d'utilisateurs ayant noté un même item). Les expériences menées sur des jeux de données de référence ont confirmé que cette méthode améliore systématiquement l'*accuracy* du filtrage collaboratif pour toutes les mesures de similarité considérées.

Notre deuxième contribution concerne la *diversification des recommandations*. Nous avons mené une étude approfondie visant à comparer et analyser la performance de sept modèles de recommandation incluant des modèles classiques comme le filtrage collaboratif et le facteur latent ainsi que ceux, plus récents, se basant sur les réseaux de neurones profonds ou les plongements (*embeddings*) de graphe de connaissances. Nous avons évalué leur capacité à fournir des items diversifiés et proposé une approche qui permet d'ajuster la diversité aux besoins spécifiques des utilisateurs. Afin d'estimer la diversité des recommandations, nous avons considéré des mesures de similarité sémantique en tirant parti des *connaissances* liées aux items à l'aide du Web sémantique.

Enfin, notre troisième contribution concerne l'*explicabilité des recommandations*. Ici, nous exploitons plus en profondeur les connaissances du domaine, en proposant une

approche de l'explication *post-hoc* des recommandations qui considère efficacement la hiérarchie des concepts au sein du graphe de connaissances de DBpedia. Les résultats de l'évaluation de notre approche basée sur une étude comprenant 155 participants suggèrent des améliorations significatives en termes d'*engagement*, de *confiance* et de *persuasion*.

Mots-clés : Système de recommandation, Filtrage collaboratif, Calibration, Diversité, Explicabilité, Ingénierie des connaissances, Graphe de connaissances, Plongement de graphe, DBpedia

Abstract

In the current information overload context caused by the large volume of accessible digital data, recommender systems allow to guide the user in his/her learning, shopping, leisure, music listening, reading activities..., by suggesting personalized items. To do so, recommendation models predict users' preferences for their unrated items. Classical recommendation approaches, such as collaborative filtering, for example, rely on data collected through user feedback, usually in the form of a rating matrix, and try to discover relevant information to characterize and predict user tastes. In addition to the user feedback *data*, the *knowledge* related to items themselves also represents a major asset for improving the performance of recommendation systems. Knowledge engineering, more specifically the semantic Web and knowledge graphs, can play a central role. In this context, our research works propose different ways to improve recommendation systems, adopting a "from-data-to-knowledge" transversal vision, and consider three different recommendation aspects: *accuracy*, *diversification* and *explicability*.

Our first contribution is mainly focused on pure user feedback data. It aims at improving the *accuracy of recommendations* in terms of the prediction of users' tastes. We propose EBCR (Empirical Bayes Concordance Ratio), a simple and generic method inspired by Bayesian inference, which allows to adjust the similarity computations between users (or between items) in collaborative filtering, according to the number of co-rated items (or the number of users having rated the same item). Experiments conducted on benchmark datasets have confirmed that this method systematically improves the predictive accuracy of collaborative filtering for all considered similarity measures.

Our second contribution concerns the *diversification of recommendations*. We have conducted an in-depth study to compare and analyze the performance of seven recommendation models including classical models such as collaborative filtering and latent factor models as well as more recent ones based on deep neural networks and knowledge graph embeddings. We have evaluated their ability to provide diversified items and proposed an approach that allows adjusting diversity to specific user needs. In order to estimate the diversity of recommendations, we considered semantic similarity measures by leveraging the semantic Web and knowledge graphs.

Finally, our third contribution concerns the *explicability of recommendations*. Here, we further exploit domain knowledge and propose a post-hoc recommendation explanation approach that effectively accounts for the hierarchy of item properties within the DBpedia knowledge graph. Evaluation results of our approach based on an online user study including 155 participants suggest significant improvements in terms of *engagement*, *trust* and *persuasion*.

Keywords: Recommender system, Collaborative filtering, Calibration, Diversity, Explicability, Knowledge engineering, Knowledge graph, Graph embedding, DBpedia

Remerciements

Pas comme chaque thèse, ces trois années ont témoigné les expériences qui sont particulièrement impactées par la pandémie lié au COVID-19, bouleversant la vie et l'habitude de chacune et de chacun.

Heureusement comme chaque thèse, ces trois années passées sont, avant tout, une expérience très riche dans laquelle il y a eu de la *joie*, de l'*émoi*, de l'*inquiétude*, de la *confusion*, du *challenge* et surtout, de la *reconnaissance*.

J'exprime vivement mes reconnaissances ici et remercie sincèrement toutes les personnes m'ayant accompagnées durant la période de ma thèse.

En premier lieu, je tiens à remercier ma directrice de thèse Sylvie Ranwez, qui m'a encadré tout au long de ma thèse avec sa gentillesses, sa patience et sa responsabilité. Je le remercie de m'avoir fait confiance et donné des conseils utiles à chaque fois où j'ai proposé de m'orienter autrement durant ces années. Je le remercie pour sa présence permanente et toutes ses disponibilités consacrées pour l'encadrement de ma thèse. Je le remercie également pour son encouragement pendant les moments les plus durs à cause des confinements.

Je tiens à remercier ensuite Vincent Ranwez, le co-directeur de ma thèse et Nicolas Sutton-Charani, l'encadrant de ma thèse. Je remercie leurs accompagnements de près pour le bon déroulement scientifique de ma thèse. Je les remercie également pour leurs conseils et propositions utiles, leurs aides pratiques et leurs présences permanentes.

J'adresse tous mes remerciements à Madame Catherine Faron Zucker, Maître de conférences à l'Université Côte d'Azur, ainsi qu'à Madame Marie-Hélène Abel, Professeur à l'Université de Technologie de Compiègne, de l'honneur qu'elles m'ont fait en acceptant d'être rapporteurs de cette thèse.

J'exprime ma gratitude à Madame Sylvie Calabretto, Professeur à l'Institut National des Sciences Appliquées, à Madame Sandra Bringay, Professeur à l'Université Paul-Valéry Montpellier 3 et à Monsieur Raphaël Troncy, Maître de conférences à l'École d'Ingénieur et Centre de recherche en Sciences du Numérique, qui ont bien voulu être examinateurs pour ce jury.

Ensuite, je tiens à remercier l'ensemble des personnes de l'école et de laboratoire CERIS qui m'ont accompagné et ont répondu avec calme et patience aux questions quotidiennes durant ma thèse. Particulièrement, j'adresse mes remerciements à Jacky Montmain et Edith Teychene, pour les temps qu'ils ont consacré pour assurer le bon déroule-

ment de ma thèse en termes de l'administration ; à Pierre Jean, pour ses aides techniques relatives aux serveur de calcul et déploiement des applications.

Enfin, je tiens à remercier les doctorants de laboratoire qui m'ont accompagné pendant ma thèse et partagé leurs vies et expériences.

« *Faire une thèse est une adventure spéciale et unique de ma vie, c'est grâce à ces personnes qui rendent ma thèse spéciale et unique.* »

Table des matières

Table des matières iii

Liste des figures vi

Liste des tableaux ix

Glossaire x

Liste des notations xi

1 Introduction 1

1.1 Contexte général . . . 1
1.2 Challenges de recherche et contributions . . . 2
1.3 Structure du manuscrit . . . 6

2 État de l'art 8

2.1 Systèmes de recommandation (SdR) . . . 9
2.1.1 Contexte et problématique . . . 9
2.1.2 Définition et catégorisation des systèmes de recommandation . . . 10
2.1.3 Approches basées sur le filtrage collaboratif . . . 13
2.1.4 Approches basées sur le contenu . . . 29
2.1.5 Approches hybrides . . . 37
2.1.6 Évaluation des systèmes de recommandation . . . 39
2.2 Apports de l'ingénierie des connaissances aux SdR . . . 47
2.2.1 Apports des ontologies . . . 48
2.2.2 Apports des graphes de connaissances . . . 56
2.3 Conclusion de chapitre . . . 64

3 Recommandation plus pertinente ? Ajustement bayésien empirique des mesures de similarité pour améliorer les recommandations du filtrage collaboratif 66

3.1 Problématique . . . 67
3.2 Contribution en termes de mesure de la similarité : EBCR . . . 70
3.2.1 Prise en compte des spécificités des utilisateurs par un ratio de concordance : CR . . . 70

3.2.2 Ajustement du ratio de concordance via le modèle bayésien empirique : EB . . . 72
3.3 Évaluation . . . 77
3.3.1 Approches comparées . . . 77
3.3.2 Jeux de données . . . 77
3.3.3 Métriques d'évaluation . . . 78
3.3.4 Protocole expérimental . . . 78
3.4 Résultats et discussions . . . 82
3.5 Exemple d'une étude de cas . . . 87
3.6 Conclusion de chapitre . . . 88

4 Recommandation plus diversifiée ? Vers une meilleure compréhension de la diversité dans les approches de recommandation 90
4.1 Motivation . . . 91
4.1.1 Problématique et questions de recherche . . . 93
4.1.2 Solutions proposées . . . 93
4.2 Diversité des recommandations . . . 94
4.2.1 Définition et métrique d'évaluation . . . 94
4.2.2 Différentes fonctions objectifs pour la diversification . . . 98
4.2.3 Heuristique gloutonne d'optimisation . . . 100
4.2.4 Autres approches de diversification . . . 101
4.3 Protocole d'évaluation . . . 102
4.3.1 Les systèmes de recommandation considérés . . . 102
4.3.2 Jeux de données & pré-traitement . . . 105
4.3.3 Construction des graphes de connaissances . . . 107
4.3.4 Construction des modèles de recommandation . . . 107
4.3.5 Métriques d'évaluation . . . 111
4.3.6 Configuration choisie pour l'étude de la diversification . . . 113
4.4 Résultats et discussions . . . 113
4.4.1 Performances en termes d'*accuracy* et de diversité des SdR testés . . . 114
4.4.2 Impact du post-traitement de la diversification . . . 117
4.4.3 Impact du nombre d'items candidats du post-traitement glouton sur la diversité et l'*accuracy* . . . 120
4.4.4 Limitation, discussion et implication . . . 122
4.5 Conclusion de chapitre . . . 124

5 Recommandation plus transparente ? Mieux comprendre la recommandation à l'aide des graphes de connaissances 127
5.1 Introduction et contexte . . . 128
5.1.1 Explications basées sur les modèles . . . 128
5.1.2 Explications indépendantes des modèles (*post-hoc*) . . . 130

5.1.3 Explications post-hoc basées sur les données liées 131
5.2 Motivations 134
5.3 Modèle d'explication basé sur les propriétés (PEM) 136
5.3.1 Définitions et notations 137
5.3.2 Fonction de score des propriétés 137
5.3.3 Calcul efficace des scores à l'aide de l'extraction de sous-ontologie . 138
5.3.4 Éviter la redondance totale entre les propriétés sélectionnées 141
5.3.5 La méthode PEM 142
5.4 Protocole d'évaluation 143
5.4.1 Jeu de données *MovieTweetings* et alignements dans DBpedia 143
5.4.2 Modèle de recommandation 144
5.4.3 Métriques d'évaluation 145
5.4.4 Étude réalisée avec des utilisateurs 146
5.5 Résultats 147
5.5.1 Efficacité de l'optimisation de la sous-ontologie 147
5.5.2 Comparaison des performances sur l'étude d'utilisateurs 148
5.5.3 Un exemple d'étude de cas 149
5.6 Conclusion de chapitre 150

6 Conclusions et perspectives **152**
6.1 Conclusions 152
6.2 Perspectives de travaux futurs 156

A **I**

Liste des figures

1.1 Apports des connaissances du domaine aux différentes phases de recommandation 4
1.2 Plan du manuscrit et mots-clés des chapitres 6

2.1 Un exemple du *feedback* explicite sous l'échelle graduée de 1 à 5. (Source Amazon) 11
2.2 Exemples de matrices de *ratings* pour des *feedbacks* explicites (a) et implicites (b) 12
2.3 Exemple de description de contenu des iPhone sur Amazon 13
2.4 Exemple d'une matrice des *ratings* pour illustrer l'approche NBCF. La case verte représente le *rating* dont on cherche à prédire la valeur. 18
2.5 Exemples de transformation d'une matrice de *ratings* explicites (a) en matrice implicite normalisée (b). 25
2.6 Architecture générale du modèle NeuMF. Source : [Zhang et al., 2019] 27
2.7 Architecture complet du modèle NeuMF. Source : [He et al., 2017] 29
2.8 Architecture générale des approches de recommandation basées sur le contenu 31
2.9 Nuage des données liées (*Linked Open Data*) en 2020. Chaque bulle représente un *triplet store* (une base de connaissances spécifique) et les liens noirs représentent leur interconnexion. La flèche noire indique DBpedia (placé au centre car les connaissances qu'il contient sont communes à plusieurs domaines et sont considérées comme données de références pour de nombreuses autres bases), la flèche rouge indique Geonames et la flèche blanche Musicbrainz. 58
2.10 Exemple d'un graphe orienté construit à partir des éléments au sein d'un système de recommandation 60
2.11 Illustration du modèle d'*embeddings TransE* 62
2.12 La recommandation vue comme un problème de complétion du graphe de connaissances 63

3.1 Exemple des distributions de *Beta* avec des hyper-paramètres différents (Source : Wikipedia) 74

3.2 Performances prédictives (MAE) de la méthode EBCR pour différentes valeurs du paramètre a (cf. équation (3.3)) et différentes tailles de voisinage (k) 81
3.3 Comparaison des résultats obtenus en termes de MAE et RMSE en fonction de la taille du voisinage en utilisant respectivement les mesures de similarité usuelles (courbes en lignes pointillés) et leurs variantes intégrant les ratios EBCR (courbes en lignes pleines de même couleur), sur trois jeux de données de référence : MovieLens-100K, MovieLens-1M et Jester. 83
3.4 Comparaison des résultats obtenus en termes de MAE et RMSE en fonction de la taille du voisinage en utilisant respectivement l'approche de *significance weighting* (SW), l'approche de *multi-level collaborative filtering* (MLCF) et l'approche EBCR sur trois jeux de données de référence : MovieLens-100K, MovieLens-1M et Jester. 84

4.1 Procédure de construction des SdR pour : A) les SdR ne disposant pas d'hyper-paramètres et B) les SdR disposant des hyper-paramètres. 110
4.2 Distributions des valeurs de diversité (ILD) fournies aux utilisateurs par différents SdR et des valeurs de diversités présentes dans les profils utilisateurs (ILD_p) 117
4.3 Diagramme de dispersion des diversités au sein des recommandations par rapport aux diversités dans les profils utilisateurs (*i.e.*, ILD par rapport à ILD_p). Les recommandations sont basées sur le modèle KGE. 117
4.4 Évaluation de la diversité *absolue* pour les fonctions objectifs classique et personnalisée dans le post-traitement glouton. Les lignes en pointillés rouges représentent la moyenne des diversités *désirées* par les utilisateurs, $\overline{\text{ILD}_p}$. . 118
4.5 Évaluation de la diversité *relative* pour les fonctions objectifs classique et personnalisée dans le post-traitement glouton. 119
4.6 Diagramme de dispersion des valeurs d'ILD (diversité recommandée) par rapport à celles d'ILD_p (diversité du profil) pour trois valeurs de α avec deux fonctions objectifs différentes. Les recommandations sont basées sur le modèle KGE. 119
4.7 Évaluation de l'*accuracy* pour les fonctions objectifs classique et personnalisée dans le post-traitement glouton. 120
4.8 *Accuracy* (F1-Mesure) et diversité absolue ($\overline{\text{ILD}}$) des SdR sur MovieLens pour différentes valeurs du nombre d'items candidats m considérés lors des post-traitements gloutons. Le facteur de diversification α est fixé à 0.5 pour les deux fonctions objectifs. 121

5.1 Exemples illustrant l'importance de considérer la hiérarchie des propriétés dans le processus d'explication afin d'identifier les relations sémantiques entre les propriétés qui annotent des items du profil de l'utilisateur et celles qui annotent les items qui lui sont recommandés. 135
5.2 Exemple *jouet* illustrant le processus d'extraction de sous-ontologie : A) fournit une mini hiérarchie de propriétés (inspirée de DBpedia) ; B) fournit la sous-ontologie associée et C) un post-ordonnancement des propriétés de cette dernière, qui est au centre de nos optimisations algorithmiques. 139
5.3 Un exemple d'étude de cas comparant la variante *broader* de l'approche ExpLOD A) et PEM B). Les films du profil utilisateur sont représentés par des symboles rouges, jaunes et orange, tandis que l'étoile verte représente le film recommandé à expliquer. Les symboles géométriques blancs représentent les descriptions indirectes des films induites par la hiérarchie des propriétés. Les trois propriétés les mieux classées pour chaque approche sont mises en évidence par la couleur verte. 150

Liste des tableaux

3.1 Caractéristiques des jeux de données pour l'évaluation de l'approche EBCR 78
3.2 Résultats comparatifs entre les approches *Laplace Smoothing* (LS) et EBCR sur trois jeux de données de référence (les meilleurs prédictions sont mises en gras). 85
3.3 Comparaison des MAE et RMSE obtenus par les approches du filtrage collaboratif basé sur les modèles et EBCR, respectivement sur 3 jeux de données (les meilleures valeurs de MAE et RMSE pour chaque jeu de données sont indiquées en gras et les secondes sont soulignées). 85
3.4 Comparaison entre les résultats obtenus par des approches classiques de filtrage collaboratif (*i.e. baseline*, SVD, SVD++, EBCR) et l'approche basée sur les réseaux de neurones profonds (NeuMF), dans le contexte des top-N recommandations. 87
3.5 Exemple d'une étude de cas extrait du jeu de données MovieLens-100K. . . 88

4.1 Principales caractéristiques des jeux de données (filtrés) 107
4.2 Propriétés de l'ontologie DBpedia prises en compte pour chaque jeu de données 107
4.3 Nombre de triplets composant les deux graphes de connaissances construits pour chaque jeu de données. 107
4.4 Récapitulation des SdR testés 108
4.5 Comparaison des performances des SdR en termes de l'*accuracy* 115
4.6 Comparaison des performances des SdR en termes de la diversité. 116

5.1 Détails du Questionnaire 146
5.2 Résultats de l'étude utilisateur. Les meilleures mesures sont en gras et les améliorations statistiquement significatives sont indiquées par (*). 149

A.1 Espaces de recherche des hyper-paramètres des modèles de recommandation comparés pendant le processus d'optimisation bayésienne I
A.2 Les configurations des hyper-paramètres optimaux trouvées pour chaque modèle de recommandation sur chaque jeu de données II

Glossaire

CBF filtrage basé sur le contenu – ou *Content-Based Filtering*.

CF filtrage collaboratif – ou *Collaborative Filtering*.

COS similarité cosinus.

CR ratio de concordance – ou *Concordance Ratio*.

DAG graphe orienté acyclique – ou *Directed Acyclic Graph*.

EB bayésien empirique – ou *Empirical Bayes*.

EBCR ratio de concordance ajusté de manière bayésienne empirique – ou *Empirical Bayes Concordance Ratio*.

IC contenu informationnel – ou *Information Content*.

ILD diversité intra-liste – ou *Intra-List-Diversity*.

KG graphe de connaissances – ou *Knowledge Graph*.

KGE plongement de graphes de connaissances – ou *Knowledge Graph Embedding*.

LOD données liées – ou *Linked Open Data*.

MSD distance quadratique moyenne – ou *Mean Squared Distance*.

NBCF filtrage collaboratif basé sur le voisinage – ou *Neighborhood-Based Collaborative Filtering*.

PCC coefficient de corrélation de Pearson – ou *Pearson Correlation Coefficient*.

SdR système de recommandation.

SGD descente de gradient stochastique – ou *Stochastic Gradient Descent*.

SVM machines à vecteurs de support – ou *Support Vector Machine*.

SW facteur de pondération significative – ou *Significance Weighting*.

Liste des notations

U ensemble des utilisateurs u.

I ensemble des items i.

I_u ensemble des items notés par l'utilisateur u.

$\mathrm{I}_{u,v}$ ensemble des items co-notés par les utilisateurs u et v.

R matrice des *ratings*.

R′ matrice des *ratings* implicites.

r *rating* – $r_{u,i}$ représente la note que l'utilisateur u a assigné explicitement à l'item i.

$\hat{r}$ prédiction d'un *rating* – $\hat{r}_{u,i}$ représente la prédiction de la note que l'utilisateur u assignerait à l'item i.

Chapitre 1

Introduction

Sommaire

1.1 Contexte général 1

1.2 Challenges de recherche et contributions 2

1.3 Structure du manuscrit 6

1.1 Contexte général

Nous vivons aujourd'hui dans un monde résolument basé sur le digital et l'information. L'intelligence artificielle (notamment via l'apprentissage automatique et l'apprentissage profond) a connu une forte croissance ces dernières décennies, notamment grâce à l'augmentation exponentielle du volume des données publiées chaque jour sur Internet. Désormais, notre vie quotidienne est significativement impactée par l'utilisation de ces technologies, mises en œuvre dans des applications réelles dont les systèmes de recommandation (SdR) font partie. A cause du flot d'informations qui nous submerge, il devient très difficile pour les utilisateurs de trouver par eux-mêmes les *items* pertinents au sein de catalogues. Ces items peuvent être des articles à acheter, des films ou des disques intéressants ou encore des voyages, pour ne citer que quelques domaines de recommandation. Parmi un large catalogue d'*items*, un SdR possède la capacité d'extraire une courte liste composée des items qui correspondent au mieux aux *goûts* de chaque utilisateur. Ce processus de recommandation est généralement réalisé en deux phases consécutives : la *prédiction* et la *recommandation* à proprement parler. La *prédiction* consiste a prédire l'intérêt de l'utilisateur pour chaque item, afin de ne filtrer et ne garder qu'une liste des items les plus pertinents pour lui. La deuxième phase consiste à affiner et ordonner ces items filtrés, en considérant éventuellement de nouveaux aspects, afin de restreindre encore cette liste qui sera finalement proposée à l'utilisateur.

Les systèmes de recommandation sont devenus essentiels dans de nombreux *domaines* liés à notre vie quotidienne. Par exemple, en tant qu'internaute ils nous conseillent des

sites web à visiter. En tant que consommateur, ils nous aident à décider quel produit acheter, quel film regarder, quelle musique écouter, quel lieu touristique visiter ou quel restaurant choisir. Mais ils peuvent aussi avoir une influence sur notre vie sociale en suggérant de nouvelles connexions dans nos réseaux sociaux, ou en identifiant des profils à recruter. Chaque domaine spécifique est associé à des *connaissances* propres qui peuvent constituer un réel atout pour un système de recommandation dudit domaine. Considérons par exemple le domaine cinématographique, les connaissances associées à un film incluent le genre du film, ses acteurs, son réalisateur, l'année et le lieu de tournage, le sujet traité, etc.

L'*ingénierie des connaissances* est une branche de l'intelligence artificielle qui s'intéresse à la modélisation, la représentation, la manipulation et l'extraction formelles de connaissances dans différents *domaines* applicatifs. Elle fournit des protocoles standardisés permettant aux machines de gérer efficacement les connaissances. Par exemple, une ontologie peut être définie avec le langage OWL[1] (*Ontology Web Language*) pour représenter les concepts majeurs d'un domaine donné ainsi que les relations sémantiques et hiérarchiques entre ces concepts. De plus, à l'aide du Web sémantique et des données liées (*Linked Data* en anglais), des graphes de connaissances peuvent être construits afin de modéliser les faits sous forme de triplets RDF[2] (*Resource Description Framework*). Un tel graphe permet de modéliser des faits relatifs à un domaine ou aux multi-domaines, en reliant les entités (nœuds) par les relations sémantiques (arêtes).

La question suivante se pose : *Quand et comment l'ingénierie des connaissances peut-t-elle enrichir les systèmes de recommandation ?*

1.2 Challenges de recherche et contributions

Les SdR sont généralement catégorisés en trois familles, définies par le type de filtrage utilisé : le filtrage collaboratif (CF), le filtrage basé sur le contenu (CBF) et le filtrage hybride qui combine CF et CBF (voir chapitre 2). Le CF ignore complètement les connaissances liées aux items (*e.g.*, le genre des films) et se focalise uniquement sur les retours (*feedbacks* en anglais) des utilisateurs (la plupart du temps fournis sous la forme de notes attribuées aux items). A partir des *feedbacks* fournis par les utilisateurs et en se basant sur des modélisations mathématiques, le CF a pour objectif d'améliorer l'*accuracy*[3] des recommandations pour un utilisateur donné. Quant aux recommandations basées sur le contenu (CBF), il s'agit d'identifier des items qui sont similaires à ceux précédemment aimés par l'utilisateur. Des représentations des caractéristiques des items sont souvent

[1] https://www.w3.org/OWL/

[2] https://www.w3.org/RDF/

[3] Dans la communauté scientifique des SdR, le terme *accuracy* possède en général un sens large qui peut désigner une meilleure *prédiction* des notes (*justesse, exactitude*) ou une liste de recommandations ayant des valeurs de *précision, rappel, ou F1-mesure* élevées. Pour éviter toute ambiguïté liée à une mauvaise traduction de l'*accuracy* par ces métriques (termes) variées, nous emploierons le terme *accuracy*.

adoptées par les approches CBF pour modéliser les items et les utilisateurs. L'inconvénient majeur de ces approches est la *sur-spécialisation*, qui engendre souvent une liste trop peu diversifiée composée d'items similaires. Le filtrage collaboratif, quant à lui, ne souffre généralement pas de cette limitation, car il prend en compte également les goûts des autres utilisateurs. De plus, un autre avantage du CF par rapport au CBF est qu'il n'a pas besoin de construire un modèle pour caractériser et représenter les items. De ce fait, le CF s'est imposé un temps comme l'approche la plus populaire des SdR. Cependant, il souffre d'autres limitations, notamment liées à la rareté des données (*sparsity* en anglais) et au démarrage à froid : il échoue à proposer des items pertinents lorsqu'il n'y a que peu de retours (*feedbacks*) fournis par les utilisateurs. Pour pallier ces inconvénients et tirer parti des avantages de chaque approche (CBF et CF), les approches hybrides peuvent être envisagées. Toutefois, la façon de réaliser l'hybridation est très variée et peut dépendre des domaines de recommandation eux-mêmes.

La méthode du filtrage collaboratif s'est montrée pertinente concernant la prédiction des préférences des utilisateurs (*accuracy*). Les approches de CF classiques reposent principalement sur la similarité entre utilisateurs ou entre items ; similarité généralement estimée en se basant sur les items co-notés. Or, le fait que les utilisateurs ne notent, d'ordinaire, que très peu d'items du catalogue peut rendre l'estimation de la similarité peu fiable. Outre cela, il faut également prendre en compte la disparité dans les comportements de notations des utilisateurs. Dans le but de remédier à ces inconvénients et d'améliorer l'*accuracy* des recommandations de CF classiques, nous posons le premier challenge de recherche de cette thèse :

Challenge 1. *Comment rendre la mesure de similarité entre utilisateur plus fiable, et ainsi améliorer l'accuracy des recommandations du filtrage collaboratif ?*

Contribution 1. Pour répondre à cette question, nous proposons une méthode qui permet d'ajuster (de corriger) les valeurs de similarité et ainsi d'améliorer l'*accuracy* de la prédiction (cf. chapitre 3). Plus précisément, nous proposons tout d'abord un ratio de concordance (CR) qui permet de mesurer la cohérence des notations de deux utilisateurs en prenant en compte d'éventuelles disparités dans leur manière de noter. Ensuite, nous proposons une approche pour ajuster ces ratios de concordance en fonction du nombre d'items co-notés. Basée sur l'inférence bayésienne *empirique* (EB) [Brown, 2008], l'approche proposée (notée EBCR) permet de tenir compte du nombre d'items co-notés par les utilisateurs et d'ajuster les ratios mesurés entre deux utilisateurs n'ayant noté que très peu d'items en commun.

Alors même que les approches de CF ignorent en général les connaissances du domaine (*e.g.* les caractéristiques des items), il semble plus naturel pour les approches de CBF de tirer parti de ces connaissances et d'exploiter la sémantique des items à travers de l'ingénierie des connaissances. La figure 1.1 illustre les différentes phases du processus

de recommandation et celles sur lesquelles l'ingénierie des connaissances peut avoir une valeur ajoutée [Du et al., 2019b].

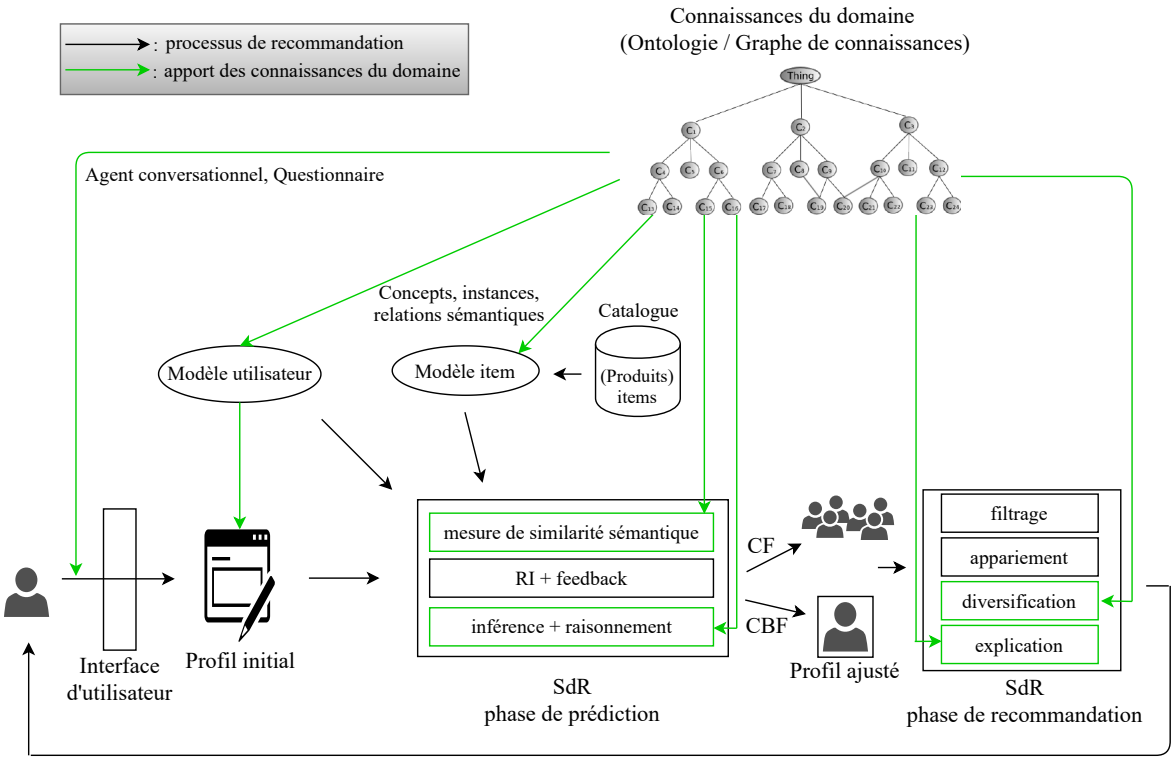


FIGURE 1.1 – Apports des connaissances du domaine aux différentes phases de recommandation

Nous constatons sur cette figure 1.1 que les connaissances du domaine et la sémantique, quelles soient modélisées par des ontologies ou par des graphes de connaissances, peuvent intervenir dans différentes étapes du processus de recommandation pour : mieux construire le modèle des utilisateurs et/ou le modèle des items, avoir une meilleure estimation de la similarité sémantique, fournir des capacités de raisonnement. Ces connaissances riches du domaine peuvent également apporter une plus-value en ce qui concerne les critères de *diversité* des recommandations faites aux utilisateurs et sur leur *explicabilité*. Ce dernier point est capital car ce n'est que s'il comprend les recommandations qui lui sont faites qu'un utilisateur accordera sa confiance au système et pourra éventuellement consommer/acheter l'item recommandé. Contrairement à l'*accuracy* des recommandations dont l'évaluation se focalise généralement sur la performance des modèles prédictifs purement orientés « *données* » (le CF par exemple), les critères de diversité et d'explicabilité peuvent avoir une forte synergie avec les « *connaissances* » du domaine.

La performance des systèmes de recommandation est depuis longtemps évaluée en fonction de leur *accuracy*. Dès lors qu'un nouveau modèle est proposé, ses auteurs le comparent systématiquement avec les modèles de référence, en termes d'*accuracy*. Avec le développement rapide de l'apprentissage automatique et de l'apprentissage profond, les modèles de recommandation récents sont de plus en plus performants en termes

d'*accuracy*, au détriment de la diversification qui constitue, pourtant, un facteur majeur de la satisfaction des utilisateurs. Ainsi, une analyse de la performance des différentes familles de recommandation en termes de *diversité* est nécessaire. Dans ce contexte, nous formulons le deuxième challenge de recherche de cette thèse :

Challenge 2. *Comment les principales familles de systèmes de recommandation se comportent-elles en termes de diversification ? Comment l'ingénierie des connaissances peut-elle contribuer à mieux comprendre la diversité des recommandations ?*

Contribution 2. Pour répondre à ces questions, nous avons conduit une étude approfondie (cf. chapitre 4) visant à comparer et analyser la performance de différentes familles de systèmes de recommandation, incluant les approches classiques (*e.g.*, CF et CBF) ainsi que celles plus récentes basées sur les réseaux de neurones [Ferrari Dacrema et al., 2019] et sur les *embeddings* de graphes [Palumbo et al., 2018a]. Notre étude est basée sur trois jeux de données de différents domaines de recommandation (films, mangas et livres). Pour mener à bien notre étude, nous avons tiré parti de l'ingénierie des connaissances, plus précisément le graphe de connaissances DBpedia, afin d'estimer la diversité des recommandations (basée sur la mesure de distance sémantique entre les items). Nous étudions et montrons aussi l'intérêt d'un *degré de diversité personnalisé* en fonction des profils utilisateurs.

Outre la *diversité* des items proposés, l'*explication* des recommandations constitue, elle aussi, une fonctionnalité importante des SdR. Alors même que de plus en plus d'aspects de notre vie quotidienne sont influencés par des décisions automatisées prises par des systèmes de recommandation, il devient naturel de se demander si ces systèmes sont dignes de confiance, compte tenu notamment de l'opacité et de la complexité de leur fonctionnement interne. Proposer des recommandations pertinentes est une bonne chose, mais fournir en plus les raisons qui expliquent ou justifient ces recommandations est encore mieux. Pour cela, l'ingénierie des connaissances peut également apporter une réelle plus-value, et favoriser ainsi l'explicabilité des recommandations. Dans ce contexte, nous formulons notre troisième challenge de recherche :

Challenge 3. *Comment les connaissances du domaine peuvent-elles intervenir pour favoriser l'explicabilité des systèmes de recommandation ?*

Contribution 3. En réponse à cette question, nous proposons une approche basée sur le graphe de connaissances DBpedia (cf. chapitre 5), qui élabore des explications concernant des recommandations faites à l'utilisateur, indépendamment du SdR utilisé (*i.e.* l'approche dite *post-hoc*). Plus précisément, notre approche est inspirée de travaux récents [Musto et al., 2016b, 2019] dans lesquels les explications des items recommandés sont basées sur les propriétés communes entre les items recommandés et ceux précédemment aimés par l'utilisateur. Nous avons amélioré la qualité des explications avec une meilleure exploration du graphe de connaissances et en considérant les relations hiérarchiques entre ses entités.

1.3 Structure du manuscrit

Partant de l'amélioration de l'*accuracy* des recommandations, en passant par la contribution à une plus grande *diversité* des recommandations pour enfin parvenir à une meilleure *explicabilité*, cette thèse explore les apports de l'IA symbolique et plus précisément des modèles de connaissance aux systèmes de recommandation. Elle propose une vision, *des « données » aux « connaissances »* de l'amélioration de leurs performances.

Le manuscrit est divisé en 6 chapitres, répondant à trois challenges de recherche que nous avons identifiés concernant les liens possibles entre l'ingénierie des connaissances et les systèmes de recommandation. La structure du manuscrit est décrite ci-dessous et schématisée dans la figure 1.2.

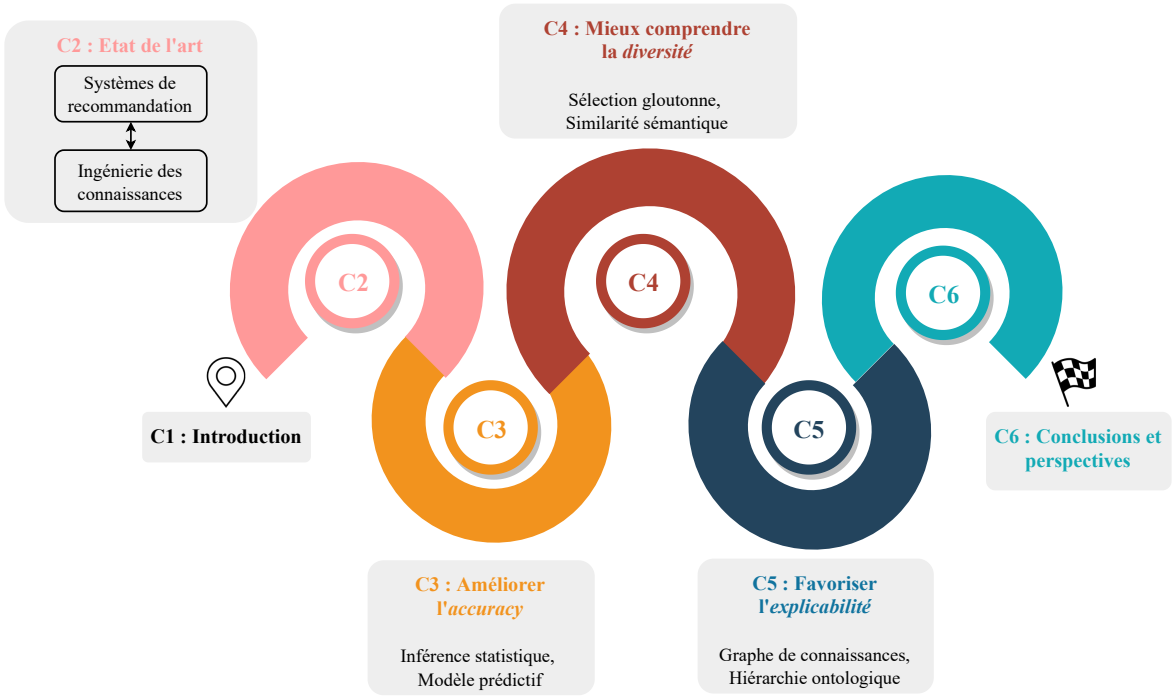


FIGURE 1.2 – Plan du manuscrit et mots-clés des chapitres

Le chapitre 2 détaille les concepts de base des systèmes de recommandation. Les principales familles de SdR sont présentées ainsi que les protocoles d'évaluation associés. Nous synthétisons également les notions de base de l'ingénierie des connaissances ainsi que ses apports aux systèmes de recommandation, à travers les approches de la littérature.

Le chapitre 3 détaille l'approche proposée (*i.e.* EBCR) pour l'amélioration de l'*accuracy* des recommandations dans le contexte du filtrage collaboratif. Nous formulerons la problématique, décrirons la méthode proposée ainsi que l'évaluation expérimentale de cette méthode.

Le chapitre 4 présente en détail l'étude conduite pour l'analyse et la comparaison de la performance des SdR en termes de *diversification*. Nous présenterons les principales

stratégies de diversification de l'état de l'art, l'approche nouvelle que nous proposons, ainsi que le protocole et les résultats de l'étude conduite.

Le chapitre 5 décrit l'approche d'*explicabilité* proposée dans cette thèse. Nous discutons d'abord des notions de base des explications de recommandation. Nous présenterons la motivation et les apports de notre contribution. Les résultats de l'évaluation de notre approche basée sur un test mené à grande échelle (155 participants) seront ensuite discutés.

Enfin, dans le chapitre 6, nous tirons les principales conclusions de la thèse et ouvrons de nouvelles perspectives de recherche dans les domaines concernés.

Chapitre 2

État de l'art

Sommaire

2.1 Systèmes de recommandation (SdR) 9

2.1.1 Contexte et problématique 9

2.1.2 Définition et catégorisation des systèmes de recommandation . . . 10

2.1.2.1 Matrice des *ratings* 11

2.1.2.2 Contenu des items 12

2.1.2.3 Catégorisation des SdR 12

2.1.3 Approches basées sur le filtrage collaboratif 13

2.1.3.1 Approches basées sur la mémoire / le voisinage 14

2.1.3.2 Approches basées sur les modèles 19

2.1.4 Approches basées sur le contenu 29

2.1.4.1 Pré-traitement et analyse 30

2.1.4.2 Construction des profils des utilisateurs 32

2.1.4.3 Filtrage, appariement et recommandation 33

2.1.4.4 Comparaison avec les approches de filtrage collaboratif . . 35

2.1.5 Approches hybrides 37

2.1.6 Évaluation des systèmes de recommandation 39

2.1.6.1 Évaluation basée sur la prédiction des *ratings* 40

2.1.6.2 Évaluation basée sur la liste de top-N recommandations . 41

2.1.6.3 Comparaison des deux types d'évaluation 43

2.1.6.4 Protocoles d'évaluation 44

2.1.6.5 Autres objectifs d'une liste de recommandations et évaluations associées 46

2.2 Apports de l'ingénierie des connaissances aux SdR 47

2.2.1 Apports des ontologies 48
2.2.1.1 Ontologie 48
2.2.1.2 Annotation conceptuelle pour représenter les utilisateurs ou les items 49
2.2.1.3 Mesures de similarité sémantique entre items et/ou entre utilisateurs 53
2.2.2 Apports des graphes de connaissances 56
2.2.2.1 Modèles de recommandation basés sur les *Linked Data* . . 56
2.2.2.2 Modèles de recommandation basés sur les plongements de graphes, *Knowledge Graph Embedding* (KGE) 60
2.3 Conclusion de chapitre 64

2.1 Systèmes de recommandation (SdR)

2.1.1 Contexte et problématique

Un système de recommandation (SdR) est un environnement logiciel qui permet de fournir à chaque utilisateur les éléments (items) les plus susceptibles de l'intéresser [Ricci et al., 2015]. Pour cela, classiquement, deux phases consécutives sont nécessaires : la phase dite de « *prédiction* » et la phase de « *recommandation* » à proprement parler. Dans la première phase, le système est doté de la capacité de découvrir les intérêts des utilisateurs, c'est-à-dire de prédire leurs possibles *goûts* concernant les items que ces utilisateurs n'ont pas encore évalués. La plupart du temps, la prédiction du goût d'un utilisateur pour un item est représentée par une note qui est l'estimation faite par le système de recommandation de l'évaluation que ferait l'utilisateur de cet item-là. Au cours de la deuxième phase, le SdR ordonne d'abord les items selon les notes prédites pour recommander par la suite une liste des n items ayant les prédictions les plus élevées (top-n).

Néanmoins, il existe également des systèmes de recommandation atypiques :

- Pour certains, il n'y a pas vraiment de distinction claire entre la phase de *prédiction* et celle de *recommandation*. Plus précisément, ce genre de SdR génère une liste de recommandations sans avoir besoin, au préalable, de prédire des notes. Un exemple représentatif est l'approche d'apprentissage par classement (*learning to rank* en anglais) [Liu, 2009; Karatzoglou et al., 2013].

- Contrairement aux cas classiques, où seuls les items non notés sont légitimes pour la recommandation, d'autres SdR considèrent que même les items pour lesquels les goûts des utilisateurs sont connus peuvent être recommandés à nouveau, dans des contextes différents. Prenant en compte ces contextes (*e.g.*, le temps), ces SdR visent

à augmenter au maximum la personnalisation pour l'utilisateur cible. Par exemple, le système de recommandation de Youtube [Covington et al., 2016] tient compte de l'aspect temporel, permettant de recommander à nouveau des vidéos déjà regardées par l'utilisateur (*e.g.*, une vidéo que l'utilisateur regarde de façon récurrente).

Considérant ces diverses approches de SdR et afin d'éviter toute ambiguïté, dans la suite de ce manuscrit on notera SdR uniquement des systèmes de recommandation classiques[1] qui :

- disposent de deux phases consécutives, *i.e.* les phases de « prédiction » et de « recommandation » ;
- ne prédisent les goûts des utilisateurs que pour les items non notés ;
- ne prennent pas en compte le contexte, ni le temps.

2.1.2 Définition et catégorisation des systèmes de recommandation

Un système de recommandation repose sur trois éléments de base : un ensemble d'utilisateurs $u \in U$, un catalogue d'items $i \in I$ et une collection de retours d'utilisateurs (en anglais *feedbacks*), qui représentent leurs goûts concernant certains items. On distingue deux types de *feedback*, détaillés ci-dessous.

- ***Feedbacks* explicites** : il s'agit de retours explicites sous la forme de notes (en anglais *ratings*[2]) que les utilisateurs attribuent aux items. Les *ratings* sont souvent saisis sur une échelle graduée qui indique le niveau d'appréciation de l'utilisateur par rapport aux items concernés [Aggarwal, 2016e]. Il est possible que les valeurs de *ratings* soient continues, comme c'est le cas dans le SdR de blagues (Jester[3]), dans lequel les *ratings* peuvent prendre toute valeur comprise entre -10 et 10. Cela est toutefois relativement rare. Généralement, les *ratings* sont basés sur un intervalle, où un ensemble discret de nombres ordonnés est utilisé pour quantifier ce que l'on aime ou n'aime pas. Un exemple typique, largement utilisé dans de nombreux SdR populaires (*e.g.* Amazon, Netflix), consiste à attribuer les *ratings* sur l'ensemble {1, 2, 3, 4, 5} (voir figure 2.1). Dans ce contexte, un *rating* de 1 indique une dégoût extrême alors que 5 indique l'appréciation maximale. Formellement, on emploie le terme $r_{u,i}$ pour représenter le *rating* que l'utilisateur u assigne à l'item i. Les *ratings* ont également été largement adoptés par la communauté scientifique depuis plusieurs années. Un exemple notable est celui du prix Netflix [Bennett et al., 2007].

[1] Le lecteur intéressé par les systèmes de recommandation sensibles au contexte/temps peut se référer à [Aggarwal, 2016b,g]

[2] Notons que, pour éviter tout ambiguïté liée au mot « note », le terme « *rating* » est employé tout au long du manuscrit pour représenter les retours explicites

[3] https://grouplens.org/datasets/jester/

Il s'agit d'une compétition où les algorithmes doivent prédire les *ratings*, *i.e.* les retours explicites des utilisateurs, pour les films qu'ils ont regardés. L'objectif consiste à proposer des algorithmes d'apprentissage automatique qui permettent de prédire les valeurs des *ratings* ($r_{u,i}$) (sur un jeu de test) à partir d'un jeu d'entraînement dans lequel les *ratings* sont connus. La prédiction du *rating* que l'utilisateur u assignerait à l'item i est notée $\hat{r}_{u,i}$.



FIGURE 2.1 – Un exemple du *feedback* explicite sous l'échelle graduée de 1 à 5. (Source Amazon)

- ***Feedbacks* implicites :** à la place, ou en complément, des retours explicites représentées par les *ratings*, un SdR peut aussi disposer de retours implicites. Il s'agit souvent de traces basées sur les comportements des utilisateurs ou les interactions « utilisateur-item », qui permettent de montrer implicitement leur intérêt par rapport aux items avec lesquels ils ont interagit. Par exemple, si un utilisateur écoute fréquemment la musique d'un artiste, il est raisonnable de supposer qu'il aime cet artiste. De même, si un utilisateur a acheté un vêtement et ne le retourne pas, on peut supposer que c'est parce qu'il l'apprécie. Bien qu'on ne dispose pas de retour explicite de l'utilisateur pour l'artiste ou le vêtement en question, des hypothèses peuvent être posées. Contrairement aux *ratings* explicites qui utilisent des nombres ordinaux pour exprimer les goûts des utilisateurs, les goûts implicites sont souvent représentés par des *ratings* unaires. Par exemple, le fait d'acheter un produit implique que l'utilisateur l'aime, par contre, le fait de ne pas acheter un produit n'implique pas forcément que l'utilisateur ne l'aime pas.

2.1.2.1 Matrice des *ratings*

Dans un système de recommandation, les retours des utilisateurs, qu'ils soient explicites ou implicites, peuvent être représentés par une matrice dont les lignes représentent les utilisateurs et les colonnes représentent les items. Formellement, pour un SdR avec |U|

utilisateurs et |I| items, la matrice des *ratings*, $R_{|U|\times|I|}$ est systématiquement construite. $r_{u,i}$, *i.e.* la valeur de *rating* assignée par l'utilisateur u pour l'item i, est stockée dans la case située à la u-ième ligne et à la i-ième colonne de la matrice R. La figure 2.2 illustre un exemple de deux matrices de *ratings* qui représentent les retours de 4 utilisateurs sur 4 items.

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

(a) *ratings* explicites sur une échelle de 1 à 5

	i_1	i_2	i_3	i_4
u_1	1			1
u_2		1		
u_3	1		1	1
u_4			1	

(b) *ratings* implicites

FIGURE 2.2 – Exemples de matrices de *ratings* pour des *feedbacks* explicites (a) et implicites (b)

2.1.2.2 Contenu des items

Mis à part la matrice des *ratings* qui joue un rôle essentiel, un système de recommandation peut disposer d'autres types d'informations, tout aussi importantes, notamment des données permettant de décrire les items du catalogue [Lops et al., 2011]. Ces descriptions, dites *de contenu*, peuvent s'apparenter à du *catalogage*, qui encode les informations relatives au *contenant* (édition, auteur, date de création,...), ou à de l'*indexation*, qui s'attache à décrire le contenu lui-même (propriétés propres à l'item comme sa taille, son poids, ...). Nous ne ferons pas la distinction ici et assimilerons parfois les items à la description de leur contenu. Les contenus des items peuvent être décrits par de nombreux types de données. Ce peut être des données non-structurées (*e.g.*, description sous forme de texte, image, etc.) aussi bien que des données semi-structurées (*e.g.*, tableau de caractéristiques) ou encore un mélange des deux. Par exemple, une vidéo de Youtube peut être décrite par son titre et sa durée alors qu'un produit vendu sur le site d'Amazon peut être décrit par ses caractéristiques principales (cf. figure 2.3).

2.1.2.3 Catégorisation des SdR

Les systèmes de recommandation peuvent être catégorisés en fonction du type de données utilisées durant la phase de prédiction : interactions utilisateur-item (matrice des *ratings*) et/ou contenus des items. Les SdR utilisant le premier type de données sont catégorisés comme des approches de filtrage collaboratif et ceux qui sont basés sur le deuxième type de données sont catégorisés comme des approches basées sur les contenus. Il existe également une troisième catégorie d'approche, dite hybride, qui combine les deux types





	iPhone 12	iPhone 12 Pro
Prix	Dès: 959,00 €	Dès: 1 509,00 €
Notes	★★★★½ (95)	★★★★½ (47)
Écran	Écran Super Retina XDR 6,1 pouces	Écran Super Retina XDR 6,1 pouces
Caméra	Double appareil photo ultra grand-angle et grand-angle 12 Mpx	Triple appareil photo ultra grand-angle, grand-angle et téléobjectif 12 Mpx
Caméra avant	Caméra TrueDepth 12 Mpx	Caméra TrueDepth 12 Mpx
Face ID or Touch ID	Face ID	Face ID
Puce	Puce A14 Bionic avec Neural Engine nouvelle génération	Puce A14 Bionic avec Neural Engine nouvelle génération
Résistance à l'eau	Indice de protection IP68 (jusqu'à 6 mètres de profondeur pendant 30 minutes maximum)	Indice de protection IP68 (jusqu'à 6 mètres de profondeur pendant 30 minutes maximum)
Alimentation et batterie	✓ (avec chargeur MagSafe jusqu'à 15 W ou recharge sans fil Qi jusqu'à 7,5 W)	✓ (avec chargeur MagSafe jusqu'à 15 W ou recharge sans fil Qi jusqu'à 7,5 W)

FIGURE 2.3 – Exemple de description de contenu des iPhone sur Amazon

de données pour prédire les goûts des utilisateurs. Notons que les SdR basés sur les contenus utilisent également les *ratings* dans la plupart des cas, mais en ne se focalisant que sur les *ratings* d'un seul utilisateur (celui pour qui on fait des recommandations) plutôt que sur ceux de tous les utilisateurs. Dans ce qui suit, on rappelle les principes de chaque catégorie de systèmes de recommandation.

2.1.3 Approches basées sur le filtrage collaboratif

Le filtrage collaboratif – ou *Collaborative Filtering* (CF), est une technique largement employée dans le domaine (à la fois scientifique et industriel) des systèmes de recommandation. Pour recommander des items à l'utilisateur u, le CF s'appuie sur les opinions des autres utilisateurs, d'où le terme « collaboratif ». GroupLens [Resnick et al., 1994; Konstan et al., 1997] est l'un des premiers systèmes de recommandation s'appuyant sur le filtrage collaboratif. Dans le domaine du commerce électronique, le SdR d'Amazon est également représentatif de cette technique : « Les clients ayant acheté ce produit ont également acheté... ». Notons que dans un SdR, il est très rare que les utilisateurs donnent un avis (*rating*) pour chaque item du catalogue, autrement dit, la matrice des *ratings* (R) est incomplète. De ce fait, l'objectif du CF est de prédire les valeurs des *ratings* manquants, en prenant appui sur les *ratings* disponibles, *i.e.* R initial. L'approche du filtrage collaboratif

est divisée en deux sous-catégories qui dépendent du procédé de prédiction : l'approche CF basée sur la mémoire et l'approche CF basée sur les modèles.

2.1.3.1 Approches basées sur la mémoire / le voisinage

L'approche de filtrage collaboratif basée sur la mémoire, également désignée par filtrage collaboratif basé sur le voisinage – ou *Neighborhood-Based Collaborative Filtering* (NBCF), fait partie des premiers algorithmes de filtrage collaboratif [Resnick et al., 1994; Sarwar et al., 2001; Herlocker et al., 2002]. La stratégie principale de l'approche repose sur les hypothèses suivantes : i) les utilisateurs similaires notent les items de façon analogue, autrement dit, leurs systèmes de notation sont similaires et ii) les items similaires reçoivent des *ratings* similaires. On peut ainsi distinguer deux types d'approches :

- NBCF basé sur les *utilisateurs*. Dans ce cas, les *ratings* fournis par les utilisateurs similaires à l'utilisateur u sont utilisés pour faire des recommandations pour u. La prédiction de $\hat{r}_{u,i}$ est calculée comme la moyenne pondérée des *ratings* pour l'item i fournis par le voisinage de l'utilisateur u.

- NBCF basé sur les *items*. Dans ce cas, on identifie d'abord un ensemble d'items similaires à l'item i. Puis on prédit $\hat{r}_{u,i}$ en fonction des *ratings* spécifiques fournis par u pour ces items.

Qu'elles reposent sur les utilisateurs ou les items, les approches basées sur le voisinage (NBCF) sont constituées de deux étapes clés consécutives : i) *l'identification du voisinage* et ii) *la prédiction des ratings* à partir du voisinage déterminé dans la première étape. Dans les sections qui suivent, nous discutons chacune de ces étapes.

2.1.3.1.1 Identification du voisinage A partir de la matrice des *ratings* R, l'idée de base des approches de NBCF est de faire la recommandation en fonction de la similarité entre utilisateurs ou entre items. Cela implique que l'on doive déterminer soit des utilisateurs similaires, soit des items similaires, *i.e.* identifier le voisinage. Pour ce faire, l'approche des k plus proches voisins (KNN pour k *nearest neighbors*) est utilisée. Généralement, comme les utilisateurs et les items sont représentés par des vecteurs de *ratings* dans la matrice R, les métriques standardisées permettant de calculer la proximité entre des vecteurs peuvent être utilisées. Les représentations vectorielles des utilisateurs $u \in$ U et des items $i \in$ I sont obtenues à partir de la matrice R. Par exemple, le vecteur pour l'utilisateur u, est défini comme $\overrightarrow{u} = (r_{u,i}, i = 1,2,...,|\text{I}|)$. De manière similaire, le vecteur d'un item i est défini comme $\overrightarrow{i} = (r_{u,i}, u = 1,2,...,|\text{U}|)$. Nous introduisons par la suite quelques métriques courantes pour le calcul de similarité entre vecteurs dans le contexte du NBCF.

La similarité cosinus (COS) est une métrique largement utilisée dans le domaine de la recherche d'information et de la fouille de textes pour calculer la similarité entre deux documents représentés par la fréquence des mots qu'ils contiennent [Crnic, 2011]. Elle

représente le cosinus de l'angle entre les deux vecteurs représentant ces documents dans l'espace à n dimensions, avec n étant la longueur des vecteurs. Dans le contexte du NBCF, elle peut être employée pour mesurer la similarité entre deux utilisateurs ou entre deux items.

Formellement, la similarité cosinus entre deux utilisateurs est définie par l'équation (2.1) :

$$sim_{\text{COS}}(u,v) = \frac{\sum\limits_{i \in \text{I}_{u,v}} r_{u,i} \times r_{v,i}}{\sqrt{\sum\limits_{i \in \text{I}_{u,v}} r_{u,i}^2} \sqrt{\sum\limits_{i \in \text{I}_{u,v}} r_{v,i}^2}} \tag{2.1}$$

où $\text{I}_{u,v}$ représente l'ensemble des items co-notés par l'utilisateur u et l'utilisateur v. De façon similaire, la similarité entre deux items est définie par l'équation (2.2), où $\text{U}_{i,j}$ représente l'ensemble des utilisateurs qui ont noté à la fois l'item i et l'item j.

$$sim_{\text{COS}}(i,j) = \frac{\sum\limits_{u \in \text{U}_{i,j}} r_{u,i} \times r_{u,j}}{\sqrt{\sum\limits_{u \in \text{U}_{i,j}} r_{u,i}^2} \sqrt{\sum\limits_{u \in \text{U}_{i,j}} r_{u,j}^2}} \tag{2.2}$$

Une autre métrique de similarité largement utilisée dans ce contexte est le coefficient de corrélation de Pearson – ou *Pearson Correlation Coefficient* (PCC) [Resnick et al., 1994]. PCC est généralement employé en statistique pour mesurer une corrélation entre deux variables, ses valeurs varient entre -1 et 1. La valeur de 1 traduit une corrélation parfaite et positive alors que la valeur de -1 signifie que les deux variables sont totalement et négativement corrélées (anti-corrélées). La valeur de 0 signifie qu'il n'y a pas de corrélation linéaire entre les deux variables. Formellement, pour mesurer la similarité entre deux utilisateurs u et v, PCC est défini par l'équation (2.3) :

$$sim_{\text{PCC}}(u,v) = \frac{\sum\limits_{i \in \text{I}_{u,v}} [(r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)]}{\sqrt{\sum\limits_{i \in \text{I}_{u,v}} (r_{u,i} - \overline{r}_u)^2 \sum\limits_{i \in \text{I}_{u,v}} (r_{v,i} - \overline{r}_v)^2}} \tag{2.3}$$

où $\overline{r}_u$ et $\overline{r}_v$ représentent respectivement la valeur moyenne des *ratings* de l'utilisateur u et de l'utilisateur v. La similarité entre deux items est, quant à elle, définie par l'équation (2.4), avec $\overline{r}_i$ et $\overline{r}_j$ représentant respectivement la valeur moyenne des *ratings* fournis par tous les utilisateurs pour l'item i et l'item j.

$$sim_{\text{PCC}}(i,j) = \frac{\sum\limits_{u \in \text{U}_{i,j}} [(r_{u,i} - \overline{r}_i)(r_{u,j} - \overline{r}_j)]}{\sqrt{\sum\limits_{u \in \text{U}_{i,j}} (r_{u,i} - \overline{r}_i)^2 \sum\limits_{u \in \text{U}_{i,j}} (r_{u,j} - \overline{r}_j)^2}} \tag{2.4}$$

Au lieu de mesurer directement la *similarité* entre des vecteurs, comme le font les métriques COS et PCC, d'autres approches calculent d'abord une *distance* entre les vecteurs et la convertissent ensuite en *similarité*. Pour cela, la distance dans l'espace Euclidien est

généralement adoptée. Formellement, étant donné deux vecteurs $\vec{a}$ et $\vec{b}$ de dimension n, la distance euclidienne entre $\vec{a}$ et $\vec{b}$ est définie par l'équation (2.5) :

$$d(\vec{a}, \vec{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2 + ... + (a_n - b_n)^2} \tag{2.5}$$

Dans de nombreuses applications, et en particulier lors de la comparaison des distances, il peut être plus pratique d'omettre la racine carrée dans le calcul des distances euclidiennes (voir équation (2.6)). La distance résultante est alors appelée distance euclidienne quadratique [Spencer, 2013]. Dans le contexte du NBCF, on peut considérer deux utilisateurs (resp. deux items) comme deux points positionnés dans l'espace Euclidien de dimension $|I_{u,v}|$ (resp. $|U_{i,j}|$). Ainsi, la distance peut être calculée par la distance euclidienne quadratique. Pourtant, la distance calculée de cette manière a tendance à être biaisée car les utilisateurs qui ont co-noté beaucoup d'items sembleront plus éloignés que ceux pour lesquels il n'y a que peu d'items notés en commun. De ce fait, il est préférable d'utiliser la distance quadratique moyenne – ou *Mean Squared Distance* (MSD). Formellement, pour calculer la distance entre deux utilisateurs u et v (resp. deux items i et j), MSD est définie par l'équation (2.7) (resp. l'équation (2.9)).

$$d(\vec{a}, \vec{b}) = (a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2 + ... + (a_n - b_n)^2 \tag{2.6}$$

$$\text{MSD}(u, v) = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - r_{v,i})^2}{|I_{u,v}|} \tag{2.7}$$

$$sim_{\text{MSD}}(u, v) = \frac{1}{\text{MSD}(u, v) + 1} \tag{2.8}$$

$$\text{MSD}(i, j) = \frac{\sum_{u \in U_{i,j}} (r_{u,i} - r_{u,j})^2}{|U_{i,j}|} \tag{2.9}$$

$$sim_{\text{MSD}}(i, j) = \frac{1}{\text{MSD}(i, j) + 1} \tag{2.10}$$

Afin de convertir une distance d en similarité sim, différentes stratégies peuvent être appliquées, comme la mesure de son inverse, *i.e.* d^{-1} ou la formule $1 - d$ lorsque la distance d est normalisée dans l'intervalle $[0, 1]$. L'équation (2.8) représente, par exemple, la similarité entre deux utilisateurs u et v convertie à partir de la distance de $\text{MSD}(u, v)$. Notons que dans ce cas, 1 est ajouté au dénominateur pour éviter une éventuelle division par 0.

2.1.3.1.2 Prédiction des *ratings* La deuxième étape de la procédure du NBCF est la prédiction des *ratings* à partir du voisinage (noté N pour *Neighborhood*) déterminé lors de la première étape. Concrètement, il s'agit de calculer soit la moyenne pondérée des *ratings* fournis par les individus du voisinage d'un utilisateur dans le cas du NBCF basé sur les utilisateurs, soit la moyenne pondérée des *ratings* associés aux éléments du voisinage d'un item donné dans le cas du NBCF basé sur les items. Considérons, par exemple, la prédiction de $\hat{r}_{u,i}$, dans le premier cas. $\hat{r}_{u,i}$ est alors obtenu en prenant la moyenne pon-

dérée des *ratings* de l'item i donnés par chacun des k plus proches voisins de l'utilisateur u ayant évalué i. La pondération des *ratings* est fonction de la similarité entre l'utilisateur u et ses voisins, autrement dit, plus un voisin est similaire à u, plus le poids associé au *rating* de ce voisin est grand. Formellement, la prédiction de $\hat{r}_{u,i}$ est obtenue par l'équation (2.11) pour l'approche de NBCF basée sur les utilisateurs, où le terme $N_u(i)$ représente les plus proches voisins de l'utilisateur u ayant noté l'item i. De façon similaire, dans le cas où la prédiction est basée sur les items, la valeur de $\hat{r}_{u,i}$ sera obtenue par l'équation (2.12) où le terme $N_i(u)$ représente l'ensemble des k plus proches voisins de l'item i ayant été notés par u.

$$\widehat{r}_{u,i} = \frac{\sum\limits_{v \in N_u(i)} r_{v,i} * sim(u,v)}{\sum\limits_{v \in N_u(i)} sim(u,v)} \quad (2.11) \qquad \widehat{r}_{u,i} = \frac{\sum\limits_{j \in N_i(u)} r_{u,j} * sim(i,j)}{\sum\limits_{j \in N_i(u)} sim(i,j)} \quad (2.12)$$

En général, la façon dont les utilisateurs évaluent les items est influencée par leur personnalité, leur humeur et aussi le contexte. Ainsi, sur une échelle d'évaluation de 1 à 5, un utilisateur optimiste (resp. pessimiste) donnera rarement un *rating* de 1 (resp. 5) à l'item même s'il ne l'aime pas (resp. s'il l'aime). Les distributions des *ratings* des utilisateurs peuvent donc être décalées ou comprimées les unes par rapport aux autres. Une distribution de *ratings* r_x est notamment caractérisée par la moyenne ($\overline{r}_x$) de ces *ratings* et leur variance (σ_x). Pour prendre en compte la disparité des systèmes de notation des utilisateurs, la normalisation des *ratings* est généralement employée. Les auteurs de [Herlocker et al., 2002] ont proposé de normaliser les *ratings* de chaque utilisateur. Deux normalisations sont envisagées : la normalisation par la *moyenne* (centré) et la normalisation par le *z-score* (centré et réduit).

Formellement, pour la première forme de normalisation, *i.e.* par la *moyenne*, les équations (2.11) et (2.12) sont remplacées par les équations (2.13) et (2.14).

$$\widehat{r}_{u,i} = \overline{r}_u + \frac{\sum\limits_{v \in N_u(i)} [(r_{v,i} - \overline{r}_v) * sim(u,v)]}{\sum\limits_{v \in N_u(i)} sim(u,v)} \quad (2.13)$$

$$\widehat{r}_{u,i} = \overline{r}_i + \frac{\sum\limits_{j \in N_i(u)} [(r_{u,j} - \overline{r}_j) * sim(i,j)]}{\sum\limits_{j \in N_i(u)} sim(i,j)} \quad (2.14)$$

Pour la deuxième forme de normalisation, *i.e.* par le *z-score*, les équations (2.11) et (2.12) sont remplacées par les équations (2.15) et (2.16).

$$\widehat{r}_{u,i} = \overline{r}_u + \sigma_u \frac{\sum\limits_{v \in N_u(i)} [(\frac{r_{v,i} - \overline{r}_v}{\sigma_v}) * sim(u,v)]}{\sum\limits_{v \in N_u(i)} sim(u,v)} \quad (2.15)$$

$$\widehat{r}_{u,i} = \overline{r}_i + \sigma_i \frac{\sum\limits_{j \in \mathrm{N}_i(u)} [(\frac{r_{u,j} - \overline{r}_j}{\sigma_j}) * sim(i,j)]}{\sum\limits_{j \in \mathrm{N}_i(u)} sim(i,j)} \tag{2.16}$$

2.1.3.1.3 Un exemple concret Dans cette section, nous illustrons la procédure des approches du NBCF par un exemple concret de prédiction des *ratings*.

Considérons la matrice des *ratings* représentée par la figure 2.4 dans laquelle on observe des *ratings* de 6 utilisateurs concernant 7 items. Notons que cette matrice est incomplète, étant donné qu'il y a des cases où les *ratings* sont inconnus. L'objectif ici est de prédire la valeur de *rating* de l'utilisateur u_1 pour l'item i_2, *i.e.* $\hat{r}_{u_1,i_2}$ (case verte), par l'approche de filtrage collaboratif basé sur le voisinage. Notons que différentes configurations de l'algorithme sont possibles selon par exemple le type de voisins choisi, leur nombre, la métrique de similarité, la méthode de normalisation, etc. Pour l'illustration, nous considérons : l'approche de NBCF basée sur les utilisateurs, la mesure PCC (cf. équation (2.3)) pour l'identification du voisinage et la normalisation par la *moyenne* (cf. équation (2.13)) pour la prédiction des *ratings*.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	1		3	1			2
u_2	2	1		5	5		
u_3		2	3				
u_4	1	2					
u_5	2	5	3	1			1
u_6	1		3	1		4	2

FIGURE 2.4 – Exemple d'une matrice des *ratings* pour illustrer l'approche NBCF. La case verte représente le *rating* dont on cherche à prédire la valeur.

Afin de prédire $\hat{r}_{u_1,i_2}$, la première étape consiste à identifier le voisinage de l'utilisateur u_1 au regard de l'item i_2, *i.e.* $\mathrm{N}_{u_1}(i_2)$. Dans ce cadre, u_6 n'est pas pertinent, puisqu'il n'a pas non plus noté l'item i_2. Ainsi, on a uniquement besoin de calculer la similarité entre l'utilisateur u_1 et les utilisateurs u_2, u_3, u_4 et u_5. En appliquant l'équation (2.3), on obtient :

$$
\begin{aligned}
sim_{\text{PCC}}(u_1, u_2) &= \frac{\sum_{i \in \text{I}_{u_1,u_2}} [(r_{u_1,i} - \overline{r}_{u_1})(r_{u_2,i} - \overline{r}_{u_2})]}{\sqrt{\sum_{i \in \text{I}_{u_1,u_2}} (r_{u_1,i} - \overline{r}_{u_1})^2 \sum_{i \in \text{I}_{u_1,u_2}} (r_{u_2,i} - \overline{r}_{u_2})^2}} \\
&= \frac{(1-1.75)(2-3.25) + (1-1.75)(5-3.25)}{\sqrt{[(1-1.75)^2 + (1-1.75)^2][(2-3.25)^2 + (5-3.25)^2]}} \\
&= -0.16
\end{aligned}
\tag{2.17}
$$

De la même manière, on obtient les autres valeurs de similarité : $sim_{\text{PCC}}(u_1, u_3) = 1.00$, $sim_{\text{PCC}}(u_1, u_4) = 1.00$, $sim_{\text{PCC}}(u_1, u_5) = 0.50$.

Une fois ces similarités calculées, la deuxième étape consiste à prédire la valeur de $\hat{r}_{u_1,i_2}$ en s'appuyant sur les *ratings* de l'item i_2 fournis par les utilisateurs appartenant au voisinage de l'utilisateur u_1. Notons que pour cet exemple, et étant donné le faible nombre d'utilisateurs, on considérera un voisinage de taille 3, mais dans la pratique, k est généralement plus grand. De ce fait, les 3 voisins de u_1 considérés pour obtenir $\hat{r}_{u_1,i_2}$ sont u_3, u_4, u_5, *i.e.* $\text{N}_{u_1}(i_2) = \{u_3, u_4, u_5\}$. En appliquant la formule de prédiction de l'équation (2.13), on obtient :

$$
\begin{aligned}
\widehat{r}_{u_1,i_2} &= \overline{r}_{u_1} + \frac{\sum_{v \in \text{N}_{u_1}(i_2)} [(r_{v,i} - \overline{r}_v) * sim(u_1, v)]}{\sum_{v \in \text{N}_{u_1}(i_2)} sim(u_1, v)} \\
&= 1.75 + \frac{(2-2.5) * 1.00 + (2-1.5) * 1.00 + (5-2.4) * 0.50}{1.00 + 1.00 + 0.50} \\
&= 2.27
\end{aligned}
\tag{2.18}
$$

Notons ici que le *rating* prédit n'est pas un entier comme les autres valeurs de la matrice (cf. figure 2.4), qui, elles, correspondent à des *ratings* réels et non prédits. Comme nous le présenterons dans la section 2.1.6, ces prédictions servent généralement à évaluer l'*accuracy* d'un SdR donné ou à ordonner les items candidats pour un utilisateur donné (*e.g.* ordonner les items i_2, i_5 et i_6 pour l'utilisateur u_1).

2.1.3.2 Approches basées sur les modèles

Les approches basées sur le voisinage, présentées dans la section précédente, ont été parmi les premières méthodes de filtrage collaboratif et ont également été parmi les plus populaires. Si elles gardent l'avantage de leur simplicité, elles ne sont pas nécessairement les méthodes les plus pertinentes pour la prédiction des *ratings* aujourd'hui.

Contrairement aux approches NBCF qui peuvent être considérées comme des méthodes spécifiques aux instances, une instance étant la valeur de $\hat{r}_{u,i}$ à prédire, il existe dans la littérature des approches de CF basées sur les modèles, dans lesquelles il y a

une claire distinction entre la phase d'entraînement et la phase de prédiction. Plus précisément, dans le cas des approches NBCF, un modèle n'est pas spécifiquement créé à l'avance pour la prédiction autre qu'une phase de pré-traitement (*e.g.* calculs des similarités, calculs des prédictions basées sur des équations pré-définies etc.). A l'inverse, pour les approches basées sur les modèles, un modèle d'apprentissage automatique (supervisé en général) est d'abord créé. Les paramètres de ce modèle sont ajustés sur la base des *ratings* disponibles dans le jeu de données, en optimisant une fonction objectif. On parle d'entraînement du modèle.

Les modèles classiques d'apprentissage automatique tels que les arbres de décision, les machines à vecteurs de support – ou *Support Vector Machine* (SVM), ou les réseaux de neurones ont été largement utilisés pour résoudre des problèmes pratiques tels que la classification, la régression, etc. Ces modèles d'apprentissage automatique peuvent tous être adaptés pour résoudre le problème de la prédiction des *ratings*, *i.e.* la complétion de la matrice de *ratings*. En effet, on peut considérer la prédiction des *ratings* comme une généralisation des autres problématiques évoquées. Dans un cadre de classification, par exemple, il s'agit également de compléter une matrice, dans laquelle tous les champs sauf ceux de la dernière colonne (la variable de classe) sont remplis par des valeurs spécifiques (*i.e.* les caractéristiques des jeux de données). Plus généralement, dans le cas du filtrage collaboratif, chaque case de la matrice dont on dispose peut être vide. Par conséquent, le problème de la prédiction des *ratings* peut être considéré comme une généralisation de la classification ou de la régression.

De nombreux modèles ont été appliqués dans le contexte du filtrage collaboratif pour prédire les *ratings*. On peut citer les *arbres de décision*, les modèles basés sur les *règles d'association*, le modèle dit *bayésien naïf*, les modèles basés sur la factorisation des matrices (également connu sous la dénomination de *facteur latent*) et les modèles basés sur les réseaux de neurones (profonds). Le présent manuscrit n'a pas pour objectif de détailler tous ces modèles, et nous présenterons dans les sous-sections suivantes uniquement quelques modèles populaires dans le domaine afin de montrer l'intérêt des approches de CF basées sur les modèles. Le lecteur intéressé par d'autres modèles de prédiction dans ce contexte peut se référer à [Aggarwal, 2016f] pour plus de détails.

2.1.3.2.1 Modèle *baseline* Le modèle *baseline*, également connu comme « modèle basé sur le *biais* », est l'un des modèles de référence dans le contexte du filtrage collaboratif [Koren, 2008; Koren and Bell, 2015]. L'idée principale de ce modèle est basée sur l'hypothèse suivante : « les utilisateurs et les items sont deux éléments indépendants qui influencent le plus les valeurs des *ratings*. Plus précisément, la valeur de $r_{u,i}$ représente la moyenne globale de l'ensemble des *ratings*, biaisée par l'influence propre de l'utilisateur u et celle de l'item i. » Autrement dit, la valeur de *rating* $r_{u,i}$ ne dépend que de l'utilisateur u et de l'item i, indépendamment de l'interaction entre u et i.

Formellement, étant donnée μ, la moyenne globale des *ratings* observée sur l'ensemble

des *ratings* dans un système de recommandation, le modèle *baseline* est défini par :

$$b_{u,i} = \mu + b_u + b_i \tag{2.19}$$

où le terme b_u représente l'effet de l'utilisateur u et le terme b_i représente celui de l'item i. Ces deux termes b_u et b_i peuvent être interprétés comme les biais introduits par l'utilisateur u et l'item i à la moyenne globale de *rating*, *i.e.* μ.

Plus concrètement, considérons l'exemple suivant. Supposons que l'on cherche à prédire la valeur de *rating* de l'utilisatrice *Alice* pour le film « Intouchables » avec le modèle *baseline*. Supposons de plus que la moyenne des notes dont on dispose (tous films et utilisateurs confondus) est de 4.2/5. Enfin, supposons d'une part qu'*Alice* est une utilisatrice critique, qui a tendance à attribuer à un film une note inférieure de 0.5 point par rapport à la moyenne des notes des autres utilisateurs, et, d'autre part, que le film « Intouchables » est très populaire et a tendance à être noté par les utilisateurs avec une note supérieure de 0.3 point par rapport à la moyenne de leurs notes sur les autres films. Alors, d'après le modèle *baseline*, $\mu = 4.2$, $b_u = -0.5$ et $b_i = 0.3$. Ainsi, la valeur de *rating* prédite serait :

$$\hat{r}_{Alice,Intouchables} = 4.2 - 0.5 + 0.3 = 4. \tag{2.20}$$

On peut considérer l'estimation des paramètres b_u et b_i, comme un problème d'optimisation. Notons que lorsque l'on parle de paramètres b_u et b_i, on parle en fait de $|U| + |I|$ paramètres à ajuster, puisque l'on a un paramètre b_u par utilisateur et un paramètre b_i par item. Ce nombre de paramètres élevé peut conduire à un problème de surapprentissage (*overfitting*, *i.e.* le modèle entraîné correspond trop précisément au jeu de données d'entraînement et n'arrivera pas à conduire des prédictions précises dans un autre jeu de données). Pour les optimiser on peut, par exemple, utiliser l'approche des moindres carrés, en incluant un terme de régularisation pour éviter le surapprentissage, comme illustré par l'équation (2.21).

$$\underset{b_u,b_i}{\operatorname{argmin}}\ J = \sum_{r_{u,i} \in R_{train}} (r_{u,i} - \mu - b_u - b_i)^2 + \lambda(b_u^2 + b_i^2) \tag{2.21}$$

Le premier terme $(r_{u,i} - \mu - b_u - b_i)^2$, que l'on peut réécrire comme $(r_{u,i} - b_{u,i})^2$, vise à trouver les b_u et b_i qui correspondent au mieux aux *ratings* observés dans le jeu de données. Le terme de régularisation, $\lambda(b_u^2 + b_i^2)$, permet d'éviter le surapprentissage en favorisant le fait d'avoir de nombreuses valeurs de b_u et de b_i très proches de ou égales à 0 (ce qui fait autant de paramètres en moins dans le modèle). Le terme R_{train} représente les *ratings* observés dans le système de recommandation et donc disponibles pour entraîner (*train*) le modèle ; il s'agit souvent du jeu de données avec lequel on construit le modèle.

D'un point de vue technique, l'estimation des valeurs pour les paramètres b_u et b_i du modèle *baseline* peut être effectuée efficacement avec des algorithmes d'optimisation tels

que l'algorithme de descente de gradient stochastique – ou *Stochastic Gradient Descent* (SGD) [Ruder, 2016].

L'objectif de l'algorithme de SGD est de déterminer les valeurs des paramètres d'un modèle (b_u et b_i dans ce cas) pour que la valeur résultante de la fonction objectif (J dans ce cas) soit la plus faible possible. Pour ce faire, l'algorithme s'appuie sur la dérivée de la fonction objectif en ce qui concerne les paramètres.

Supposons que l'on cherche à déterminer la valeur optimale pour le paramètre b_u étant donné l'ensemble des *ratings* observé R_{train} et la fonction objectif J, définie par l'équation (2.21). La dérivée de la fonction J par rapport à la variable b_u est calculée comme suit :

$$\begin{aligned}\frac{\partial J}{\partial b_u} &= \sum_{r_{u,i} \in R_{train}} -2(r_{u,i} - \mu - b_u - b_i) + 2\lambda b_u \\ &= \sum_{r_{u,i} \in R_{train}} -2(r_{u,i} - b_{u,i} - \lambda b_u)\end{aligned}$$

Notons que le calcul de la dérivée ci-dessus considère tous les *ratings* (puisqu'on calcule la somme). C'est la manière de procéder de l'algorithme classique de descente de gradient. Alors que l'algorithme de SGD, ne considère généralement qu'une seule instance (parmi tous les $r_{u,i}$), ou seulement un petit ensemble d'instances (*mini-batch*), durant chaque itération d'apprentissage. Dans le cas de l'optimisation par l'algorithme de SGD, la dérivée devient :

$$\begin{aligned}\frac{\partial J}{\partial b_u} &= -2(r_{u,i} - \mu - b_u - b_i) + 2\lambda b_u \\ &= -2(r_{u,i} - b_{u,i} - \lambda b_u)\end{aligned}$$

L'algorithme de SGD procède de manière itérative pour déterminer les valeurs optimales des paramètres. A chaque itération, une instance des *ratings* (ou des instances, suivant la taille de *mini-batch*) est utilisée pour mettre à jour la valeur des paramètres simultanément. Considérant l'exemple de mise à jour pour le paramètre b_u, la règle est la suivante :

$$\begin{aligned}b_u &\leftarrow b_u - \gamma \cdot \frac{\partial J}{\partial b_u} \\ &\leftarrow b_u - \gamma \cdot (-2(r_{u,i} - b_{u,i} - \lambda b_u)) \\ &\leftarrow b_u + 2\gamma \cdot (r_{u,i} - b_{u,i} - \lambda b_u)\end{aligned}$$

où γ (taux d'apprentissage) est un hyper-paramètre qui détermine la vitesse d'optimisation. De même, la règle de mise à jour pour le paramètre b_i est la suivante :

$$b_i \leftarrow b_i + 2\gamma \cdot (r_{u,i} - b_{u,i} - \lambda b_i).$$

L'algorithme de SGD s'arrête lorsque la valeur des erreurs de prédiction ($r_{u,i} - b_{u,i}$) cesse de diminuer ou qu'un nombre maximal de n itérations est atteint, ce qui constitue un autre hyper-paramètre de l'optimisation.

2.1.3.2.2 Modèle SVD (*Singular Value Decomposition*) Il existe dans la littérature une autre famille de modèles de filtrage collaboratif appelée : « modèles basés sur les facteurs latents ». Ces modèles utilisent généralement la factorisation matricielle, également connue sous l'expression *Singular Value Decomposition* (SVD).

Dans le domaine de la recherche d'information, la SVD est bien établie pour identifier les facteurs sémantiques latents [Deerwester et al., 1990]. Pour autant, dans le contexte du CF, la matrice de *ratings* présentant de nombreuses valeurs vides, il est difficile d'appliquer directement la SVD à la matrice initiale. De plus, ne considérer que trop peu de *ratings* explicites peut conduire au surapprentissage. Pour y faire face, des solutions anciennes, comme par exemple l'imputation de la matrice de *ratings*, proposent de remplir la matrice afin de la rendre plus dense [Sarwar et al., 2000]. Par contre, cette imputation de données manquantes peut devenir *lourde* car elle augmente la taille des données d'une part, et d'autre part les données imputées peuvent être déformées à cause d'une imputation imprécise. De ce fait, les méthodes plus récentes (que nous détaillons dans la suite) préfèrent ne modéliser que les *ratings* explicites qui sont effectivement observés dans le jeu de données et utilisent la *régularisation* pour éviter le surapprentissage [Koren, 2008; Koren et al., 2009; Koren and Bell, 2015].

Le modèle SVD est parmi les modèles de factorisation matricielle les plus populaires dans le contexte du filtrage collaboratif. L'idée principale d'un modèle de factorisation matricielle est de considérer la matrice des *ratings* R comme une approximation de la multiplication de deux matrices, comme montré dans l'équation (2.22) :

$$\mathrm{R} \approx \mathrm{P}\mathrm{Q}^{\mathrm{T}} \tag{2.22}$$

où P est une matrice de taille $|\mathrm{U}| \times k$ et Q est une matrice de taille $|\mathrm{I}| \times k$. Chaque colonne de P (resp. Q) se réfère à un *vecteur latent* et chaque ligne de P (resp. Q) se réfère à un *facteur latent*. Par exemple, la n-ième ligne $\overrightarrow{p_n}$ (resp. $\overrightarrow{q_n}$) de la matrice P (resp. Q) correspond au facteur latent du n-ième utilisateur (resp. n-ième item). Autrement dit, $\overrightarrow{p_n}$ peut être considéré comme une modélisation de l'utilisateur, dont chacune des k valeurs mesure l'affinité de l'utilisateur avec l'un des k concepts latents de la matrice des *ratings*. D'une manière similaire, $\overrightarrow{q_n}$ peut être considéré comme un item dont la valeur de chacun des k dimensions représente l'affinité de l'item avec l'un des k concepts latents de la matrice des *ratings*.

La factorisation matricielle permet de *plonger* les utilisateurs et les items dans un même espace *latent* de dimension k, de telle sorte que les interactions entre les utilisateurs et les items sont modélisées comme des produits scalaires dans l'espace concerné.

Cet espace latent tend à expliquer les *ratings* en caractérisant à la fois les utilisateurs et les items par des concepts inférés à partir des *feedbacks* des utilisateurs. Par exemple, considérons le contexte de la recommandation de films. Les concepts latents pourraient représenter et mesurer les dimensions évidentes telles que : comédie, film d'action, etc. Ils pourraient également représenter des dimensions beaucoup moins parlantes comme la longueur du film voire des dimensions non-interprétables (celles que le modèle a déduit lors de l'entraînement mais qui restent peu interprétable). Notons que le vecteur de facteurs d'un utilisateur, *i.e.*, $\overrightarrow{p_n}$ peut contenir des valeurs positives et négatives, indiquant le taux d'intérêt de l'utilisateur pour ces concepts latents. De même, le vecteur de facteurs d'un item, *i.e.*, $\overrightarrow{q_n}$ peut contenir des valeurs positives et négatives, signifiant dans quelle mesure l'item possède ces concepts latents.

Le modèle SVD repose sur la factorisation matricielle présentée précédemment. Formellement, dans le contexte de SVD, chaque item i est représenté par un vecteur $\mathbf{q_i} \in \mathbb{R}^k$, et chaque utilisateur u est représenté par un vecteur $\mathbf{p_u} \in \mathbb{R}^k$. Le produit scalaire de ces deux vecteurs, *i.e.*, $\mathbf{q_i}^{\mathrm{T}} \cdot \mathbf{p_u}$, capture l'interaction entre l'utilisateur u et l'item i, c'est-à-dire l'intérêt global de l'utilisateur u pour les caractéristiques de l'item i. Finalement, pour prédire la valeur de $\hat{r}_{u,i}$, c'est-à-dire le *rating* que u mettrais à i, le modèle SVD ajoute également les effets individuels de l'utilisateur u et de l'item i (*i.e.*, le modèle *baseline* introduit précédemment). De ce fait, le modèle SVD permet non seulement de considérer les interactions entre les utilisateurs et les items, mais aussi les effets individuels de l'utilisateur et de l'item concerné. Dans ce modèle, la prédiction de $\hat{r}_{u,i}$ est effectuée par l'équation (2.23) suivante :

$$\hat{r}_{u,i} = \mu + b_u + b_i + \mathbf{q_i}^{\mathrm{T}} \cdot \mathbf{p_u} \tag{2.23}$$

De même que dans le cas du modèle *baseline*, les valeurs des paramètre du modèle, *i.e.*, $b_u, b_i, \mathbf{q_i}$ et $\mathbf{p_u}$, doivent être apprises. Pour ce modèle on a $k+1$ paramètres à ajuster par utilisateur et par item, soit : $(k+1)(|\mathrm{U}|+|\mathrm{I}|)$ paramètres au total. Avec un tel nombre de paramètres, il y a un fort risque de surapprentissage ; il est donc important de contrôler ce facteur en favorisant le fait de mettre à 0 un grand nombre de ces paramètres. Pour ce faire, l'ajustement des paramètres se fait en optimisant la fonction objectif suivante (équation (2.24)) :

$$\underset{b_u, b_i, \mathbf{q_i}, \mathbf{p_u}}{\operatorname{argmin}} \mathrm{J} = \sum_{r_{u,i} \in \mathrm{R}_{train}} (r_{u,i} - \mu - b_u - b_i - \mathbf{q_i}^{\mathrm{T}} \cdot \mathbf{p_u})^2 + \lambda (b_u^2 + b_i^2 + ||\mathbf{q_i}||^2 + ||\mathbf{p_u}||^2) \tag{2.24}$$

où la constante λ représente le coefficient de régularisation, dont la valeur, souvent déterminée par la validation croisée, permet d'éviter le surapprentissage du modèle.

2.1.3.2.3 Modèle SVD++ Le modèle SVD++ est une extension du modèle SVD qui vise à capturer et intégrer dans les prédictions les informations implicites relatives aux interactions *utilisateur-item* [Koren, 2008]. Les retours implicites des utilisateurs, introduits dans la section 2.1.2, pourraient être particulièrement utiles dans les cas où des utilisateurs ont fourni très peu de *ratings* explicites ou lorsque le système de recommandation ne dispose pas de moyen pour ses utilisateurs de saisir des *ratings* explicites. Par exemple, Youtube ne demande pas à ses utilisateurs de noter explicitement une vidéo, mais se base sur la durée et le nombre de visionnages. Idéalement, nous aimerions pouvoir collecter différents types de retours implicites, comme par exemple l'historique des achats des utilisateurs sur un site marchand, ou bien l'historique des films regardés sur Netflix, etc. En revanche, ce genre de données n'est souvent pas disponibles au grand public pour des raisons de protections des données personnelles. Dans la littérature, la solution fréquemment retenue par des chercheurs consiste à transformer les *ratings* explicites en *feedbacks* implicites. Concrètement, il s'agit de transformer la matrice des *ratings* explicites R en une matrice binaire R' de même dimension, dont la valeur à l'indice (u, i) vaut 1 si l'utilisateur u a explicitement noté i (quelle que soit la valeur de sa note) et 0 sinon. La matrice implicite R' est ensuite normalisée de manière à ce que la norme de *Frobenius* (L_2-Norm) de chaque ligne de la matrice R' soit égale à 1. Pour ce faire, étant donné $R'(u)$ le vecteur ligne correspondant à l'utilisateur u, chaque valeur de $R'(u)$ est normalisée en la divisant par la L2-Norme de $R'(u)$, *i.e.*, par la racine carrée du nombre d'items évalués par u, *i.e.* $|R'(u)|^{\frac{1}{2}}$. Un exemple concret est représenté dans la figure 2.5.

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

(a) R

	i_1	i_2	i_3	i_4
u_1	$1/\sqrt{2}$			$1/\sqrt{2}$
u_2		$1/\sqrt{1}$		
u_3	$1/\sqrt{3}$		$1/\sqrt{3}$	$1/\sqrt{3}$
u_4			$1/\sqrt{1}$	

(b) R'

FIGURE 2.5 – Exemples de transformation d'une matrice de *ratings* explicites (a) en matrice implicite normalisée (b).

Le modèle SVD++ est formellement défini par l'équation (2.25). Basé sur le modèle SVD (voir équation (2.23)), SVD++ permet de considérer à la fois les *ratings* explicites (le terme $\mathbf{p_u}$) et les ratings *implicites* (le terme $|R'(u)|^{-\frac{1}{2}} \sum_{j \in R'(u)} \mathbf{y_j}$). Spécifiquement, dans l'approche de SVD++, chaque item i est associé avec deux vecteurs de facteurs $\mathbf{q_i}$ et $\mathbf{y_i}$, qui représentent respectivement les vecteurs déduits pour les *feedbacks* explicites et implicites. On peut également interpréter le modèle SVD++ comme suit : le vecteur de facteurs pour chaque utilisateur u est composé de deux parties : $\mathbf{p_u}$ d'une part qui est basée sur les *ra-*

tings explicites de l'utilisateur u, et $|R'(u)|^{-\frac{1}{2}} \sum_{j \in R'(u)} \mathbf{y_j}$ d'autre part, qui peut être considéré comme un complément de $\mathbf{p_u}$, et qui prend en considération les *feedbacks* implicites.

$$\hat{r}_{u,i} = \mu + b_u + b_i + \mathbf{q_i}^T (\mathbf{p_u} + |R'(u)|^{-\frac{1}{2}} \sum_{j \in R'(u)} \mathbf{y_j}) \tag{2.25}$$

Afin d'obtenir les valeurs optimales pour ces paramètres du modèle, on cherche à optimiser la fonction objectif suivante (équation (2.26)) :

$$\begin{aligned} \underset{b_u, b_i, \mathbf{q_i}, \mathbf{p_u}, \mathbf{y_j}}{\operatorname{argmin}} \; J = & \sum_{r_{u,i} \in R_{train}} (r_{u,i} - \mu - b_u - b_i - \mathbf{q_i}^T (\mathbf{p_u} + |R'(u)|^{-\frac{1}{2}} \sum_{j \in R'(u)} \mathbf{y_j}))^2 \\ & + \lambda_1 (b_u^2 + b_i^2) + \lambda_2 (||\mathbf{q_i}||^2 + ||\mathbf{p_u}||^2 + ||\mathbf{y_j}||^2) \end{aligned} \tag{2.26}$$

Les études ont montré que la performance du modèle SVD++ est nettement meilleure que celle de SVD, en termes de prédiction des *ratings*. Autrement dit, en plus de l'aspect de *retours explicites* (comme SVD), la prise en compte de l'aspect de *retours implicites* des utilisateurs permet d'obtenir des prédictions plus justes.

2.1.3.2.4 Modèle NeuMF (*Neural Matrix Factorization*) En raison notamment de l'augmentation exponentielle des données sur le Web depuis ces dernières années, les réseaux de neurones artificiels et l'apprentissage profond ont été largement employés dans plusieurs domaines relatifs à l'intelligence artificielle, comme le traitement d'images, la vision par ordinateur, la reconnaissance vocale, ou le traitement automatique du langage naturel [Liu et al., 2017]. Le domaine de la recommandation ne fait pas exception, et il est lui aussi fortement impacté par ces évolutions technologiques [Ferrari Dacrema et al., 2019; Zhang et al., 2019]. L'apprentissage profond permet de capturer efficacement les relations non-linéaires et non-triviales entre les utilisateurs et les items, et de découvrir des relations plus abstraites et complexes à partir des données d'interaction entre les utilisateurs et items. De nombreuses approches de recommandation basées sur les réseaux de neurones ont ainsi été proposées [Sedhain et al., 2015; He et al., 2017; Guo et al., 2017; Wu et al., 2017; Liang et al., 2018]. Dans cette section, nous introduisons et détaillons le modèle NeuMF [He et al., 2017] qui est l'une des approches de recommandation basées sur les réseaux de neurones profonds les plus citées. Le lecteur intéressé peut se référer à [Zhang et al., 2019] pour une vision plus globale de l'ensemble des modèles de recommandation basés sur les réseaux de neurones et l'apprentissage profond.

NeuMF est un modèle de filtrage collaboratif basé sur les réseaux de neurones profonds. Il s'agit d'une approche qui vise à coupler la technique de factorisation matricielle présentée dans les sections précédentes et les réseaux de neurones afin de renforcer la performance des modèles basés sur les facteurs latents.

L'architecture générale du NeuMF est schématisée dans la figure 2.6. La couche *Input* (entrée) en bas du schéma représente deux vecteurs qui décrivent respectivement l'utili-

sateur u et l'item i. Les auteurs de [He et al., 2017] ont adopté l'encodage *one-hot vector* pour modéliser u et i. Ce dernier est effectué selon l'identifiant de l'utilisateur u et de l'item i (*e.g.* un vecteur de dimension |U| avec 1 dans la case qui correspond à l'identifiant de l'utilisateur cible et 0 pour les U − 1 autres dimensions). Notons que les vecteurs de descriptions des utilisateurs et des items peuvent aussi être basés sur d'autres informations que les *ratings* si elles sont disponibles, *e.g.*, les contenus des items. Au dessus de la couche *Input* se trouve la couche *Embedding* (plongement). Il s'agit souvent d'une *couche entièrement connectée*[4] (*fully connected layer*) qui projette la représentation de départ (généralement très creuse, *e.g.* vecteur de |U| dimensions composant de |U| − 1 zéros et d'un 1 pour un utilisateur donné) dans un vecteur plus dense (64 ou 128 dimensions en général). Les plongements obtenus respectivement pour l'utilisateur u et pour l'item i sont analogues aux vecteurs latents du modèle des facteurs latents discuté précédemment. La couche *Neural CF* qui vient ensuite, *i.e.*, une architecture de neurones multi-couche, est utilisée afin d'aligner les vecteurs latents (couche *Embedding*) aux scores des prédictions. Enfin, la dernière couche *Output* (sortie) est la prédiction effectuée $\hat{r}_{u,i}$. L'entraînement du modèle est généralement effectué en minimisant l'erreur (*Loss*) entre la prédiction $\hat{r}_{u,i}$ et la vraie valeur de $r_{u,i}$.

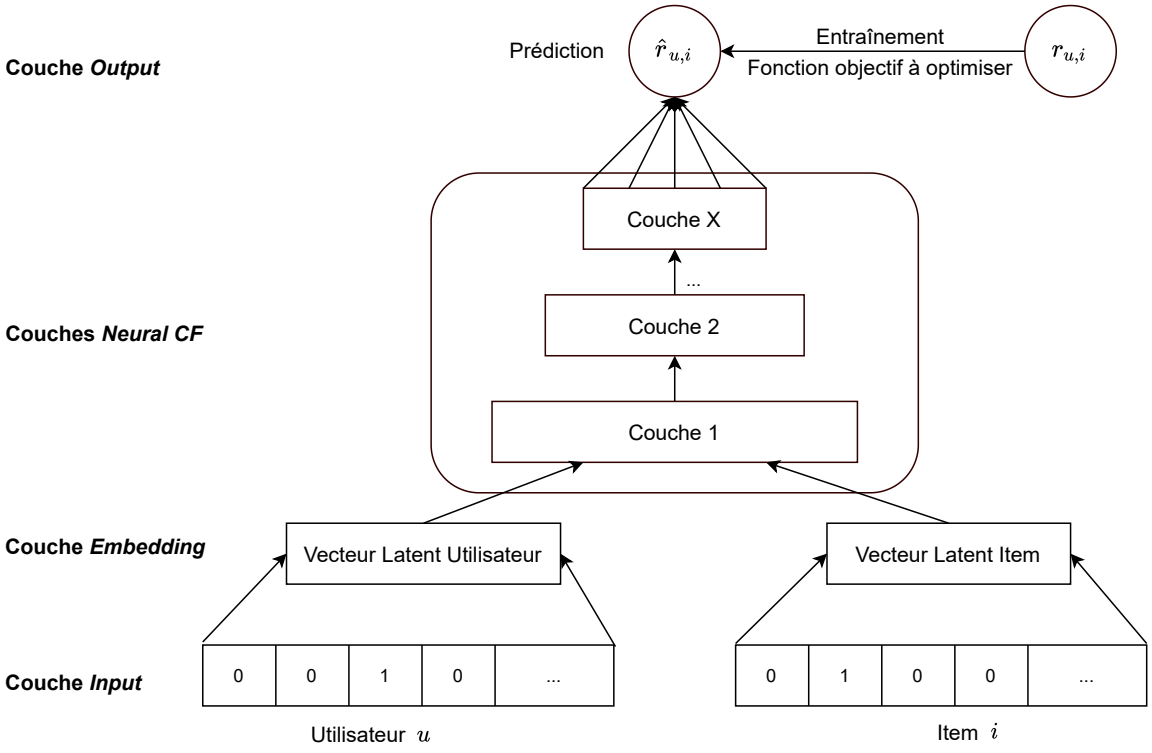


FIGURE 2.6 – Architecture générale du modèle NeuMF. Source : [Zhang et al., 2019]

Dans le contexte du filtrage collaboratif, comme montré dans la section précédente

[4]Une couche entièrement connectée est une structure couramment utilisée dans les réseaux de neurones convolutifs. Il s'agit de réseaux de neurones dans lesquels tous les neurones de la couche courante sont connectés avec tous les neurones de la couche suivante. https://datafranca.org/wiki/Couche_enti%C3%A8rement_connect%C3%A9e

liée au modèle SVD++ (voir section 2.1.3.2.3), deux modélisations sont possibles, basées respectivement sur les *feedbacks* explicites ou implicites. L'architecture de neurones présentée ci-dessus peut s'adapter automatiquement en fonction du type de données (explicites ou implicites) utilisé pendant l'entraînement. Il s'agit seulement de choisir la fonction objectif appropriée selon le contexte (*feedback* explicite ou implicite). Dans le contexte du *feedback* explicite, la fonction objectif peut être définie pour minimiser l'erreur quadratique entre la prédiction $\hat{r}_{u,i}$ et la vraie valeur de *rating* $r_{u,i}$, comme dans le cas des modèles du facteur latent. Dans [He et al., 2017], les auteurs ont choisi d'apprendre les paramètres du modèle dans le contexte des *feedbacks* implicites. Autrement dit, $r_{u,i} = 1$ si l'interaction entre l'utilisateur u et l'item i est observée dans le jeu de données des *ratings*, et $r_{u,i} = 0$ sinon. De ce fait, on peut voir $r_{u,i}$ comme un label : 1 signifie que l'item i est pertinent pour l'utilisateur u, et 0 sinon. La prédiction $\hat{r}_{u,i}$ représente ainsi la probabilité que l'item i soit pertinent pour l'utilisateur u. On note R' l'ensemble des instances de *ratings* positives, *i.e.*, les interactions (u,i) observées dans le jeu de données, et R'^{-} l'ensemble des items avec lesquels l'utilisateur u n'a pas interagit, *i.e.*, les instances négatives. On souhaite que les *ratings* prédits pour des éléments de R' soient proches de 1 et ceux prédits pour des éléments de R'^{-} soient proches de 0. L'utilisation de la fonction logarithmique (log) permet de pénaliser fortement les écarts importants à ces prédictions attendues. De ce fait, dans le contexte du *feedback* implicite, la fonction objectif J à minimiser pour apprendre les paramètres du réseau de neurones est généralement l'entropie croisée (*cross-entropy loss*) :

$$\begin{aligned} \text{Minimise J} &= -\sum_{(u,i)\in \mathrm{R}'} \log \hat{r}_{u,i} - \sum_{(u,j)\in \mathrm{R}'^{-}} \log(1-\hat{r}_{u,j}) \\ &= -\sum_{(u,i)\in \mathrm{R}' \bigcup \mathrm{R}'^{-}} r_{u,i} \log \hat{r}_{u,i} + (1-r_{u,i}) \log(1-\hat{r}_{u,i}) \end{aligned} \tag{2.27}$$

De manière plus spécifique, le modèle de NeuMF proposé dans [He et al., 2017] combine deux architectures de réseaux de neurones : *Generalised Matrix Factorization* (GMF) et *Multi-Layer Perceptron* (MLP), comme illustré dans la figure 2.7. L'idée principale de l'architecture de neurones de GMF est de généraliser les modèles basés sur la factorisation matricielle, *i.e.*, $\mathbf{p_u} \cdot \mathbf{q_i}$, dans laquelle i) les facteurs latents sont traités de la même manière (avec les même poids pour chaque facteur latent) et ii) une fonction linéaire (le produit scalaire) est utilisée pour modéliser l'interaction entre l'utilisateur et l'item. Le modèle de GMF peut automatiquement apprendre différents poids pour chaque facteur latent et envisager des liens non-linéaires entre les utilisateurs et les items. La non-linéarité peut être obtenue par une fonction d'activation non-linaire (*e.g.*, sigmoïde) sur le produit scalaire. Le réseau de neurone de MLP, contrairement à celui de GMF qui n'utilise qu'un produit fixé de $\mathbf{p_u} \cdot \mathbf{q_i}$, permet d'apprendre des interactions plus subtiles entre utilisateurs et items. Cela est rendu possible par l'ajout de couches cachées (*hidden layers*) sur le vec-

teur concaténé de $\mathbf{p_u}$ et de $\mathbf{q_i}$. Enfin, les dernières couches cachées de GMF et MLP sont concaténées pour générer la prédiction finale du *rating* d'un item par un utilisateur.

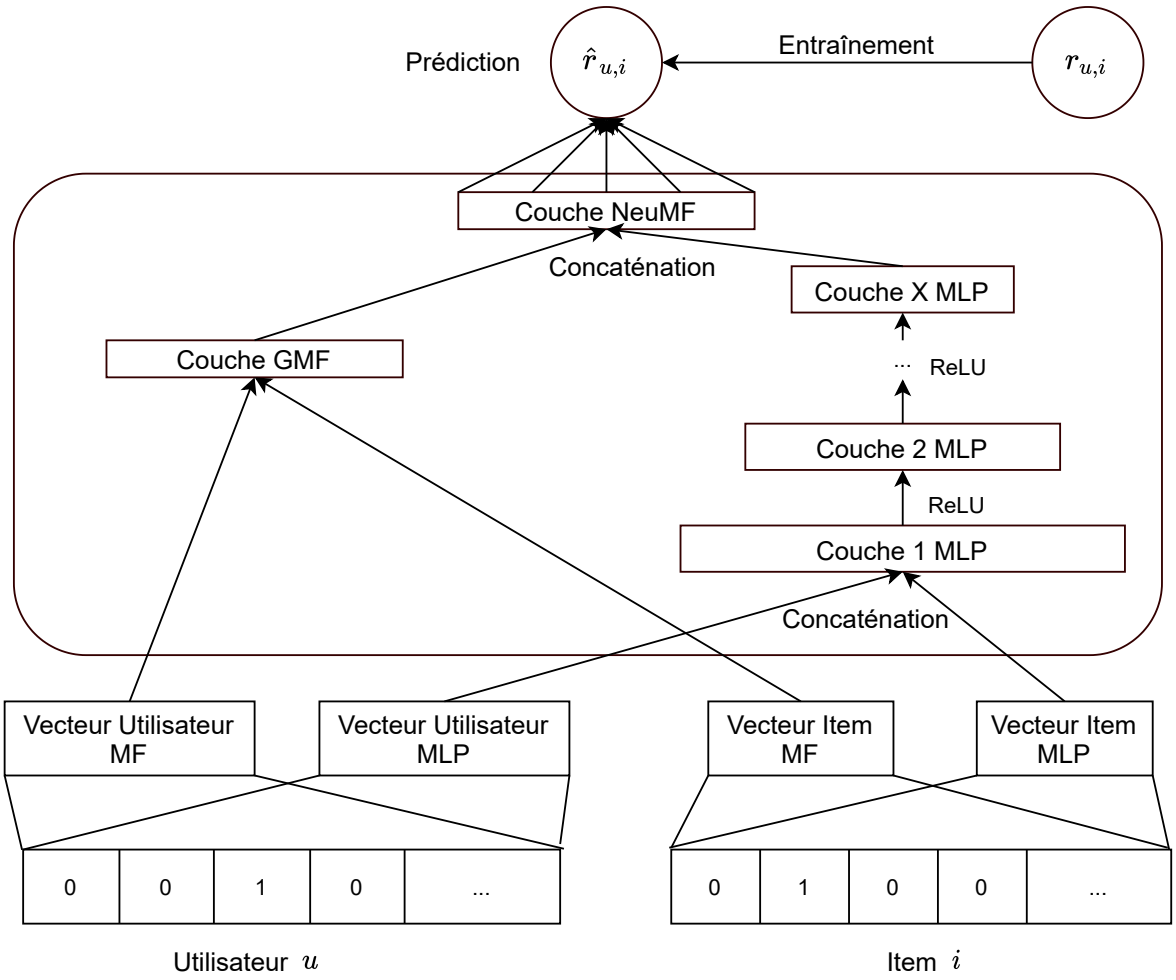


FIGURE 2.7 – Architecture complet du modèle NeuMF. Source : [He et al., 2017]

2.1.4 Approches basées sur le contenu

« Si Alice aime regarder le film *Transformers*, il semble pertinent de lui recommander le film *Transformers 2 : La Revanche* ou d'autres films du genre *Science Fiction* ». L'hypothèse sous-jacente est que les utilisateurs ont une forte probabilité d'aimer les items qui sont similaires à ceux qu'ils ont aimés auparavant, en termes de contenu. Cette section s'intéresse à un autre type d'approche de recommandation, *i.e.*, les approches de filtrage basé sur le contenu – ou *Content-Based Filtering* (CBF).

Contrairement aux systèmes de recommandation basés sur le filtrage collaboratif introduits dans la section précédente, et qui utilisent les corrélations des systèmes de notations entre utilisateurs pour faire les recommandations, les approches de CBF se focalisent sur les *feedbacks* individuels d'un seul utilisateur, *i.e.* celui pour qui on va faire des recommandations [Lops et al., 2011; De Gemmis et al., 2015; Aggarwal, 2016a]. En plus des retours de cet utilisateur cible, pour que les approches de CBF fonctionnent, il est indis-

pensable de disposer de données relatives aux contenus des items. Ces dernières peuvent être des données de différents types, structurées ou non-structurées selon la source des données. Par exemple, il peut s'agir de descriptions des items sous forme de textes (dans la plupart des SdR basés sur le contenu) ou de manière plus structurée, sous la forme d'un ensemble de mot-clés (éventuellement hiérarchisés). Il peut s'agir également d'attributs (caractéristiques) des items stockés dans des basés de données relationnelles, *e.g.*, le prix d'un produit, sa couleur, sa taille, etc.

De manière générale, les approches de recommandation basées sur le contenu des items procèdent en trois phases. Premièrement, disposant d'un ensemble de données relatives aux caractéristiques des items (attributs ou contenus) et quels que soient leurs sources ou leurs types, les SdR basés sur le CBF analysent ces données afin de construire une représentation structurée (généralement sous la forme d'une modélisation vectorielle) de chaque item présent dans le catalogue du SdR. Il s'agit souvent d'une phase de pré-traitement qui vise à extraire des informations pertinentes permettant de décrire les items. Ensuite, en se focalisant sur les *feedbacks* individuels de l'utilisateur cible, la deuxième phase vise à construire un profil pour l'utilisateur, qui révèle ses intérêts relatifs à chacune des caractéristiques des items. Enfin, la troisième phase consiste à comparer le profil de cet utilisateur avec les caractéristiques correspondant aux items du catalogue afin d'identifier ceux qui sont les plus pertinents pour lui. Il s'agit donc d'une phase de filtrage des items pour exclure ceux dont les attributs (contenus) ne correspondent pas au profil utilisateur.

Le processus de recommandation des approches par CBF est illustré par la figure 2.8. Les sections qui suivent décrivent chacune des trois phases introduites ci-dessus et représentées par des rectangles bleutés sur la figure.

2.1.4.1 Pré-traitement et analyse

Lorsque les données collectées ne sont pas structurées (par exemple composées essentiellement de textes), une étape de pré-traitement est nécessaire afin d'extraire des informations pertinentes sous une forme structurée. L'objectif de cette étape est de représenter les contenus des items du catalogue (par exemple des documents, des pages web, des descriptions de produits, etc.) sous une forme adaptée aux étapes de traitement ultérieures. Les contenus des items sont analysés par des techniques d'extraction de caractéristiques et modélisés dans un espace qui permettra ensuite de construire des profils utilisateurs (*e.g.*, les pages web peuvent être représentées comme des vecteurs de mots-clés) sur lesquels la recommandation pourra être calculée.

Généralement, dans le contexte des SdR basés sur le contenu (CBF), un item est représenté par un vecteur dont chaque dimension correspond à un attribut ou à une caractéristique et dont la valeur reflète l'importance (le poids) de cette caractéristique pour cet item. Par exemple, dans le domaine de la recommandation cinématographique, ces

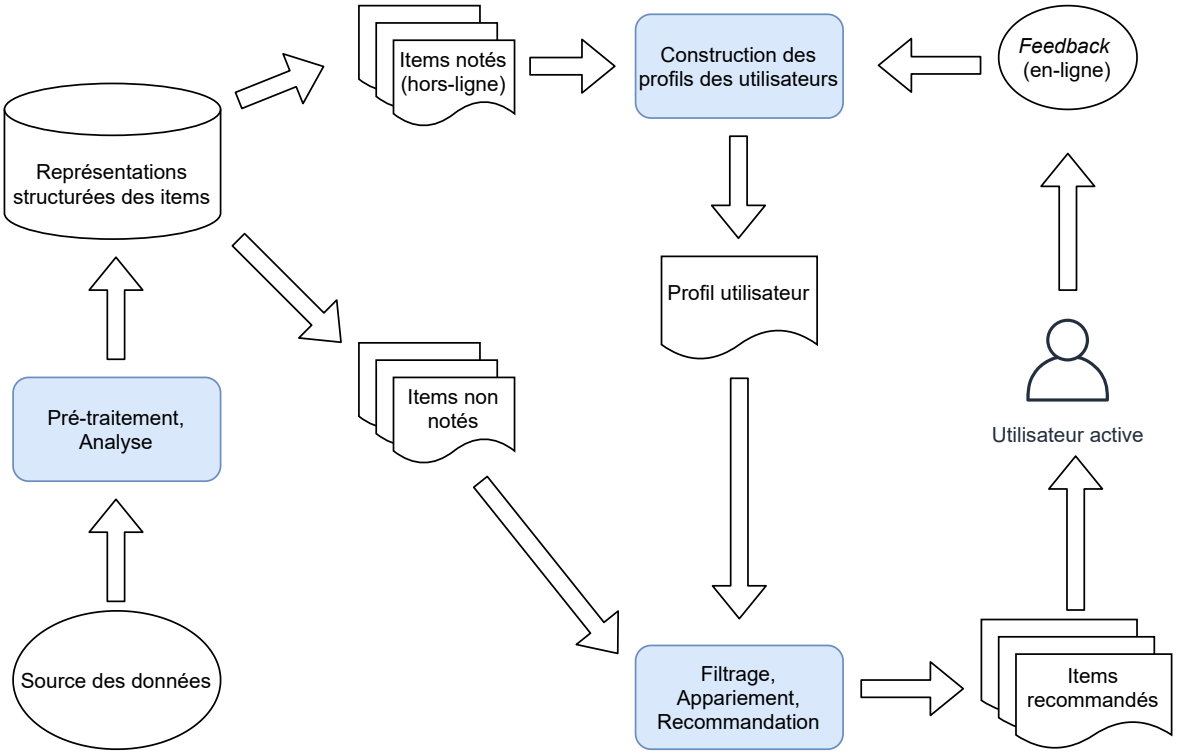


FIGURE 2.8 – Architecture générale des approches de recommandation basées sur le contenu

caractéristiques peuvent être des acteurs, des genres, etc. Le système le plus largement répandu pour pondérer les attributs d'un item s'appuie sur le TF-IDF *(term frequency – inverse document frequency)*, une méthode classique dans le domaine de la recherche d'information [Crnic, 2011].

En Recherche d'Information, chaque document est représenté par un vecteur dans un espace à n dimensions, n étant la taille du vocabulaire employé dans le domaine [Salton et al., 1975]. Ce vocabulaire est formé des termes les plus discriminants du domaine, extraits à partir de la collection complète des documents. Ces termes peuvent être extraits par des techniques classiques de traitement automatique du langage naturel : analyse lexicale, exclusion des mots vides (*stop words*), lemmatisation, etc. Chaque dimension du vecteur associé à un document représente un terme de ce vocabulaire. La valeur d'une dimension représente le degré d'importance du terme correspondant dans la description du contenu du document en fonction des statistiques d'usage de ce terme dans la collection. Formellement, notons $D = \{d_1, d_2, ..., d_{|D|}\}$ l'ensemble des documents et $T = \{t_1, t_2, ..., t_{|T|}\}$ l'ensemble des termes extraits. Chaque document $d_i \in D$ est représenté par un vecteur de dimension $|T|$, *i.e.* $d_i = [w_{1_i}, w_{2_i}, ..., w_{|T|_i}]$ avec w_{k_i} ($k \in \{1, 2, ..., |T|\}$) étant le poids du terme t_k dans le document d_i.

TF-IDF (*Term Frequency - Inverse Document Frequency*) est une pondération très populaire pour déterminer les poids des termes pour chaque document. L'hypothèse est que les termes qui apparaissent fréquemment dans un document (TF élevé) et n'apparaissent pas souvent dans l'ensemble du corpus sont probablement plus pertinents (discriminants) que les autres pour caractériser ce document. La pondération du terme t_k

pour le document d_i calculée par TF-IDF est définie par l'équation (2.28) :

$$\begin{aligned} &\text{TF-IDF}(t_k, d_i) = \text{TF}(t_k, d_i) * \text{IDF}(t_k) \\ &\text{TF}(t_k, d_i) = \frac{f_{t_k}(d_i)}{f_{max}(d_i)} \\ &\text{IDF}(t_k) = \log \frac{|\text{D}|}{n_{t_k}} \end{aligned} \tag{2.28}$$

où $f_{t_k}(d_i)$ désigne le nombre de fois où le terme t_k apparaît dans le document d_i, $f_{max}(d_i)$ représente la fréquence du terme le plus fréquent dans d_i, et n_{t_k} dénote le nombre de documents dans lesquels le terme t_k apparaît.

En transposant cette méthode au domaine des SdR, on note que les vecteurs de caractéristiques basés sur les textes posent un certain nombre de difficultés, principalement en raison de l'ambiguïté du langage naturel. Le problème est que des mots-clés ne suffisent pas à capturer la sémantique des intérêts des utilisateurs, parce qu'ils sont principalement conduits par une opération de correspondance entre chaînes de caractères. Une représentation qui prenne en compte la sémantique des mots-clés permettrait une meilleure modélisation, notamment lorsqu'on construit les profils utilisateurs. Nous rediscuterons ce point dans la section 2.2, qui introduit les SdR basés sur l'ingénierie des connaissances.

2.1.4.2 Construction des profils des utilisateurs

La première phase de pré-traitement et d'analyse permet d'obtenir une représentation vectorielle pour chaque item du catalogue. Ces représentations servent de point d'entrée à la deuxième phase de traitement, dont l'objectif est de construire les profils utilisateurs, étant donnés leur historique de notation (leurs *ratings*) et les représentations structurées des items obtenues à l'issue de la première phase.

La construction des profils utilisateurs s'apparente fortement à une problématique de classification ou de régression dans le domaine de l'apprentissage automatique. En effet, lorsque l'on traite les *ratings* comme des valeurs discrètes (*e.g.*, aime, n'aime pas), le problème correspond à un scénario de classification, alors que si l'on considère les *ratings* comme des valeurs continues, la question s'apparente à un problème de régression.

Dans tous les cas de figure, l'ensemble des items avec lesquels l'utilisateur a interagi (les "items notés" sur la figure 2.8) constitue un jeu d'entraînement, dont les *ratings* sont les labels des instances fournis par l'utilisateur cible. Notons ici que les labels des items fournis par d'autres utilisateurs sont exclus étant donné que chaque utilisateur est considéré indépendamment, contrairement aux approches de filtrage collaboratif. De plus, nous disposons également d'un ensemble d'items non-notés dont les labels sont inconnus. Comme dans le cas du filtrage collaboratif, le jeu d'entraînement est utilisé pour construire un profil (modèle) pour chaque utilisateur. Ensuite, pour chaque item non-

noté, le profil utilisateur est utilisé afin de prédire son intérêt pour cet item (un score dans le cas de régression ou la classe dans le cas de classification).

Les modèles classiques de classification et de régression peuvent être utilisés pour construire les profils utilisateurs, *e.g.*, modèle des k plus proches voisins, arbre de décision, modèle de SVM, modèle bayésien, modèle de la régression linéaire etc. Le jeu de données d'entraînement est utilisé pour déterminer les paramètres des modèles utilisateur avant de pouvoir procéder à la prédiction et à la recommandation. La phase de construction du profil utilisateur est suivie par la dernière phase de filtrage et de recommandation à proprement parler. Ces deux phases sont, en général, fortement liées. Autrement dit, la phase de prédiction dépend fortement du modèle choisi durant la construction du profil utilisateur pendant l'entraînement. Dans la section suivante, on discutera quelques modèles largement utilisés dans la littérature dans le contexte du filtrage basé sur le contenu.

2.1.4.3 Filtrage, appariement et recommandation

On introduit ici deux modèles qui sont largement utilisés dans la littérature : le modèle dit des k plus proches voisins et le modèle de la régression linaire.

2.1.4.3.1 Modèle des k plus proches voisins Il s'agit de l'un des modèles de classification les plus simples et intuitifs de la littérature. La première étape consiste à définir un voisinage à l'aide d'une fonction qui permet de mesurer la similarité entre deux items. Comme les items sont représentés par des vecteurs de caractéristiques, des métriques de similarité classiques qui mesurent la distance entre deux vecteurs sont généralement utilisées, *e.g.*, la similarité cosinus, la distance euclidienne, etc.

Cette fonction de similarité est utilisée pour faire des prédictions concernant les items pour lesquels les goûts de l'utilisateur actif sont inconnus. Formellement, notons I_u comme l'ensemble des items notés par l'utilisateur focal u, ainsi, $\mathrm{I} \backslash \mathrm{I}_u$ représente l'ensemble des items non-notés par u. Pour chaque item $i \in \mathrm{I} \backslash \mathrm{I}_u$, l'ensemble des k items $j \in \mathrm{I}_u$ les plus proches de l'item i peut être identifié grâce à la fonction de similarité choisie. La valeur moyenne des *ratings* de u pour les items voisins de i est utilisée, dans l'approche KNN, comme prédiction de $r_{u,i}$. Par la suite, les items de $\mathrm{I} \backslash \mathrm{I}_u$ ayant les meilleurs scores (prédits) sont ceux recommandés à l'utilisateur u. Dans le cas où l'on cherche à classifier l'item i, (cas où les *ratings* sont discrétisés, *e.g.*, {peu pertinent, pertinent, très pertinent}), on peut calculer la fréquence de chaque classe parmi les k items voisins et la classe la plus fréquente correspondra au label prédit.

2.1.4.3.2 Modèle de régression linaire Un autre modèle largement utilisé est la régression linaire.

Dans ce modèle, on considère la matrice D_c décrivant les items notés par u et qui est donc de taille $|\mathrm{I}_u| \times c$, avec $|\mathrm{I}_u|$ le nombre d'items notés par l'utilisateur u, *i.e.*, la taille du

jeu d'entraînement, et c le nombre de caractéristiques des items. Notons $\overrightarrow{y}$ le vecteur-colonne de dimension $|\mathrm{I}_u|$ contenant les *ratings* des items de I_u. L'idée principale du modèle de régression linaire repose sur l'hypothèse qu'il existe des relations linaires entre les *ratings* ($\overrightarrow{y}$) et les caractéristiques des items (D_c) et que l'on peut considérer que le *rating* peut être approximé par une combinaison linaire de ces caractéristiques. Sous ce modèle, on cherche donc à identifier les coefficients (poids) de chaque caractéristique, qui conduisent aux prédictions les plus proches possibles des valeurs de *ratings* observées. Soit $\overrightarrow{\mathrm{W}}$ un vecteur-ligne de dimension c représentant les coefficients des caractéristiques relatifs à la valeur de *rating*. La régression linaire suppose que

$$\overrightarrow{y} \approx \mathrm{D}_c\overrightarrow{\mathrm{W}}^{\mathrm{T}}$$

et de ce fait, le vecteur $\mathrm{D}_c\overrightarrow{\mathrm{W}}^{\mathrm{T}} - \overrightarrow{y}$ est un vecteur-colonne de dimension $|\mathrm{I}_u|$ représentant les erreurs des prédictions. Afin de maximiser la qualité des prédictions, on cherche à identifier le vecteur de poids $\overrightarrow{\mathrm{W}}$, *i.e.*, les coefficients des caractéristiques, minimisant la fonction objectif J suivante :

$$\underset{\overrightarrow{\mathrm{W}}}{\operatorname{argmin}}\ \mathrm{J} = ||\mathrm{D}_c\overrightarrow{\mathrm{W}}^{\mathrm{T}} - \overrightarrow{y}||^2 + \lambda||\overrightarrow{\mathrm{W}}||^2$$

où $||\cdot||^2$ représente la norme L_2 d'un vecteur et le terme de régularisation λ est ajouté au vecteur des paramètres $||\overrightarrow{\mathrm{W}}||^2$ pour éviter le surapprentissage. Le problème peut être résolu de manière analytique en mettant à 0 le gradient de la fonction J par rapport à $\overrightarrow{\mathrm{W}}$, comme montré dans les équations suivantes (2.29) :

$$\begin{aligned}
&\frac{\partial \mathrm{J}}{\partial \overrightarrow{\mathrm{W}}} = 2\cdot\mathrm{D}_c^{\mathrm{T}}(\mathrm{D}_c\overrightarrow{\mathrm{W}}^{\mathrm{T}} - \overrightarrow{y}) + 2\cdot\lambda\overrightarrow{\mathrm{W}}^{\mathrm{T}} = 0 \\
&(\mathrm{D}c^{\mathrm{T}}\mathrm{D}_c + \lambda\mathrm{I})\overrightarrow{\mathrm{W}}^{\mathrm{T}} = \mathrm{D}_c^{\mathrm{T}}\overrightarrow{y} \\
&\overrightarrow{\mathrm{W}}^{\mathrm{T}} = (\mathrm{D}_c^{\mathrm{T}}\mathrm{D}_c + \lambda\mathrm{I})^{-1}\mathrm{D}_c^{\mathrm{T}}\overrightarrow{y}
\end{aligned} \tag{2.29}$$

où I représente la matrice identité de taille $c \times c$ avec des 1 sur la diagonale et des 0 partout ailleurs.

Enfin, étant donné un item non-noté $i \in \mathrm{I}\backslash\mathrm{I}_u$ et d_i le vecteur de ses caractéristiques, la prédiction des $r_{u,i}$ est obtenue en prenant la $u^{ième}$ valeur du produit scalaire entre le vecteur des coefficients $\overrightarrow{\mathrm{W}}$ et le vecteur descriptif de i, *i.e.*, d_i. Les valeurs prédites sont utilisées par la suite pour ordonner les items non-notés et les items ayant les meilleures prédictions seront recommandés à l'utilisateur focal.

2.1.4.4 Comparaison avec les approches de filtrage collaboratif

Les stratégies sous-jacentes aux approches de recommandation basées sur le contenu (CBF) et aux approches basées sur le filtrage collaboratif (CF) discutée dans la section 2.1.3 comportent des différences fondamentales. Il est donc instructif de les comparer plus avant. Dans cette section, nous discutons des avantages et des inconvénients de l'approche CBF par rapport à l'approche CF.

Voyons tout d'abord les atouts des approches basées sur le contenu par rapport aux approches par filtrage collaboratif :

- *Démarrage à froid* — Les problèmes liés à l'introduction d'un *nouvel item* et/ou d'un *nouvel utilisateur* sont désignés par l'expression « problèmes du démarrage à froid » (*cold-start*), un phénomène répandu dans les systèmes de recommandation. Dans un cas, le système peut avoir des difficultés à recommander les nouveaux items aux utilisateurs. Dans l'autre cas il peut avoir des difficultés à générer des recommandations pour un nouvel utilisateur du système. Étant basées sur des *ratings*, les approches par filtrage collaboratif souffrent particulièrement de ce problème. Au moment où un nouvel utilisateur/item est introduit dans le système, on ne dispose d'aucune information à son sujet, *i.e.*, l'utilisateur n'a pas d'historique de notation et l'item n'a été noté par personne. Dans ce cas, il est impossible d'appliquer le filtrage collaboratif. En revanche, les approches basées sur le contenu n'ont pas de difficultés particulières lors de l'ajout de nouveaux items puisqu'ils sont, dès leur introduction, représentés par leurs vecteurs de caractéristiques, comme tous les autres. Il est donc toujours possible, par exemple, de mesurer la similarité avec d'autres items et donc de faire les recommandations, même si un item ne dispose pas de notation explicite fournie par des utilisateurs.

- *Indépendance par rapport aux utilisateurs* — Un autre avantage est qu'en général, les performances des approches basées sur le contenu ne dépendent pas des retours des autres utilisateurs, contrairement aux approches de CF. En effet, dans un scénario où les utilisateurs notent très peu d'items, la performance des approches CF n'est pas garantie puisque leurs prédictions sont basées sur très peu d'interactions entre les utilisateurs et les items. Cela entraîne, par exemple, une faible fiabilité lorsque l'on identifie les voisins pour les approches NBCF ou des problèmes de surapprentissage pour les méthodes de CF basées sur les modèles. Le manque de *ratings* dans des systèmes de recommandation (matrice de *ratings* très creuse) est connu sous le terme de rareté (*sparsity* en anglais). Il s'agit d'un problème courant dans les SdR basés sur le filtrage collaboratif.

- *Explicabilité* — Les recommandations générées par les approches basées sur le contenu sont simples à expliquer par rapport à celles faites par des modèles de filtrage collaboratif. Comme on dispose dans le premier cas des caractéristiques des items

et que le profil de chaque utilisateur est également basé sur ses intérêts relatifs à ces caractéristiques, il est relativement intuitif d'expliquer les raisons qui ont conduit à la recommandation de certains items à un utilisateur donné. En revanche, les approches basées sur le filtrage collaboratif exploitent la matrice des *ratings*, aucune connaissance liée aux items n'est disponible. Il est donc naturellement plus difficile d'expliquer les recommandations faites par ces approches de CF.

Cependant, les approches basées sur le contenu possèdent les inconvénients suivants :

- *Sur-spécialisation* — L'inconvénient majeur des approches basées sur le contenu est la sur-spécialisation, autrement dit, la difficulté à introduire de la nouveauté, de la diversité dans les recommandations faites à un utilisateur final. En effet, les SdR basés sur le contenu ont tendance à recommander des items qui sont très similaires à ceux que l'utilisateur a déjà vus et qui sont identifiés dans son profil. Ce phénomène est désigné par sérendipité (de l'anglais *serendipity*). Par exemple, si un utilisateur n'a jamais écouté de musique classique, le système basé sur CBF va rarement lui en recommander. En revanche, les recommandations générées par une approche collaborative souffrent moins de la sur-spécialisation, puisqu'elles considèrent les goûts des autres utilisateurs en explorant les items présents dans les profils des voisins. Ainsi, si parmi ces voisins certains utilisateurs ont sélectionné des items *surprenants*, le système de recommandation CF les intègre à son calcul. De ce fait, même si une approche CBF recommande des items particulièrement pertinents pour un utilisateur donné, car proches de son profil, le manque de nouveauté et de diversité peut faire perdre de l'intérêt à cette recommandation. Ce point sera détaillé dans le chapitre 4 qui propose une étude approfondie sur la diversité dans les différents systèmes de recommandation.

- *Analyse inappropriée des contenus* et *pré-traitement lourd* — Contrairement aux approches de filtrage collaboratif, les SdR basés sur le contenu nécessitent une lourde phase de pré-traitement qui vise à collecter, extraire, nettoyer, analyser et structurer des données brutes. L'extraction de caractéristiques (*features*) à partir de données textuelles souffre de limites principalement dues à l'ambiguïté du langage naturel, *e.g.*, la polysémie, la synonymie, etc. Cette problématique rejoint celle de l'analyse des requêtes dans le domaine de la recherche d'information. Un élément de réponse consiste à considérer les connaissances liées au domaine de recommandation afin de couvrir la sémantique des caractéristiques des items. L'ingénierie des connaissances fournit des modèles et des techniques permettant de gérer les connaissances, qui peuvent représenter de réels atouts pour améliorer la modélisation des contenus des items, comme nous en discuterons dans la section 2.2.

- *Nouvel utilisateur* — Même si les systèmes de recommandation basés sur le contenu aident à résoudre les problèmes de démarrage à froid pour les nouveaux items, ils

ne permettent pas de résoudre les problèmes liés aux nouveaux utilisateurs. En effet, lorsqu'un utilisateur a noté très peu d'items, son profil est peu fiable, il est impossible pour le système de connaître ses intérêts par rapport aux caractéristiques des items et la recommandation en souffrira inévitablement.

En dépit de ces inconvénients, les systèmes de recommandation basés sur le contenu complètent souvent assez bien les approches de CF grâce à leur capacité à exploiter les connaissances liées au contenu des items dans le processus de recommandation. Cette complémentarité est souvent mise à profit dans les systèmes de recommandation hybrides qui sont introduits brièvement dans la section suivante, et dans lesquels l'objectif est de combiner les avantages des deux types d'approches pour générer un système de recommandation plus robuste.

2.1.5 Approches hybrides

Dans les deux sections précédentes, *i.e.*, la section 2.1.3 et la section 2.1.4, deux types d'approches de système de recommandation ont été discutés. D'une part, les approches basées sur le filtrage collaboratif qui n'exploitent que les retours des utilisateurs (généralement fournis sous la forme d'une matrice de *ratings*) pour générer leurs recommandations, sans considération du contenu et des caractéristiques propres aux items. D'autre part, les approches basées sur le contenu qui se focalisent sur les seuls intérêts de l'utilisateur ciblé afin de construire son profil en se basant sur les caractéristiques des items.

Pour pailler les inconvénients majeurs de ces deux types d'approche et tirer parti des avantages de chacun, des approches hybrides ont été proposées dans la littérature. Dans les travaux de [Burke, 2002; Aggarwal, 2016c], différentes méthodes d'hybridation ont été introduites, qui visent à combiner deux ou plusieurs techniques de recommandation. Les auteurs de [Çano and Morisio, 2017] ont proposé une revue complète de la littérature de ces systèmes de recommandation hybrides.

Même si de nombreuses méthodes d'hybridation ont été proposées, l'idée principale derrière ces méthodes reste la même : la combinaison de différentes techniques de recommandation et de différents types de données (*ratings*, connaissances liées aux items, données démographiques des utilisateurs, etc.) afin d'améliorer la performance des SdR et de les rendre plus robustes. Dans ce qui suit, quelques méthodes d'hybridation largement utilisées dans la littérature sont brièvement présentées :

- *hybridation pondérée* — L'idée d'une hybridation pondérée est de combiner (généralement de manière linéaire) les scores obtenus par différents systèmes de recommandation. Autrement dit, la prédiction finale du *rating* que l'utilisateur u donnerait à l'item i, *i.e.*, $\hat{r}_{u,i}$, est une combinaison des prédictions faites par différents systèmes de recommandation. Par exemple, les auteurs de [Miranda et al., 1999] proposent une approche qui combine linéairement les prédictions issues d'un CBF

et d'un CF. Les auteurs assignent initialement les mêmes poids pour les scores des deux SdR et proposent d'adapter ensuite ces poids en fonction des *feedbacks* des utilisateurs en-ligne.

- *hybridation commutée* — Dans un SdR hybride de manière *commutée*, le système bascule entre différentes techniques de recommandation selon certains critères. Les auteurs de [Billsus et al., 2000] proposent une approche d'hybridation commutée dans laquelle une approche CBF est utilisée lorsque le nombre des *ratings* est petit, tandis qu'une approche CF est utilisée lorsque le système dispose de suffisamment de retours. Suivant le même principe, [Benouaret, 2017] propose un SdR hybride pour la visite personnalisée de sites culturels. Dans ses travaux, il combine trois approches de recommandation, qui sont utilisées de manière commutée selon le contexte. Plus précisément, une approche *démographique* est initialement utilisée pour faire des recommandations à un nouvel utilisateur et ainsi pallier le problème du démarrage à froid. Ensuite, une approche CBF se basant sur la similarité sémantique est utilisée lorsque l'utilisateur a noté un certain nombre d'items (10 dans leurs travaux). Enfin, une approche collaborative est utilisée lorsque le SdR dispose de plus de retours de l'utilisateur.

- *hybridation mixte* — Il s'agit de présenter une liste d'items qui contient des recommandations de chaque approche utilisée. Les items résultants de chaque approche de recommandation sont présenté simultanément à l'utilisateur [Smyth and Cotter, 2000].

- *hybridation en cascade* — Les hybridations en cascade sont des exemples de processus de recommandation par étapes. Une première technique de recommandation est utilisée pour générer un classement grossier des items candidats, puis une seconde technique permet d'affiner cette première liste de candidats. Notons que cette hybridation est sensible à l'ordre des étapes : un CF-CBF produirait certainement des résultats différents d'un CBF-CF. Par exemple, les auteurs de [Lampropoulos et al., 2012] proposent un SdR hybride (CBF-CF) en cascade dans le domaine musical. Un premier filtrage consiste à extraire les titres musicaux du même genre que ceux aimés par l'utilisateur, ces titres candidats sont ensuite affinées par un filtrage collaboratif avant de les proposer à l'utilisateur.

En général, une approche de recommandation hybride est meilleure qu'une approche simple puisqu'elle considère plus d'informations, combine les avantages des approches simples et pallie leurs inconvénients. Cependant, cela n'est pas toujours vérifié car une mauvaise stratégie d'hybridation peut conduire à combiner les défauts des approches simples. De ce fait, une approche simple de CF basée sur les réseaux de neurones pourraient dépasser une approche hybride si cette dernière n'est pas conçue proprement. Afin

de pouvoir comparer la performance des différentes approches de recommandation, plusieurs métriques d'évaluation sont définies dans la littérature. Nous présentons les plus utilisées dans la section suivante.

2.1.6 Évaluation des systèmes de recommandation

L'objectif d'un système de recommandation est de proposer des items jugés « pertinents » pour un utilisateur cible, c'est-à-dire lui fournir une liste des N items ayant les scores prédits (pour cet utilisateur) les plus élevés. Rappelons ici qu'un système de recommandation procède en deux phases consécutives : la *prédiction* et la *recommandation*. L'évaluation d'un système de recommandation peut donc porter sur l'une et/ou l'autre de ces phases et deux scénarios sont envisagés : l'évaluation basée sur les prédictions et l'évaluation basée sur la liste des top-N items recommandés [Herlocker et al., 2004; Cremonesi et al., 2010; Steck, 2013; Gunawardana and Shani, 2015].

Notons ici que, sauf indication claire, l'évaluation des SdR que nous discutons dans cette section relève du paradigme d'une évaluation *hors-ligne*, et qu'il s'agit donc de comparer les SdR en utilisant des jeux de données *benchmark* et des métriques d'évaluation standardisées dédiées à cette comparaison. Un exemple répandu de jeux de données *benchmark* est celui des *ratings* du prix Netflix [Bennett et al., 2007]. Il a été initialement publié dans le contexte d'un concours en-ligne et a depuis été utilisé comme l'une des références standardisées pour évaluer de nombreux algorithmes de recommandation. Le principal avantage de l'évaluation hors-ligne est qu'elle ne nécessite pas d'accès à une large base d'utilisateurs. En outre, de multiples jeux de données provenant de divers domaines (*e.g.*, musique, film, dessin animé) peuvent être utilisés pour tester la généricité du système de recommandation. L'inconvénient des évaluations hors-ligne est qu'elles ne mesurent pas la propension réelle de l'utilisateur à réagir face au système de recommandation à l'avenir.

A savoir qu'il existe bien dans la littérature d'autres paradigmes d'évaluation tels que, par exemple, l'évaluation *en-ligne* et l'évaluation sous formes de cas d'études des utilisateurs. Comme nous le verrons, c'est d'ailleurs ce type d'évaluation qui a été utilisé pour notre troisième contribution concernant l'explicabilité (cf. section 5.4 du chapitre 5) ; sans doute parce que c'est la plus sensible au ressenti personnel de chaque utilisateur. Pour plus de détails concernant ces paradigmes d'évaluation, le lecteur pourra se référer à [Gunawardana and Shani, 2015], mais ici nous resterons concentrés sur l'évaluation hors-ligne.

Comme mentionné précédemment, l'évaluation hors-ligne peut être effectuée en se basant sur : i) les prédictions des *ratings*, ou bien ii) les listes des top-N recommandations. L'évaluation basée sur ii) représente sans doute une perspective plus réaliste de la véritable utilité des systèmes de recommandation, puisqu'en général les utilisateurs ne *consomment* que les top-N items et non tous les items candidats. Toutefois, grâce à sa

simplicité, l'évaluation basée sur i) est généralement préférée dans un contexte qui utilise le *benchmarking*. C'est le cas, par exemple, de l'évaluation du prix de Netflix qui se base sur les prédictions des *ratings*. Dans les deux sections qui suivent, nous discutons ces deux familles d'évaluation des SdR et présentons des métriques utilisées dans chaque contexte d'évaluation.

2.1.6.1 Évaluation basée sur la prédiction des *ratings*

Dans ce cas, on considère que la comparaison de la pertinence (*accuracy*) des SdR est effectuée en fonction de leur capacité à prédire correctement (précisément) les valeurs des *ratings* des utilisateurs. Comme les *ratings* sont généralement représentés par des valeurs numériques, les métriques d'évaluation permettant de mesurer les erreurs de prédiction dans le cas des modèles de régression sont éligibles.

Notons S l'ensemble des *ratings* observés et $S_{test} \subset S$ un sous-ensemble des *ratings* utilisé pour tester un algorithme de recommandation. Chaque paire d'index $(u,i) \in S_{test}$ correspond à une position dans la matrice des *ratings*, *i.e.*, R. La valeur de la paire $(u,i) \in S_{test}$ est connue, *i.e.*, $r_{u,i} = R[u,i]$. Par ailleurs, $\hat{r}_{u,i}$ représente la valeur prédite par un algorithme de recommandation entraîné à partir des *ratings* dans $S \backslash S_{test}$, *i.e.*, le jeu d'entraînement. L'erreur de prédiction spécifique à la paire d'index (u,i) est donc représentée par l'écart entre $\hat{r}_{u,i}$ et $r_{u,i}$. Cet écart sur une paire (u,i) peut être mesuré de différentes manières et les écarts des différentes paires de S_{test} peuvent être combinés de différentes manières pour calculer l'erreur globale du modèle testé.

La métrique *Root Mean Squared Error* (RMSE) (2.30) est fréquemment utilisée pour évaluer l'erreur de prédiction globale, et elle est définie comme suit :

$$\text{RMSE} = \sqrt{\frac{\sum_{(u,i) \in S_{test}} (\hat{r}_{u,i} - r_{u,i})^2}{|S_{test}|}} \tag{2.30}$$

Notons que des petites valeurs de RMSE indiquent de meilleurs performances de prédiction. Le RMSE a été utilisé comme la métrique standardisée pour la compétition du prix de Netflix. Une caractéristique de RMSE est que cette métrique tend à pénaliser de manière disproportionnée les erreurs importantes en raison du terme au carré dans la somme [Aggarwal, 2016d]. Une autre métrique, nommée *Mean Absolute Error* (MAE), ne souffre pas de ce défaut :

$$\text{MAE} = \frac{\sum_{(u,i) \in S_{test}} |\hat{r}_{u,i} - r_{u,i}|}{|S_{test}|} \tag{2.31}$$

D'un point de vue global, le RMSE pourrait être préféré au MAE lorsqu'il s'agit d'évaluer l'*accuracy* d'un SdR en termes de prédiction des *ratings*, parce qu'il est significativement affecté par les erreurs importantes de prédiction (terme au carré) et un SdR serait donc considérablement pénalisé par quelques *ratings* mal prédits. Par contre, le principal

problème de RMSE est qu'il ne reflète pas fidèlement l'erreur moyenne, et cela peut parfois conduire à des résultats trompeurs [Willmott and Matsuura, 2005]. Des discussions plus approfondies concernant la comparaison entre MAE et RMSE sont proposées dans [Chai and Draxler, 2014].

2.1.6.2 Évaluation basée sur la liste de top-N recommandations

Contrairement à l'évaluation basée sur la prédiction des *ratings* dans laquelle toutes les paires $(u, i) \in S_{test}$ sont considérées, l'évaluation discutée dans cette section ne considère que les items présents dans la liste d'items recommandés afin de mesurer l'utilité de la liste entière. Formellement, notons $S_{test}(u)$ l'ensemble des items pertinents dans le jeu de test pour l'utilisateur u. Les items pertinents pour un utilisateur peuvent être les items notés par l'utilisateur (*feedback* implicite) ou des items dont les valeurs de *ratings* dépassent un seuil (*feedback* explicite). Soit $L_{rec}(u)$, la liste d'items recommandés à l'utilisateur u, avec $|L_{rec}(u)| = N$, le nombre d'items recommandés. Pour mesurer l'*accuracy* d'un système de recommandation en termes d'évaluation basée sur $L_{rec}(u)$, des métriques standardisées ont été proposées dans la littérature telles que : la *Précision*, le *Rappel*, la *F1-mesure*, la *Mean Average Precision* (MAP), le *Normalised Discounted Cumulative Gain* (NDCG), et le *Hit Rate* (HR).

Étant donnée l'utilisateur $u \in U$, la *Précision* de la liste recommandée pour u est définie par l'équation (2.32). Il s'agit de la proportion des items pertinents pour l'utilisateur u parmi tous les items recommandés pour l'utilisateur. Le *Rappel* (2.33), quant à lui, correspond au pourcentage des items pertinents, *i.e.*, $S_{test}(u)$, qui sont recommandés.

$$Précision(L_{rec}(u)) = \frac{|L_{rec}(u) \cap S_{test}(u)|}{|L_{rec}(u)|} \tag{2.32}$$

$$Rappel(L_{rec}(u)) = \frac{|L_{rec}(u) \cap S_{test}(u)|}{|S_{test}(u)|} \tag{2.33}$$

La *F1-mesure* est une métrique qui permet de combiner la *Précision* et le *Rappel*, en prenant la moyenne harmonique de ces deux mesures :

$$F1\text{-}Mesure(L_{rec}(u)) = \frac{2 \cdot Précision(L_{rec}(u)) \cdot Rappel(L_{rec}(u))}{Précision(L_{rec}(u)) + Rappel(L_{rec}(u))} \tag{2.34}$$

Les trois métriques d'évaluation présentées au-dessus sont largement utilisées dans le domaine de la fouille des données, la recherche d'information et l'apprentissage automatique pour évaluer la qualité de la classification [Powers, 2011]. Néanmoins, ces métriques ne tiennent pas compte de l'ordre des items dans la liste. Autrement dit, pour ces métriques, il n'y a pas de différence lorsqu'un item pertinent est présenté dans la tête ou dans la queue de la liste recommandée. Alors même qu'un SdR ordonne les items dans une liste de recommandations, il est souhaitable de considérer l'ordre dans lequel les items se présentent dans la liste. MAP et NDCG sont deux métriques qui prennent en

compte cette information. Ces deux métriques favorisent des listes de recommandations dans lesquelles les items les plus pertinents sont présentés en tête de liste.

Formellement, MAP est définie par l'équation (2.35) qui est basée sur la métrique *Average Precision* (AP) définie par l'équation (2.36). MAP représente la valeur moyenne de toutes les valeurs d'AP sur l'ensemble des utilisateurs U. Dans l'équation (2.36), le terme *Précision@k* représente la précision de la liste des k premiers items de $\mathrm{L}_{rec}(u)$.

$$\mathrm{MAP} = \frac{1}{|\mathrm{U}|} \sum_{u=1}^{|\mathrm{U}|} \mathrm{AP}(\mathrm{L}_{rec}(u)) \tag{2.35}$$

$$\mathrm{AP}(\mathrm{L}_{rec}(u)) = \frac{1}{|\mathrm{S}_{test}(u)|} \sum_{k=1}^{|\mathrm{L}_{rec}(u)|} (\textit{Précision@k} \text{ si le } k^{\text{ième}} \text{ item est pertinent, 0 sinon}) \tag{2.36}$$

Afin d'illustrer la métrique AP, considérons l'exemple suivant. Supposons un utilisateur u, qui a aimé 3 items i_1, i_2 et i_3 ($|\mathrm{S}_{test}(u)| = 3$). Supposons trois listes de recommandations suivantes : $l_1 = [i_4, i_5, i_1], l_2 = [i_4, i_1, i_2]$ et $l_3 = [i_1, i_2, i_4]$. D'après l'équation (2.36), les valeurs d'AP pour ces trois listes sont calculées comme :

$$\begin{aligned} \mathrm{AP}(l_1) &= \frac{1}{3} \times [0 + 0 + \frac{1}{3}] = 0.11 \\ \mathrm{AP}(l_2) &= \frac{1}{3} \times [0 + \frac{1}{2} + \frac{2}{3}] = 0.38 \\ \mathrm{AP}(l_3) &= \frac{1}{3} \times [\frac{1}{1} + \frac{2}{2} + 0] = 0.67 \end{aligned} \tag{2.37}$$

Normalised Discounted Cumulative Gain (NDCG) est une autre métrique d'évaluation de l'*accuracy* d'une liste de recommandations qui tient compte de l'ordre des items présents dans la liste. Elle est définie par l'équation (2.38) où $rel_{u,i}$ représente la pertinence (*relevance* en anglais) de l'item i pour l'utilisateur u. Généralement, $rel_{u,i}$ correspond à la valeur de *rating* $r_{u,i}$ dans le contexte des *feedbacks* explicites et $rel_{u,i} = 1$ ou $rel_{u,i} = 0$ dans le contexte des *feedbacks* implicites. Le terme index_i représente la position de l'item i dans la liste de recommandations et le terme IDCG est choisi de sorte que le NDCG de la liste idéale des recommandations, dans laquelle tous les items sont ordonnés en fonction de leur pertinences réelles, vaille 1 [Aggarwal, 2016d].

$$\mathrm{NDCG}(\mathrm{L}_{rec}(u)) = \frac{1}{\mathrm{IDCG}} \sum_{i \in \mathrm{L}_{rec}(u)} \frac{2^{rel_{u,i}} - 1}{\log_2(1 + \mathrm{index}_i)} \tag{2.38}$$

La dernière métrique que nous évoquons ici, *Hit Rate* (HR), est également largement utilisée par la communauté. Elle est généralement liée au protocole d'évaluation *leave-one-out* que nous discuterons dans la section suivante. Dans ce cas, $|\mathrm{S}_{test}(u)| = 1$, autrement dit nous considérons un seul item pertinent (l'item cible) pour l'utilisateur u et

toutes les autres données sont utilisées pour entraîner l'algorithme. On dit que l'on a un *hit* si, et seulement si, l'item cible est présent dans $L_{rec}(u)$. HR mesure la capacité de l'algorithme à pouvoir recommander l'item cible. Autrement dit, on mesure ici la proportion d'utilisateurs $u \in U$ pour qui le système a réussi à classer l'item cible dans leur liste de N meilleurs items, *i.e.*, $L_{rec}(u)$, parmi d'autres items candidats que ces utilisateurs n'ont pas noté [Aggarwal, 2016d]. Formellement, HR est défini par l'équation (2.39) en se basant sur l'ensemble des utilisateurs :

$$HR = \frac{\#hits}{|U|} \tag{2.39}$$

Comme en général chaque utilisateur ne note qu'une petite proportion des items du catalogue, le nombre d'items candidats pour l'utilisateur est très élevé. En pratique pour un utilisateur u, nous ne considérons donc qu'un sous-ensemble des items non-notés *e.g.*, 99 items et mesurons la capacité d'un SdR à classer l'item cible dans la tête des 100 items [Ferrari Dacrema et al., 2019].

2.1.6.3 Comparaison des deux types d'évaluation

Les deux familles d'évaluation présentées dans les sections précédentes permettent d'évaluer la performance d'un système de recommandation en termes d'*accuracy*. Par contre, comme mentionné plus haut, elles sont basées sur différentes phases d'un SdR (*prédiction* ou *recommandation*). Cette subtilité peut conduire au fait qu'un système de recommandation se verra attribuer des performances différentes selon le mode d'évaluation. Autrement dit, avoir de bonnes performances en termes de prédiction des *ratings* n'implique pas forcément qu'un système de recommandation fonctionne également bien en recommandation de la liste des top-N items et inversement. La différence fondamentale entre ces deux approches d'évaluation est liée à la prise en compte des items non-pertinents. En effet, pour la tâche des top-N recommandations, la différence entre les items *non-aimés* et ceux *détestés* (qui traduirait une gradation dans le *dégoût*) n'a pas d'importance, alors qu'elle est cruciale pour la tâche de prédiction des *ratings*. De plus, cette différence peut également être liée à la façon de procéder d'un système de recommandation. En général, les SdR dont l'apprentissage est basé sur les *feedbacks* explicites (*i.e.*, *ratings*) fonctionnent mieux en prédiction des *ratings* alors que les SdR dont l'objectif (ou l'apprentissage) est basé sur les *feedbacks* implicites peuvent être plus pertinents pour la tâche de recommandation des top-N items.

Des travaux ont été menés pour étudier cette différence et ainsi comparer les deux familles d'évaluation [Herlocker et al., 2004; Cremonesi et al., 2010; Steck, 2013]. Les auteurs de [Herlocker et al., 2004] ont comparé différents algorithmes de recommandation basés sur le filtrage collaboratif classique, c'est-à-dire ceux pour lesquels l'objectif est de prédire des *ratings*. Les résultats ont montré qu'il n'y a pas ou peu de corrélation entre ces deux types d'évaluation, autrement dit, un algorithme qui réussit à prédire correctement les *ratings* des utilisateurs n'est pas nécessairement aussi performant lorsqu'il s'agit de

sélectionner les top-N recommandations. Dans [Cremonesi et al., 2010], les auteurs ont comparé différents algorithmes de recommandation avec les métriques dédiées à l'évaluation d'une liste de top-N recommandations. Ils ont proposé des variantes des algorithmes de méthodes de CF classiques (*i.e.*, NBCF, SVD, SVD++) dont l'objectif n'est plus de minimiser l'erreur de prédiction (*e.g.*, RMSE) mais plutôt de favoriser un bon *ranking* des items. Les résultats ont montré que ces variantes sont plus performantes que les algorithmes classiques, qui sont optimisés pour une meilleure prédiction des *ratings*, en termes de *Précision* et de *Rappel*. [Steck, 2013] a proposé une étude comparative des deux familles d'évaluation d'*accuracy* des recommandations. Il a montré que la différence entre ces deux approches d'évaluation est fortement liée aux jeux de données d'entraînement et de test : la prédiction des *ratings* ne concerne que les *items* observés dans le jeu de test (S_{test}), alors que la tâche des top-N recommandations prend généralement en compte tous les items du catalogue, que ce soit des items notés ou non par les utilisateurs.

2.1.6.4 Protocoles d'évaluation

Dans cette section, nous discutons des protocoles d'évaluation classiquement utilisés dans le contexte de l'évaluation *hors-ligne*. Étant donné S, un ensemble de *ratings* contenant |S| entrées, chaque instance d'entrée correspond à une paire utilisateur-item, (u, i), avec $r_{u,i}$ la valeur de *rating* donnée par l'utilisateur u à l'item i.

Il est crucial que l'*accuracy* d'un système de recommandation ne soit pas sous-estimée ou sur-estimée selon le protocole d'évaluation utilisé. Par exemple, si on teste un algorithme sur le jeu de données utilisé lors de l'apprentissage, la performance du système évalué sera sur-estimée. Pour éviter cela, en général, le jeu de données initial S est divisé en deux sous-ensembles S_{test} et S_{train} avec $S_{test} \cup S_{train} = S$. S_{train} est alors utilisé pour entraîner les systèmes de recommandation candidats, tandis que leur performances sont évaluées en utilisant S_{test}.

Afin de diviser S en deux portions S_{test} et S_{train}, plusieurs méthodes ont été proposées :

- *Hold-out* — Il s'agit d'une approche simple et classique dans le domaine de l'apprentissage automatique, dans laquelle on segmente S selon un taux de segmentation $t \in [0,1]$. Autrement dit, t% des données sont utilisées pour tester l'algorithme, alors que les $(1-t)$% des données restantes sont utilisées pour l'entraînement de l'algorithme. Une valeur couramment utilisée pour ce taux de segmentation est $t = 0.2$. Cette méthode de segmentation est notamment très utilisée lorsque les informations temporelles des données sont disponibles.

- *Validation-croisée* — Dans la méthode de validation croisée, les entrées de S sont divisées en n ensembles de même taille. Par conséquent, la taille de chaque sous-ensemble est $\frac{|S|}{n}$. L'un des n segments est utilisé comme S_{test}, et les $n-1$ segments

restants sont utilisés comme S_{train}. En d'autres termes, un total de $\frac{|S|}{n}$ entrées sont cachées au cours de chaque processus d'entraînement, et la performance est ensuite évaluée sur ces entrées. Ce processus est répété n fois en utilisant successivement chacun des n segments comme S_{test}. La performance moyenne sur les n différents ensembles de S_{test} est finalement observée. L'avantage de cette approche est qu'elle permet d'estimer l'*accuracy* réelle d'un SdR quand le nombre de sous-ensembles n est grand. Par contre, cela demandera plus de temps de calcul. En pratique, une validation-croisée de 5 ou de 10 est souvent adoptée.

- *Leave-one-out* — Cette méthode est un cas particulier de la méthode précédente lorsque $n = |S|$. De ce fait, chaque S_{test} contient 1 seule instance et le S_{train} correspondant en contient $|S|-1$. L'avantage de cette façon de segmenter est que presque toutes les données sont utilisées pour l'entraînement. Notons que cette méthode est très lourde notamment lorsque $|S|$ est élevé puisque $|S|$ phases d'entraînement doivent être réalisés. En pratique, un alternatif est souvent utilisé [Hug, 2020]. Il consiste à ne garder qu'une seule instance par utilisateur. Autrement dit, pour chaque utilisateur $u \in U$ avec I_u étant l'ensemble des items notés par u, par la segmentation *leave-one-out*, S_{test} contiendrait $|U|$ instances et S_{train} en contiendrait $|S|-|U|$. Par exemple, pour un jeu de données d'un million de *ratings* avec 6 000 utilisateurs, 994 000 instances sont utilisées. De ce fait, on évite une sous-estimation de l'*accuracy* de l'algorithme. Notons ici que la segmentation *leave-one-out* est souvent utilisée avec la métrique d'évaluation *Hit Rate* (HR) présentée dans la section précédente.

Disposant de S_{test} et S_{train}, l'évaluation basée sur la prédiction vise à prédire pour chaque entrée (paire utilisateur-item) $(u,i) \in S_{test}$, la valeur de *rating* $r_{u,i}$. En revanche, il n'est pas intuitif de procéder de la même manière pour l'évaluation basée sur une liste d'items. Pour recommander une liste de N items à l'utilisateur u, on se base en général sur une liste d'items candidats $L_{candidats}(u)$ avec $|L_{candidats}(u)| > |L_{rec}(u)| = N$. La manière de construire $L_{candidats}(u)$ peut impacter la performance du système qu'on évalue. Différentes méthodes ont été proposées concernant la construction des $L_{candidats}(u)$ pour chaque utilisateur $u \in U$ [Bellogin et al., 2011; Steck, 2013] :

- *testRatings* — Dans ce cas, $L_{candidats}(u)$ ne contient que les items dans S_{test} qui sont notés par l'utilisateur u. Notons ici qu'il s'agit exactement du même scénario que lorsque l'on évalue un algorithme en se basant sur la prédiction.

- *testItems* — Dans ce scénario, $L_{candidats}(u)$ contient tous les items présents dans S_{test}, qu'ils soient notés ou non par l'utilisateur u. De ce fait, en comparant avec la méthode *testRatings*, cette méthode permet d'inclure certains items non-notés par l'utilisateur u, puisque d'autres utilisateurs les ont notés.

- *allUnratedItems* — Dans ce cas, $L_{candidats}(u)$ contient tous les items du catalogue qui ne sont pas notés par l'utilisateur u, *i.e.* l'entrée $r_{u,i}$ n'est pas observée dans S_{train}. De ce fait, l'ensemble des items candidats contient en général un nombre élevé d'items non-notés. Ce scénario est plus proche d'un cas d'application réel puisque l'objectif d'un SdR est de recommander des items que l'utilisateur n'a pas encore notés.

2.1.6.5 Autres objectifs d'une liste de recommandations et évaluations associées

Les métriques d'évaluations discutées précédemment, *i.e.* RMSE, MAE, Précision, Rappel, F1-mesure, MAP, NDCG et HR sont toutes utilisées pour évaluer l'*accuracy* d'un système de recommandation, qui est considéré comme l'objectif majeur d'un tel système. En outre, la communauté scientifique des SdR a proposé, en plus de l'objectif d'*accuracy*, d'autres objectifs complémentaires qui méritent également d'être pris en compte lors de l'évaluation des recommandations faite par un algorithme. Dans cette section, nous discutons quelques uns de ces objectifs populaires dans le domaine scientifique et industriel des systèmes de recommandation.

- *Nouveauté et sérendipité* — La *nouveauté* d'un système de recommandation représente sa capacité à pouvoir recommander des items dont l'utilisateur n'a pas connaissance ou qu'il n'a jamais vus auparavant. Notons ici qu'il existe une différence entre les items que l'utilisateur n'a pas notés et ceux qui sont nouveaux pour lui. Le fait qu'un item n'ait pas été noté par l'utilisateur ne signifie pas qu'il ne le connaît pas. Quant à la *sérendipité*, il s'agit d'un niveau supérieur de *nouveauté*, qui concerne des items qui peuvent surprendre l'utilisateur. Notons également que tous les items surprenants sont nouveaux pour l'utilisateur mais que l'inverse n'est pas forcément vrai. Tout dépend de leur *ressemblance* avec les items connus et aimés par l'utilisateur. Les notions de nouveauté et de sérendipité sont discutées plus avant dans [Murakami et al., 2007; Castells et al., 2015]

- *Diversité* — La *diversité* d'un système de recommandation représente sa capacité à pouvoir recommander des items diversifiés. Par exemple, imaginons le cas où trois films sont recommandés à un utilisateur dans la liste des top-3 items. Si les trois films sont exactement du même genre et impliquent les mêmes acteurs, alors les recommandations sont peu diversifiées. Une liste d'items peu diversifiée peut lasser les utilisateurs et ainsi réduire leur satisfaction [Ziegler et al., 2005]. La notion de diversité est fortement liée avec celle de *nouveauté* et de *sérendipité*. Le chapitre 4, comme nous le verrons par la suite, présente une étude systématique visant à comparer les performances de différentes familles de systèmes de recommandation en termes de diversification.

- *Explicabilité* — L'*explicabilité* d'un système de recommandation représente son aptitude à proposer aux utilisateurs les justifications pour lesquelles les items leur sont présentés. Une explication pertinente et argumentée permet aux utilisateurs de mieux comprendre l'intérêt de la recommandation pour eux et influence leur décision de considérer ou pas la recommandation [Tintarev and Masthoff, 2015]. Par exemple, imaginons que vous hésitiez à acheter un produit et que le système vous fournisse l'explication suivante : *99% d'utilisateurs ayant acheté le produit x ont également acheté ce produit.* Sachant que le produit x fait partie des produits que vous avez achetés, cette justification pourra influencer votre choix. Une étude devenue célèbre de l'Université d'Harvard a même montré que nous avons tendance à accorder de la valeur à une tentative d'explication même si elle n'est pas informative [Langer et al., 1978]. Dans le chapitre 5, nous discuterons plus en détails des aspects liés à l'explicabilité des recommandations et présenterons une approche d'explicabilité basée sur l'ingénierie des connaissances.

2.2 Apports de l'ingénierie des connaissances aux SdR

L'ingénierie des connaissances est une sous-branche de l'intelligence artificielle qui s'intéresse à la gestion des connaissances et à leur mise en œuvre dans différents contextes, notamment en recherche d'information, pour la conception de systèmes experts, ou l'aide à la décision, pour n'en citer que quelques uns. Plus précisément, l'ingénierie des connaissances englobe les processus de conception de modèles de connaissance, c'est-à-dire la représentation des concepts et de leur interconnexion (*e.g.*, ontologies, graphes de connaissances, bases de connaissances, etc.) et les différentes techniques qui permettent d'exploiter ou d'enrichir cette connaissance : extraction de connaissances à partir de textes, mesures de similarité, appariement de ressources, visualisation, etc. Elles disposent également de capacités de raisonnement permettant d'inférer de nouvelles connaissances à partir des faits existants et des règles logiques définies.

Bien qu'il existe des modèles de connaissances très génériques (*e.g.*, DBpedia), ils sont le plus souvent spécifiques à un champ thématique particulier, ce qui permet une modélisation plus fine. De nombreux domaines d'application ont bénéficié d'une intégration effective de l'ingénierie des connaissances dans leurs applications : la médecine, l'agronomie, la biologie, etc., notamment car l'intégration de connaissances du domaine et d'informations plus fiables permet aux spécialistes et experts du domaine d'affiner l'analyse de leurs données et de les manipuler plus efficacement au travers d'une meilleure représentation, et d'automatiser certains traitements/raisonnements.

La plupart des systèmes de recommandation sont spécialisés et visent à proposer des items liés à un domaine particulier : un film à regarder (domaine cinématographique), un roman à lire (domaine littéraire), un parcours de visite (domaine touristique), etc. Chaque domaine dispose de connaissances spécifiques qui leur sont propres. L'ingénie-

rie des connaissances constitue donc un réel atout pour améliorer les systèmes de recommandation en y intégrant une part de l'expertise du domaine. De nombreux travaux ont été proposés dans la littérature visant à améliorer la qualité des recommandations en tirant parti de l'ingénierie des connaissances. Par ailleurs, l'évolution méthodologique et technologique de l'ingénierie des connaissances influencent également la manière dont on l'exploite dans les systèmes de recommandation.

Les travaux plus anciens s'appuient notamment sur les ontologies du domaine en exploitant les relations hiérarchiques et sémantiques entre les concepts, et le calcul de similarité sémantique basée sur la structure ontologique [Middleton et al., 2001, 2004; Cantador et al., 2008; Sieg et al., 2010; El-Dosuky et al., 2012; Carrer-Neto et al., 2012; Harispe et al., 2013; Moreno et al., 2013; Rodríguez-García et al., 2015].

La standardisation des protocoles du Web sémantique (*e.g.*, triplets RDF, langages OWL et SPARQL, etc.) a facilité la représentation et la liaison des connaissances, et a permis de générer des bases et des graphes de connaissances couvrant simultanément de multiples domaines, *e.g.* DBpedia, *Knowledge Graph* de Google, etc. De plus, la réussite du couplage entre l'ingénierie des connaissances et l'apprentissage automatique a permis aux chercheurs de trouver des solutions plus robustes et flexibles, par exemple via l'utilisation de modèles de plongements (*embeddings*) de graphes de connaissances. Ainsi, des travaux plus récents concernant les apports de l'ingénierie des connaissances aux systèmes de recommandation s'appuient notamment sur les données liées (*linked data*) et les graphes de connaissances de grandes échelles [Di Noia et al., 2012b,a; Ostuni et al., 2013; Di Noia and Ostuni, 2015; Musto et al., 2016a; Palumbo et al., 2018b, 2020, 2018a].

Les sections suivantes présentent les apports de l'ingénierie des connaissances aux systèmes de recommandation.

2.2.1 Apports des ontologies

2.2.1.1 Ontologie

Dans le domaine de l'informatique et plus précisément de l'ingénierie des connaissances et de l'intelligence artificielle, une ontologie peut être définie comme une spécification formelle et explicite d'une conceptualisation partagée [Gruber, 1993; Studer et al., 1998]. *Conceptualisation* fait référence à un modèle abstrait d'un domaine donné qui identifie les concepts représentatifs de ce domaine. *Explicite* signifie que le type de concepts utilisés et les contraintes sur leurs utilisations doivent être explicitement définis, c'est-à-dire ne pas comporter d'ambiguïté. *Formelle* se réfère au fait qu'une ontologie doit être compréhensible par la machine, *i.e.*, cette dernière doit être capable d'interpréter la sémantique de l'information fournie. *Partagée* indique que l'ontologie représente une connaissance consensuelle, et qu'elle n'est pas restreinte à quelques individus mais acceptée par un large groupe [Broekstra et al., 2002]. Le modèle ainsi obtenu constitue un graphe de connaissances du domaine dont les nœuds représentent des concepts et les arêtes les

relations. On parle de *T-Box* ou *Terminological Box*.

On peut également *peupler* cette ontologie en associant aux concepts des instances réelles qui les illustrent. On rajoute pour ce faire des assertions qui traduisent des *faits* du monde réel conformes au modèle conceptuel défini dans la *T-Box*. On parle alors de *A-Box* ou d'*Assertion Component*. L'ensemble (*T-Box* et *A-Box*) constitue une *base de connaissances*.

La construction d'une base de connaissances débute donc par la définition de l'ontologie. Pour ce faire il est nécessaire de définir un ensemble de concepts représentatifs du domaine qu'elle représente. Par exemple, pour le domaine cinématographique, on emploie souvent des concepts comme *Film, Acteur, Réalisateur, Producteur, Genre, etc.* Ensuite, des relations doivent être établies entre ces concepts. On distingue deux types de relation : les relations *taxonomiques* et les relations *sémantiques*. Le premier type structure la hiérarchie de concepts de l'ontologie en établissant des liens de spécificité ou de généricité entre-eux, *e.g. Film d'action* est une spécification de *Film*. Le second type représente une relation sémantique entre deux concepts, *e.g.* la relation "*est réalisé par*" entre *Film* et *Réalisateur*. Ensuite, une phase de peuplement permet d'associer les *instances* à ces concepts, *e.g. Titanic, Matrix, Spider-man* peuvent être rattachés au concept *Film* (ce sont des *instances* de *Film*).

Pour rendre effective cette modélisation d'un domaine, *i.e.* implémenter l'ontologie, des langages de représentation de connaissances tels que RDFS et OWL sont utilisés. Basés sur les principes de la logique de premier ordre, ils possèdent un mécanisme d'inférence, ou de raisonnement, permettant de déduire des connaissances supplémentaires entre des éléments (*e.g.* instances, concepts) définis dans l'ontologie. De ce fait, l'ontologie permet d'inférer de nouvelles connaissances, ce qui est utilisé dans de nombreuses applications liées, entre autres, à l'aide à la décision.

Un modèle sémantique et ontologique peut être exploité à chaque phase du processus de recommandation : depuis l'inscription de l'utilisateur, la création de son profil initial, la prédiction de ses goûts par rapport aux items, l'ajustement de son profil en fonction des *feedbacks* et jusqu'à finalement la recommandation à proprement parler [Du et al., 2019b].

2.2.1.2 Annotation conceptuelle pour représenter les utilisateurs ou les items

L'apprentissage des profils utilisateurs demeure une étape cruciale pour les SdR basés sur le contenu (CBF) ou les SdR hybrides. La façon dont ils sont construits est donc déterminante concernant les performances du système. Dans une approche classique de CBF, le profil d'un utilisateur (resp. d'un item) est représenté par un vecteur dont la taille est égale au nombre de caractéristiques considérées et dont la valeur à l'indice i reflète l'importance de cette caractéristique pour cet utilisateur (resp. cet item). Pour créer ces profils, on s'appuie souvent sur des techniques provenant du *Traitement Automatique du*

Langage Naturel (TALN), *e.g.*, le pré-traitement du corpus, l'extraction des termes représentatifs, la pondération des termes en fonction de leur fréquence relative (*e.g.*, TF-IDF). De ce fait, la méthode CBF est naturellement adaptée pour recommander des objets textuels (*e.g.* livres, articles, pages web). Cette manière de construire les profils, qui se base principalement sur des opérations d'alignement entre chaînes de caractères pour détecter des mots clés, possède un inconvénient majeur : elle ne capture pas la sémantique du profil [Lops et al., 2011]. Pour y remédier, les ontologies ont été utilisées dans certains systèmes de recommandation pour servir de support à la représentation des profils utilisateurs (resp. des profils des items). Deux types d'approches sont proposés : l'un consiste à considérer les items comme des instances [Carrer-Neto et al., 2012; Rodríguez-García et al., 2015], l'autre associe un ensemble de concepts/instances de la base de connaissances aux éléments dont on veut définir le profil (items et/ou utilisateurs) [Middleton et al., 2001; Sieg et al., 2010; Moreno et al., 2013].

2.2.1.2.1 Représenter des items à l'aide des instances de concepts Comme présenté plus haut, la *A-Box* comporte l'ensemble des instances des concepts définis dans une ontologie du domaine ciblé. Il s'agit de l'ensemble des faits ou des objets rattachés à ces concepts lors de la phase de peuplement. Ces derniers sont donc catégorisés selon la structure de l'ontologie. Pour un SdR qui vise à recommander des items appartenant à un domaine particulier, il est donc possible de rattacher ces items aux éléments de cette *A-Box*. Deux choix sont possibles : i) soit la mise en correspondance est directe et les items sont eux-même assimilés à des instances, ii) soit la mise en correspondance est indirecte et les items sont alors stockés au sein d'un container (*e.g.*, une base de données) et possèdent des références les liant aux instances. La différence est que dans le premier cas, le SdR est totalement basé sur l'ontologie du domaine dont les instances des concepts forment le catalogue d'items du système [Carrer-Neto et al., 2012], alors que dans le deuxième cas, le catalogue d'items du SdR est indépendant de l'ontologie et on cherche à aligner (*mapper*) les items du SdR avec les instances de l'ontologie concernée. La plupart des approches de la littérature visent à générer des *mappings* entre les items et les objets au sein d'une ontologie [Di Noia et al., 2012b; Ostuni et al., 2013; Palumbo et al., 2020].

Dans les deux cas on peut tirer parti du modèle de connaissances et des relations définies entre concepts/instances. Dans le domaine cinématographique, par exemple, « *Bienvenue chez les Ch'tis* » est une instance du concept *Film* possédant des attributs (*i.e.*, des *propriétés d'objets* dans le langage du Web sémantique) qui le caractérisent. Pour ce film, nous aurons par exemple les propriétés suivantes : *Genre* (associé avec *Comédie*), *Acteur* (associé avec *Dany Boon*, *Kad Merad*, ...), *Réalisateur* (associé avec *Dany Boon*), *Pays d'origine* (associé avec *France*), *etc.*

Le fait de considérer les items au sein d'un SdR comme des instances d'une ontologie du domaine liée à la recommandation possède des avantages significatifs. Une clas-

sification des items est systématiquement établie grâce à la structuration des concepts auxquels ils se rattachent. Contrairement à un SdR ne disposant d'aucune information sémantique sur les items, un tel système a l'avantage de disposer de connaissances *a priori* sur les items. Ainsi, lors du peuplement d'une ontologie, on associe les instances aux concepts pertinents qui sont les plus bas dans l'arborescence, *i.e.* les concepts les plus spécifiques. Les liens qui peuvent exister entre une instance et des concepts plus génériques, ancêtres de ceux qui lui sont directement associés, ne sont pas explicités car ils peuvent être inférés par raisonnement. Outre certains raisonnements logiques qui peuvent être appliqués grâce aux relations sémantiques du modèle (*i.e.*, inférences), il est également possible d'y appliquer certaines mesures sémantiques et ainsi de considérer que deux items sont proches (resp. éloignés) en fonction des résultats de ces mesures. Ces calculs peuvent être d'une grande finesse si l'ontologie du domaine sur laquelle repose le système est définie de façon détaillée. Ainsi, selon les préférences exprimées par un utilisateur pour un item particulier (ou un ensemble d'items), des items considérés comme étant proches à l'aide de ces mesures peuvent constituer des recommandations judicieuses. De ce fait, on réduit le problème du démarrage à froid pour des nouveaux items dans le contexte d'une approche hybride. En effet, lorsqu'un item n'est encore noté par aucun utilisateur et qu'on ne connaît donc pas le vecteur de *ratings* qui lui est associé, il est tout de même possible de le recommander.

2.2.1.2.2 Modéliser le profil utilisateur par des éléments d'une ontologie Comme nous l'avons vu, dans une base de connaissances les instances sont caractérisées par les concepts de l'ontologie auxquels elles sont rattachées et par les attributs liés à ces concepts. Un profil utilisateur vise à caractériser les préférences de cet utilisateur au vu des items sur lesquels il s'est déjà exprimé, soit par des préférences explicitées à partir de *ratings*, soit par des interactions avec le système. Si les caractéristiques de ces items sont issues de l'ontologie du domaine de recommandation, le profil utilisateur pourra prendre la forme d'un vecteur dont chaque indice se réfère à un concept (*i.e.* une classe ou un attribut) et dont la valeur représente le taux d'intérêt de l'utilisateur pour ce concept. En d'autres termes, nous pouvons considérer que le profil utilisateur est représenté par les éléments présents dans la *T-Box* associés à différents poids selon les intérêts exprimés. Dans la littérature, cette approche est appelée *profil utilisateur ontologique*. Les auteurs de [Sieg et al., 2010] définissent un profil utilisateur comme un ensemble de nœuds dont chacun est représenté sous la forme d'une paire, $\langle C_j, IS(C_j)\rangle$, où C_j est un concept défini dans l'ontologie et $IS(C_j)$, *i.e.*, *Interest Score*, est le taux d'intérêt d'un utilisateur pour le concept C_j. Ils utilisent ensuite ces profils pour estimer la similarité entre utilisateurs dans une approche hybride. Ce formalisme se base sur le travail de [Middleton et al., 2004] qui adopte des triplets $\langle$*utilisateur*, *thème*, *taux d'intérêt*$\rangle$ pour modéliser l'intérêt des utilisateurs pour des articles scientifiques de différents domaines.

Les auteurs de [Carrer-Neto et al., 2012] proposent un autre moyen de représenter

le profil utilisateur, nommé *Recon*, qui repose sur la *A-Box*. Comme précédemment, ce profil est composé d'un vecteur associé à chaque utilisateur, mais dont chaque dimension traduit un degré d'intérêt pour une instance (un item particulier) et non pas un concept. L'avantage d'encapsuler les instances dans le profil utilisateur est que chaque nouvelle évaluation affine la définition du profil. Ainsi une part du filtrage est déjà réalisée puisqu'il suffira d'ordonner les composantes du vecteur pour déterminer les items (*instances*) qui ont le plus de poids et en faire la recommandation.

Contrairement aux approches qui adoptent un vecteur (de concepts ou d'instances) pour modéliser le profil utilisateur, les auteurs de [Blanco-Fernandez et al., 2008] présentent un système CBF pour recommander des programmes de télévision numérique. La spécificité de leurs travaux est que le profil utilisateur est modélisé par un sous-graphe extrait à partir d'une base de connaissances associée à un domaine, *i.e.* celui de la *TV* dans leur cas. Un tel profil contient des instances (des programmes) pertinentes, leurs principaux attributs et les genres sous lesquels ces programmes sont classés dans l'ontologie. L'intégration des différents types de nœuds dans le profil permettra un filtrage plus fin puisque l'on dispose de plus d'informations.

2.2.1.2.3 Inférer le taux d'intérêt et mettre à jour le profil utilisateur L'initialisation des différentes valeurs du vecteur correspondant au degré d'intérêt des utilisateurs est importante puisqu'elle permet de proposer des items dès la première connexion au système. Pour ce faire, en amont de la recommandation, un formulaire ou un questionnaire est souvent soumis à l'utilisateur lors de son inscription. Un exemple classique est celui des applications de réseaux sociaux, *e.g. Twitter, TikTok*, qui, à l'arrivée d'un utilisateur, lui demandent de choisir les *Tags* (étiquettes) qui l'intéressent. Si ces étiquettes ne représentent généralement que des mots-clés indépendants, en les associant à des ontologies on bénéficie d'avantages certains fournis par la sémantique sous-jacente entre les termes. Au lieu de questionner les utilisateurs sur leurs préférences par rapport à chaque concept de l'ontologie pour engendrer un profil initial complet, les auteurs de [Moreno et al., 2013] proposent dans leur système de recommandation d'activités touristiques, de ne questionner l'utilisateur que sur quelques concepts généraux mais suffisamment significatifs pour représenter les principaux centres d'intérêt des touristes (*e.g. Plage, Shopping, Culture, Gastronomie*, etc.). L'avantage de demander directement à l'utilisateur d'exprimer ses goûts pour construire son profil est que cela permet d'obtenir des recommandations pertinentes, *e.g.*, s'il a exprimé qu'il aimait la plage, on lui recommandera des activités proches de la mer ou contenant des activités nautiques. Cependant, cela demande un effort à l'utilisateur, et cette activité, en plus d'être chronophage, peut également être perçue comme trop intrusive. Paradoxalement, dans la plupart des cas, les utilisateurs cherchent à recevoir des recommandations précises et pertinentes mais sans trop se dévoiler au système, sans être interrogés de façon précise. Dans ce cas, une solution consiste à attribuer initialement le même poids à chaque concept du profil, comme

c'est le cas dans [Sieg et al., 2010]. Ce n'est qu'au fur et à mesure de l'utilisation du SdR que le profil de chaque utilisateur sera affiné et que les recommandations gagneront en pertinence.

Pour améliorer la qualité de la recommandation, le profil utilisateur doit être mis à jour à chaque *feedback* fait au système. Ces preuves d'intérêt fournies par l'utilisateur peuvent exprimer ses préférences d'une façon explicite (*e.g.* un *rating*) ou implicite, par exemple un clic, une sauvegarde, un achat, une écoute, etc. Différents types de retour possèdent des significations différentes. Un retour explicite est souvent considéré comme plus fiable qu'un retour implicite, car ce dernier est soumis à l'interprétation automatique de la machine et est donc moins fiable. Par exemple écouter une musique qui nous a été recommandée ne signifie pas forcément qu'on l'a aimée, on peut l'avoir fait par simple curiosité. Afin d'ajuster le profil, on ajoutera ou enlèvera des points à certains attributs du profil pour qu'il reflète au mieux l'ensemble des interactions de l'utilisateur avec le système. Parmi les retours d'information explicites, nous pouvons distinguer l'importance relative des actions en fonction des concepts sur lesquels portent ces retours. Par exemple dans les travaux de [Carrer-Neto et al., 2012], un utilisateur a la possibilité de noter non seulement des films, *i.e.* des instances de la classe de référence *Film*, mais aussi d'autres types d'éléments (*e.g.*, des acteurs, des réalisateurs, etc.) qui peuvent être associés à ces films. Les valeurs du vecteur proposé (*Recon*, cf. section 2.2.1.2.2) sont ajustées de manière différente selon le type d'instances sur lequel un utilisateur a fourni son retour. Ainsi, on ajuste le taux d'intérêt pour un film en attribuant un poids plus faible s'il s'agit d'un retour sur ses acteurs que s'il s'agit d'un retour sur le film lui-même.

Dans une ontologie, on dispose des relations hiérarchiques et sémantiques entre les concepts. Elles permettent de relier les concepts et de leur conférer du sens. De ce fait, une propagation des préférences au travers des relations pourra être prise en compte lors de l'ajustement du profil, *e.g.* si on aime des romans de suspense alors on pourra éventuellement aimer ceux de mystère vu que la catégorie suspense est une sous-catégorie de mystère. Pour ce faire, l'algorithme de *Spreading Activation* permet de propager des préférences entre les nœuds via leur relations hiérarchiques [Middleton et al., 2001; Sieg et al., 2007; Moreno et al., 2013].

2.2.1.3 Mesures de similarité sémantique entre items et/ou entre utilisateurs

« *Si vous aimez l'objet A, vous aimerez l'objet B* », voici un exemple typique de ce que l'on peut lire sur un site de commerce électronique. Derrière cette suggestion se cache un SdR qui nous recommande l'objet B parce qu'il a jugé que son contenu est *similaire* à celui de l'objet A que nous venons d'acheter ou de consulter, *i.e.* un SdR de type CBF. Cette proximité entre items repose sur une mesure de distance entre leurs profils, qui se résume par conséquent, la plupart du temps, à une distance entre leurs vecteurs de caractéristiques (*e.g.* l'angle, la corrélation, etc.). Deux problèmes se posent : i) l'extraction et la sélec-

tion des propriétés représentatives est une tâche complexe et souvent les caractéristiques sélectionnées n'arrivent pas à couvrir la totalité des descriptions des items ; ii) il existe souvent des propriétés redondantes parmi les dimensions du vecteur qui peuvent donc biaiser les résultats. Par exemple, dans le domaine cinématographique, les propriétés *suspense* et *mystère* traduisent une granularité différente et sont, par conséquent, en partie redondantes.

L'autre moyen d'apprécier la proximité entre items, adopté par des approches de filtrage collaboratif, consiste à mesurer la distance entre des vecteurs de *ratings* présents dans la matrice *utilisateur-item*. La limite de cette mesure vient du fait que la matrice est généralement creuse et qu'on n'a qu'une vue partielle de la similarité entre items si l'on exploite uniquement leurs notes (deux films peuvent avoir les mêmes notes pour des raisons très différentes).

Comme nous l'avons vu, les items d'un système de recommandation peuvent être représentés par des *instances de concepts* définis dans l'ontologie du domaine ciblé par la recommandation. La mesure de similarité entre ces ressources au travers de leur caractérisation sémantique pourra être adoptée. Cette mesure sémantique s'applique soit d'une façon indépendante [Harispe et al., 2013], soit en combinant d'autres mesures [Al-Hassan et al., 2011; Carrer-Neto et al., 2012; Benouaret, 2017].

L'atout majeur de l'estimation des proximités entre items grâce à une mesure sémantique entre instances des concepts d'une ontologie est que, d'une part, on lève le problème de la sélection des propriétés représentatives des items dans le cas d'un SdR basé sur le contenu. En effet, étant donné que les propriétés de ces items sont contenues et structurées directement dans le modèle, elles peuvent intervenir dans le calcul sans filtrage préalable. De plus, les relations qui les connectent fournissent une sémantique riche qui permet d'interpréter la recommandation produite. D'autre part, l'utilisation de l'ontologie de domaine pour estimer ces similarités pallie le problème du *démarrage à froid* dans le cas du filtrage collaboratif basé sur les items, car cela permet de proposer une mesure de comparaison solide entre items, même si ceux-ci n'ont été notés que par peu ou pas d'utilisateurs.

Par exemple, pour apprécier la similarité entre deux instances de la classe *Film* dans l'approche proposée par [Carrer-Neto et al., 2012], le calcul se base sur la proportion du nombre d'instances partagées par deux films au vu de l'ensemble des propriétés non taxonomiques (propriétés d'objet ou de données – *object properties* et *datatype properties* en OWL) que la classe de référence *Film* comporte. Formellement, la similarité entre deux instances a et b est mesurée par l'équation (2.40) où P représente l'ensemble des #P propriétés de la classe de référence, $common(a,b,\mathrm{P}[i])$ représente le nombre d'instances partagées par a et b via la propriété $\mathrm{P}[i] \in \mathrm{P}$, $deg(a,\mathrm{P}[i])$, le nombre d'instances associées à l'individu via la propriété $\mathrm{P}[i]$ et $Weight(\mathrm{P}[i])$, quant à lui, est une valeur entre 0 et 1 qui indique le poids associé à la propriété, *e.g.* le genre d'un film est certainement plus important que l'endroit où le film a été réalisé. L'avantage de cette approche est sa

simplicité de calcul. Son inconvénient est qu'elle ne considère que le nombre d'instances partagées par deux items mais pas les contenus (types, valeurs, etc.) de ces instances ; or, deux genres de film pourraient être proches même s'il s'agit de deux instances différentes. Les auteurs de [Rodríguez-García et al., 2015] étendent cette mesure de similarité par la prise en compte des valeurs des propriétés du type *datatype properties*, *e.g.* s'il s'agit de chaînes de caractères alors la distance de *Levenshtein* s'appliquera.

$$\mathrm{S}(a,b) = \sum_{i=1}^{\#\mathrm{P}} \left(\frac{common(a,b,\mathrm{P}[i])}{max(deg(a,\mathrm{P}[i]),deg(b,\mathrm{P}[i]))}\right) \cdot \mathrm{W}eight(\mathrm{P}[i]) \tag{2.40}$$

Les auteurs de [Harispe et al., 2013] proposent une mesure de similarité sémantique dédiée à l'estimation des proximités des instances présentes dans une base de connaissances RDF. La notion de projection des instances est introduite dans cette approche pour caractériser leurs propriétés. Une projection représente un ensemble de chemins dont les points de départ sont les instances elles-même, et les points d'arrivée définissent le type de propriété considéré. Parmi ces types on distingue : les *données*, si le point d'arrivée est de type littéral (*e.g.* valeur numérique, chaîne de caractères, etc.) ; les *instances*, si le point d'arrivée est un ensemble d'instances ; les *concepts*, s'il s'agit d'un ensemble de concepts (classes en OWL) ; enfin le type *complexe*, si plusieurs chemins sont nécessaires pour caractériser une propriété. Ainsi, pour mesurer la proximité entre deux instances, on associe à chaque type de projection, une mesure spécifique. Par exemple, pour une paire de projections de type *conceptuel*, l'approche introduite par [Pesquita et al., 2009] pourra être adoptée. Dans le cas *complexe*, une agrégation de différentes mesures peut être appliquée.

Dans des approches hybrides, différents types de mesures de similarité peuvent être activés ou désactivés selon le contexte de la recommandation. [Benouaret, 2017] propose un système de recommandation qui vise à suggérer à ses utilisateurs des œuvres à visiter en s'adaptant à leurs préférences. Trois composantes, *i.e.* i) démographique, ii) sémantique et iii) collaborative s'activent consécutivement en fonction du contexte de la recommandation. Plus précisément, étant donné un utilisateur, ses informations démographiques (i), *e.g.* l'âge, le sexe, le pays, sont utilisées dans le cas où il s'agit de sa première authentification pour trouver l'ensemble des utilisateurs qui possèdent un profil démographique similaire afin de procéder à la recommandation. Ensuite, lorsqu'il a exprimé ses préférences par rapport aux items recommandés par la première approche, l'approche sémantique (ii) s'active afin de lui proposer des items qui sont sémantiquement proches de ce qu'il a aimé. Enfin, après avoir collecté suffisamment de notes de la part de l'utilisateur, le filtrage collaboratif (iii), basé sur la similarité des notes, s'active afin de prendre en compte ce qu'ont aimé ses voisins. Notons que pour la composante sémantique (ii), différents calculs s'appliquent pour estimer la proximité entre deux œuvres en fonction des valeurs que prennent leurs propriétés définies dans la base de connaissances. On notera ici l'importance de la qualité des ontologies utilisées. En effet, leur ni-

veau de détail et leur degré de couverture du domaine impactent fortement l'estimation de mesures sémantiques.

2.2.2 Apports des graphes de connaissances

Un graphe de connaissances (KG pour *Knowledge Graph*) est une structure de données basée sur le standard des triplets RDF, permettant de représenter des faits liées à un domaine spécifique, *e.g.* $\langle$*L. DiCaprio, est_acteur_de, Titanic*$\rangle$. Dans un tel graphe, on dispose des nœuds et des arêtes qui les relient. D'un point de vue terminologique, les nœuds se réfèrent aux *entités* et les arêtes se réfèrent aux *relations*. Un KG constitue donc un ensemble des triplets $\langle s, p, o\rangle$ où $s, o \in$ E représentent des entités et $p \in$ P représente des prédicats (relations) qui relient l'entité *sujet* s et l'entité *objet* o. Ainsi, un graphe de connaissances, souvent de portée plus large et regroupant différents champs thématiques, peut néanmoins être considéré comme une ontologie puisqu'il respecte les mêmes principes que ces dernières.

Dans une perspective différente de celle présentée dans la section précédente, nous allons discuter ici des apports des KG aux systèmes de recommandation, dans un contexte plutôt quantitatif orienté *données*. Pour cela, nous nous focaliserons principalement sur les contributions relatives aux données liées (*Linked Data*) et aux techniques récentes associées aux KG (*A-Box*) à grande échelle (*e.g.* DBpedia), comme par exemple la technique de plongement (*embeddings*) qui permet de représenter des entités dans un espace défini par le KG.

2.2.2.1 Modèles de recommandation basés sur les *Linked Data*

L'ambition du Web sémantique et des données liées (*Linked Data*), tels que définis à l'origine par Tim Berners Lee, est de générer un *Web de données* interopérable et *lisible* par les humains comme par les machines. Cette vision est étayée par : (1) une représentation commune des données lisible par une machine (langage RDF) ; (2) une sémantique commune rattachée à ces données (modélisée au travers de langages ontologiques tels que RDFS ou OWL) et (3) un moyen commun d'interrogation de ces données (via le langage SPARQL). Basée sur ce paradigme, une approche plus pragmatique a émergé, qui se concentre sur la mise à disposition de données structurées et interconnectées plutôt que sur une sémantique et un raisonnement formels de haut niveau. Le terme *Web de données* est ainsi défini, en se basant sur les principes des données liées présentés par [Berners-Lee, 2006]. Ces principes sont : (1) utiliser les URI (*Uniform Resource Identifier*) pour désigner les entités manipulées ; (2) utiliser le protocole HTTP et les URI pour accéder à ces entités ; (3) fournir des informations utiles à chaque URI ; (4) inclure des liens vers d'autres URI qui permettent de découvrir des informations complémentaires. Par exemple, considérons l'identifiant unique `http://dbpedia.org/resource/Titanic_(1997_film)`, (1) il s'agit de l'URI du film *Titanic* ; (2) il utilise le protocole HTTP pour qu'on puisse y accé-

der via un navigateur web ; (3) lorsque nous accédons à cet URI, nous disposons d'informations concernant le film *Titanic*, comme le nom de son réalisateur, une liste de ses acteurs principaux, etc. ; (4) des liens vers d'autres ressources sont également proposés via leur propre URI, *e.g.* http://dbpedia.org/resource/20th_Century_Studios.

Le projet Linked Open Data (LOD) a débuté en 2007 en tant qu'effort communautaire et a contribué à produire des milliards de triplets RDF qui sont maintenant publiés sur le Web. L'objectif du LOD est de relier entre elles des données disponibles gratuitement sur le Web, mais qui sont contenues dans des silos de données indépendants. Ces données sont ensuite reliées et publiées au format RDF, fournissant ce qui est appelé le nuage des données liées[5] (voir figure 2.9). Par exemple, les informations de DBpedia [Auer et al., 2007] (une exportation RDF de *Wikipedia*, au centre de ce nuage) sont liées aux informations géographiques de *Geonames*[6] ainsi qu'aux informations relatives à la musique de *MusicBrainz*[7].

En 2010, un système de recommandation d'artistes musicaux basé sur DBpedia appelé *dbrec* a été proposé [Passant, 2010a]. Il s'agit d'une approche basée sur le contenu. Lorsqu'un utilisateur consulte la page d'un artiste, *e.g. Johnny Cash*, le système lui propose des artistes proches de celui-ci, *e.g. Elvis Presley*. Pour calculer la distance entre deux ressources (dans ce cas, des artistes), *dbrec* s'appuie sur DBpedia. Plus précisément, pour calculer la distance entre deux entités e_a et e_b, la mesure proposée repose sur le nombre d'arêtes qui lient directement e_a et e_b et/ou celles qui les lient indirectement via une troisième entité. En se basant sur les distances obtenues entre ressources, Passant a ainsi défini une ontologie nommée LDSD (*Linked Data Semantic Distance*), contenant une classe *Distance* permettant de représenter la distance sémantique entre deux artistes quelconques.

Les auteurs de [Heitmann and Hayes, 2010] proposent de résoudre le problème du démarrage à froid lié au filtrage collaboratif en utilisant différentes sources disponibles au travers des données liées. L'idée est de rassembler les données provenant de différentes bases de connaissances (sources) qui sont structurées et reliées entre elles. Prenons le cas d'un éditeur de *Wikipedia* qui crée un nouveau compte sur un site de recommandation musicale. Le système ne dispose alors d'aucune information de base sur les préférences de ce nouvel utilisateur. Cependant, si nous pouvons trouver, via d'autres sources, des données le concernant relatives aux artistes musicaux qu'il apprécie, nous pouvons alors lui fournir instantanément des recommandations personnalisées. Toutefois, cette approche est assez atypique et peut être controversée parce qu'elle dépend de la disponibilité des données sur les utilisateurs et pose donc le problème de la confidentialité des données personnelles et de leur utilisation.

Les auteurs de [Di Noia et al., 2012b] proposent un système de recommandation de

[5] Dans la suite nous employons le terme LOD pour désigner ce nuage de données liées
[6] http://geonames.org
[7] https://fr.wikipedia.org/wiki/MusicBrainz

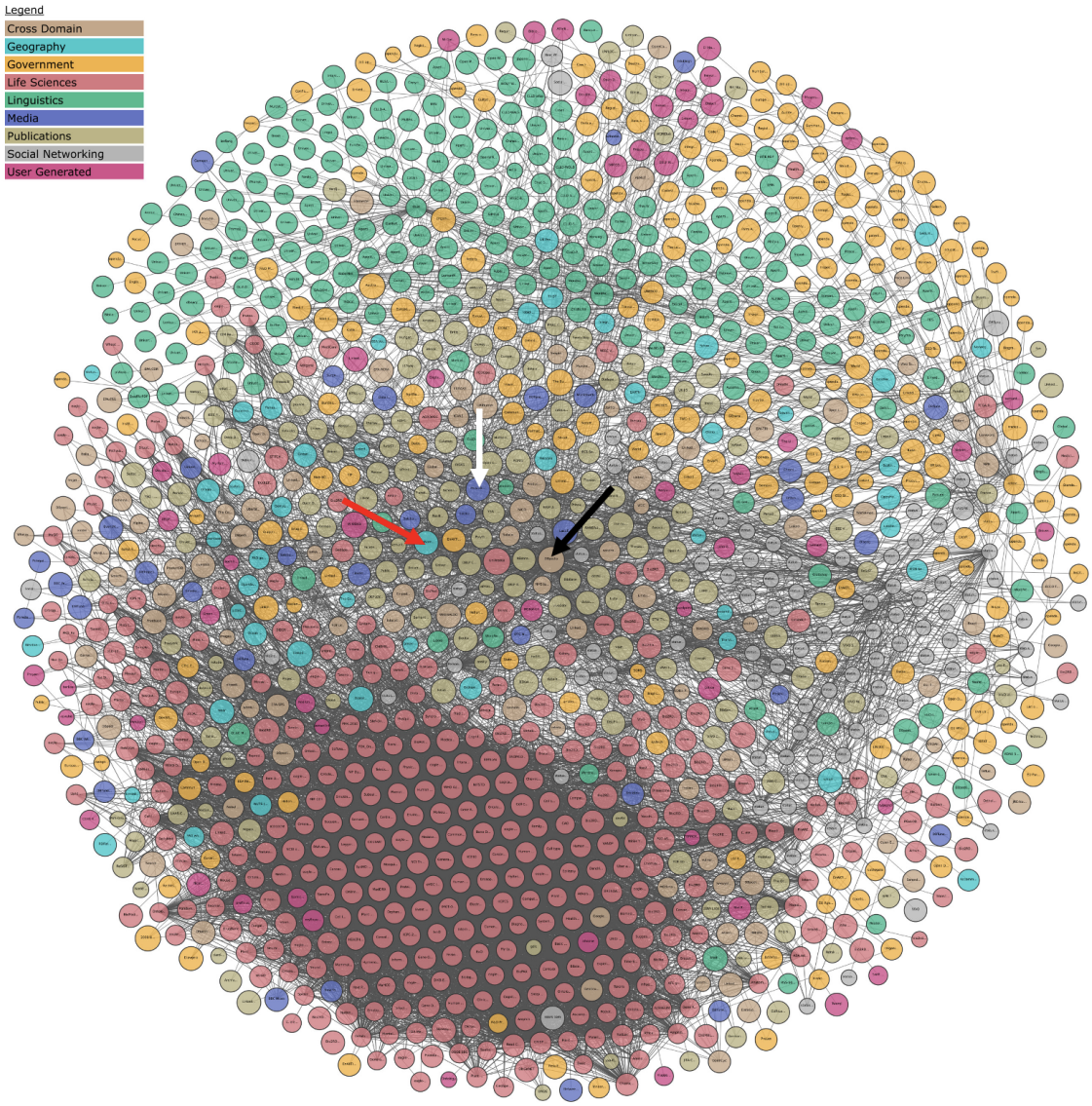


FIGURE 2.9 – Nuage des données liées (*Linked Open Data*) en 2020. Chaque bulle représente un *triplet store* (une base de connaissances spécifique) et les liens noirs représentent leur interconnexion. La flèche noire indique DBpedia (placé au centre car les connaissances qu'il contient sont communes à plusieurs domaines et sont considérées comme données de références pour de nombreuses autres bases), la flèche rouge indique Geonames et la flèche blanche Musicbrainz.

type CBF basé sur les données liées dans le domaine cinématographique. Dans leurs travaux, trois bases de connaissances (DBpedia, Freebase et LinkedMDB) ont été utilisées afin d'extraire des connaissances liées aux films. Le modèle de recommandation proposé repose également sur la distance sémantique entre les items (dans ce cas, les films). Ils ont proposé une mesure de similarité sémantique basée sur un modèle vectoriel permettant de représenter chaque film par des vecteurs selon différentes propriétés. Formellement, étant donné le film m_i et la propriété p, *e.g. dbo:director*, m_i est représenté par un vecteur de t dimensions $\overrightarrow{m_{i,p}} = (w_{1,i,p}, w_{2,i,p}, ..., w_{t,i,p})$. $w_{n,i,p}$ désigne le poids associé à chaque dimension et t représente le nombre d'objets dans le KG qui sont liés avec la propriété *dbo:director* (*i.e.* dans ce cas, t désigne le nombre des *réalisateurs* dans le KG). Les auteurs

ont ainsi proposé de mesurer la similarité cosinus entre deux films par rapport à une propriété donnée. Enfin, pour prédire le *rating* de l'utilisateur u pour le film m_i, $\hat{r}(u, m_i)$, les auteurs proposent l'équation suivante :

$$\hat{r}(u, m_i) = \frac{\sum\limits_{\langle m_j, v_j \rangle \in profil(u)} v_j \cdot \frac{\sum_{p \in \mathrm{P}} \alpha_p \cdot sim_p(m_i, m_j)}{|\mathrm{P}|}}{|profil(u)|} \tag{2.41}$$

où P représente l'ensemble des propriétés considérées, α_p représente le poids pour la propriété p en fonction du profil utilisateur et $sim_p(m_i, m_j)$ la similarité entre deux films m_i et m_j par rapport à la propriété p. Le terme $profil(u)$ dénote l'ensemble des items notés par l'utilisateur u :

$$profil(u) = \{\langle m_j, v_j \rangle \,|\, v_j = 1 \text{ si } u \text{ aime } m_j, v_j = -1, \text{sinon}\}$$

Dans une perspective un peu différente de celles présentées précédemment, qui s'appuient essentiellement sur le LOD pour calculer la distance sémantique des entités, les auteurs de [Di Noia et al., 2016] proposent une approche qui se focalise sur la structure de graphe. En fait, les éléments au sein d'un système de recommandation, tels que les utilisateurs, les items et les *ratings*, ainsi que les connaissances liées aux items accessibles au travers du LOD, peuvent constituer un graphe orienté, comme montré par la figure 2.10. Notons qu'en général, un tel graphe contient beaucoup plus de nœuds et de relations que présenté dans cet exemple illustratif.

Pour recommander des items à l'utilisateur ciblé, l'approche de [Di Noia et al., 2016] consiste à identifier un ensemble de chemins (*path*) avec une longueur inférieure à un seuil (6 dans leur travaux), qui débutent de cet utilisateur et se terminent par un item spécifique. Ils distinguent trois types de chemin selon les types de relations présentes dans le chemin : (1) *collaboratif, i.e.* les chemins de l'utilisateur à l'item qui passent par d'autres nœuds du type *utilisateur*; (2) *content-based, i.e.* les chemins de l'utilisateur à l'item qui passent par les entités représentant les connaissances liées aux items et (3) hybride, *i.e.* les chemins contenant les deux types de nœuds. Par exemple, un chemin collaboratif peut être {*aime, n'aime pas, aime*} à partir des faits : $u_1 \xrightarrow{\text{aime}} \textit{Titanic} \xrightarrow{\text{n'aime pas}^{-1}} u_2 \xrightarrow{\text{aime}} \textit{Star Wars : Episode I.}$ Ensuite, pour l'utilisateur focal, les auteurs proposent une méthode qui ordonne tous les chemins en utilisant des algorithmes d'apprentissage d'ordonnancement (*learning to rank*), qui visent à *apprendre* ou *estimer* une fonction de permutation à partir des données disponibles de préférences des utilisateurs. Enfin, les top-N chemins sont utilisés pour faire la recommandation.

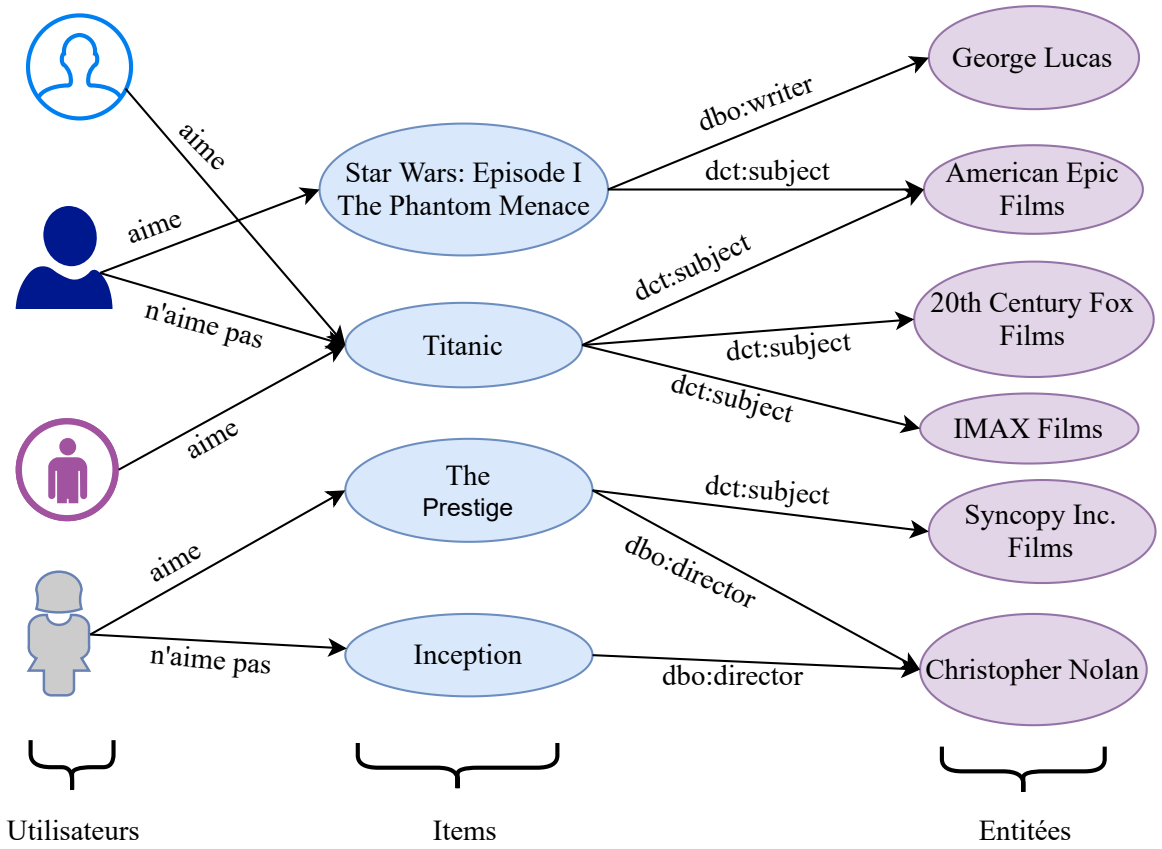


FIGURE 2.10 – Exemple d'un graphe orienté construit à partir des éléments au sein d'un système de recommandation

2.2.2.2 Modèles de recommandation basés sur les plongements de graphes, *Knowledge Graph Embedding* (KGE)

Bien qu'ils soient efficaces pour représenter des connaissances de manière structurée, la nature symbolique sous-jacente des triplets RDF rend généralement les graphes de connaissances (KG) difficiles à manipuler, notamment lorsque leur taille est très grande [Wang et al., 2017].

Afin de rendre la manipulation des KG de grand volume plus flexible, une nouvelle direction de recherche a été proposée, et rapidement adoptée par une large communauté de chercheurs : les modèles de plongement de graphes de connaissances – ou *Knowledge Graph Embedding* (KGE). L'idée principale de KGE est de *plonger* les éléments d'un graphe de connaissances, incluant les entités et les relations, dans des espaces vectoriels afin de simplifier la manipulation du KG tout en préservant l'essentiel de sa sémantique. Autrement dit, nous souhaitons que les éléments qui sont proches dans le graphe de connaissances se trouvent également proches dans l'espace vectoriel où ils sont plongés. En général, les modèles de KGE emploient les techniques relatives au domaine d'apprentissage automatique, et se basent sur des fonctions objectifs précises. Les représentations vectorielles des entités et des relations à l'issue de l'apprentissage par des modèles KGE peuvent être utilisées par la suite pour bénéficier à de nombreuses tâches, telles que la

complétion de KG [Socher et al., 2013], l'extraction de relations [Weston et al., 2013], la classification des entités [Nickel et al., 2012].

Dans cette section, nous allons discuter des travaux de recherche concernant la recommandation à l'aide de techniques de plongements de graphes de connaissances. Avant cela, nous introduisons des modèles de KGE de l'état de l'art.

2.2.2.2.1 Modèles des *embeddings* L'objectif d'un modèle de KGE est d'apprendre les représentations vectorielles pour que les éléments *proches* dans le graphe se trouvent également *proches* dans l'espace induit. Pour ce faire, il faut commencer par définir une mesure de proximité entre éléments d'un graphe de connaissances. Comme un KG est composé d'un ensemble de triplets $\langle s, p, o\rangle$ dont chacun possède une sémantique qui relie l'entité sujet s avec l'entité objet o par le prédicat p, il est primordial de considérer cette information sémantique. Par exemple, nous souhaitons que $\langle \text{Pékin}, \ capitale_de, \ \text{Chine}\rangle$ au lieu de $\langle \text{Pékin}, \ capitale_de, \ Japon\rangle$ même si *Chine* et *Japon* représentent deux entités relativement *proches*. Autrement dit, nous souhaitons que les vecteurs déduits pour l'entité *Pékin* et la relation *capitale_de* nous amènent plus près du vecteur de *Chine* que du vecteur de *Japon*.

Les auteurs de [Bordes et al., 2013] proposent *TransE*, un modèle de KGE représentatif de l'état de l'art. L'idée de base de cette approche est de considérer l'entité queue (o de $\langle s, p, o\rangle$) comme une traduction de l'entité tête s et la relation p, *i.e.* s et p traduisent o. De nombreuses extensions de *TransE* ont été proposées telles que *TransH* [Wang et al., 2014], *TransR* [Lin et al., 2015], *TransD* [Ji et al., 2015]. Formellement, étant donné un fait $\langle s, p, o\rangle$, l'objectif de *TransE* est d'apprendre pour chaque élément sa représentation vectorielle dans l'espace à d dimensions, *i.e.* $\mathbb{R}^d$ avec d un hyper-paramètre du modèle. L'intuition est qu'on souhaite $\mathbf{s} + \mathbf{p} \approx \mathbf{o}$ quand le fait $\langle s, p, o\rangle$ existe dans le graphe de connaissances, où $\mathbf{s}$, $\mathbf{p}$ et $\mathbf{o}$ correspondent respectivement aux représentations déduites pour l'entité s, la relation p et l'entité o.

Pour apprendre ces *embeddings*, étant donné S, l'ensemble des triplets $\langle s, p, o\rangle$ ($s, p \in$ E, l'ensemble d'entités) pour l'entraînement du modèle, *TransE* minimise la fonction objectif définie par l'équation (2.42) :

$$J = \sum_{(s,p,o)\in S} \sum_{(s',p,o')\in S'_{s,p,o}} \gamma + ||\mathbf{s} + \mathbf{p} - \mathbf{o}||_{1/2} - ||\mathbf{s}' + \mathbf{p} - \mathbf{o}'||_{1/2} \tag{2.42}$$

où $\gamma > 0$ est un hyper-paramètre de *margin* [Gilad-Bachrach et al., 2004] et où $S'_{s,p,o}$ est un ensemble de triplets négatifs (*e.g.* $\langle$*Pékin, capitale_de, Japon*$\rangle$) défini par :

$$S'_{s,p,o} = \{(s', p, o) | s' \in E\} \cup \{(s, p, o') | o' \in E)\}.$$

L'optimisation des paramètres du modèle peut être conduite par l'algorithme de décente de gradient stochastique (SGD), comme présenté dans la section des CF basés sur

les modèles 2.1.3.2. Une illustration graphique du *TransE* est représentée par la figure 2.11.

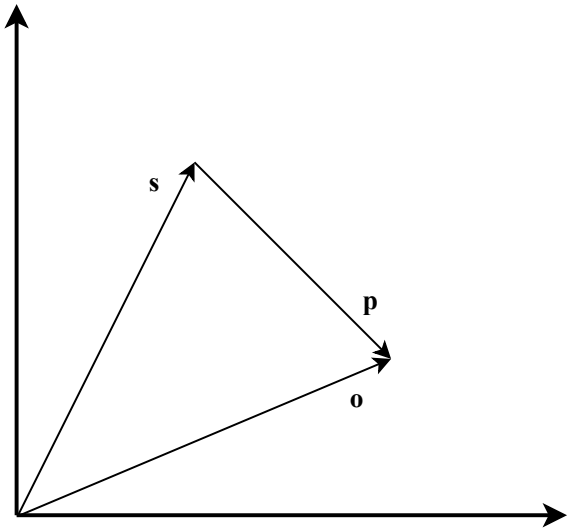


FIGURE 2.11 – Illustration du modèle d'*embeddings TransE*

Notons que le lecteur intéressé par d'autres modèles d'*embeddings* peut se référer à [Wang et al., 2017] pour une étude complète.

2.2.2.2.2 Recommandation comme complétion d'un graphe de connaissances Comme mentionné précédemment, la technique du plongement de graphe de connaissances permet d'estimer des représentations vectorielles pour les entités et les relations qui constituent le graphe. Une représentation vectorielle de l'ensemble des éléments au sein du graphe facilite sa manipulation, ainsi de nombreuses tâches peuvent en bénéficier, comme par exemple, la complétion du graphe en prédisant les relations *manquantes* entre des entités.

Les auteurs de [Palumbo et al., 2018b] proposent de considérer le problème de recommandation comme une approximation d'un problème de complétion du graphe de connaissances construit à partir des *ratings* (préférences) des utilisateurs ainsi que des connaissances liées aux items. Puisqu'ici on considère à la fois la matrice des *ratings* et les contenus (connaissances) des items, l'approche de [Palumbo et al., 2018b] peut être considérée comme un modèle de recommandation hybride.

La figure 2.12 illustre l'idée de considérer la recommandation des films comme un problème de complétion d'un graphe de connaissances. Dans ce graphe de connaissances, il y a trois types de nœuds : i) les utilisateurs (icônes) ; ii) les items (orange) et iii) les propriétés des items (bleu), et il y a deux types de relation : les caractéristiques des items (arêtes noirs) et les préférences des utilisateurs (les relations *like* représentées par les arêtes violets). Les relations représentées en pointillé signifient qu'il n'y a pas d'interaction entre l'utilisateur source de la relation et les items ciblés, autrement dit, elles indiquent les items que l'utilisateur n'a pas notés. La recommandation revient à compléter

ce graphe en prédisant les relations *manquantes* de type *préférence* entre les utilisateurs et les items avec lesquels ils n'ont pas interagi, par exemple entre l'utilisateur et le film *Captain* dans la figure 2.12.

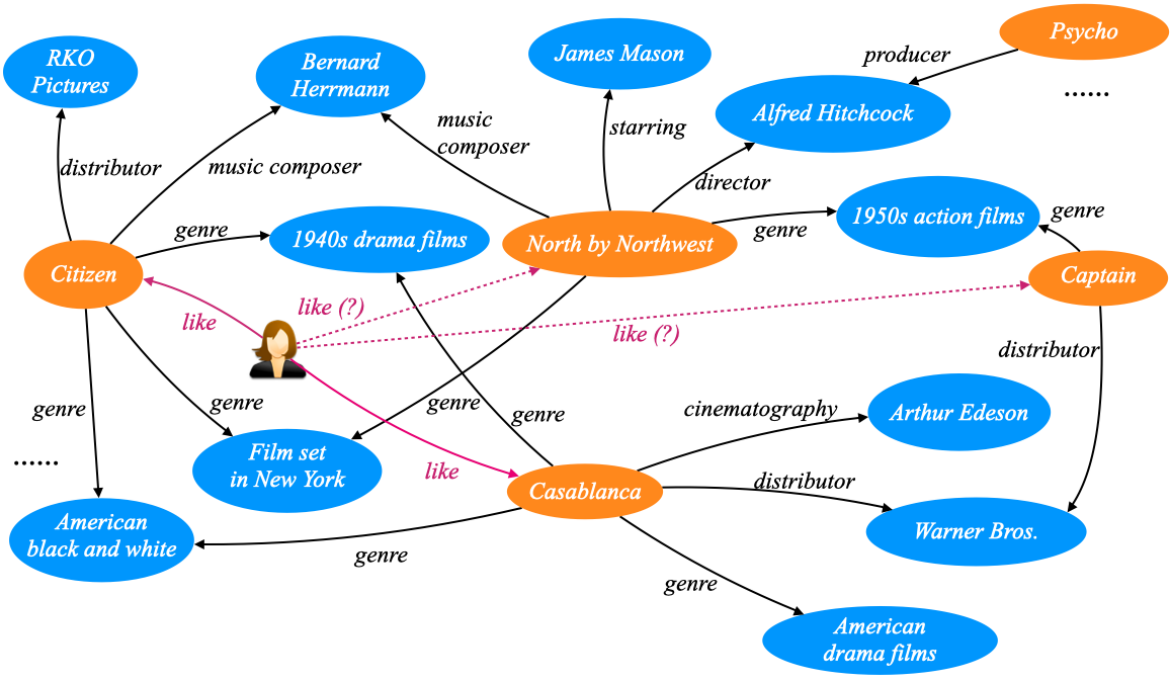


FIGURE 2.12 – La recommandation vue comme un problème de complétion du graphe de connaissances

Afin de résoudre ce problème, des modèles de KGE sont employés, afin d'apprendre pour chaque entité (*utilisateur, item, propriété*) et chaque relation (*caractéristique, préférence*), leur représentation vectorielle tout en préservant au maximum la sémantique du graphe de connaissances initial. Concrètement et formellement, étant donné un ensemble de *ratings* S contenant |U| utilisateurs et |I| items, l'approche de [Palumbo et al., 2018b] procède en quatre étapes :

1. *Aligner les items aux entités dans DBpedia* – Pour bénéficier les connaissances (faits) accessibles au travers du LOD, les items doivent être *alignés* aux ressources dans le Web sémantique. Notons ici que le terme d'*alignement* consiste généralement à associer un lien (*mapping*) entre deux concepts au sein de deux ontologies [Euzenat et al., 2010]. Dans notre cas, il s'agit d'identifier les entités qui représentent les items dans le catalogue du SdR. Pour ce faire, une méthode classique consiste à mesurer la similarité de *Levenshtein* entre le label d'un item (titre du film par exemple) et celui présent dans le KG cible au travers, par exemple, de la propriété *rdfs:label*.

2. *Construire le graphe de connaissances* – A partir des *ratings* des utilisateurs et des items alignés, on peut construire un graphe de connaissances (hybride) qui contient deux types d'information : les préférences des utilisateurs concernant les items (in-

formation collaboratives) et les connaissances liées aux items eux-mêmes (information sur leur contenu).

3. *Apprendre les embeddings des éléments du graphe* – La troisième étape consiste à apprendre les représentations vectorielles pour les éléments au sein du graphe hybride construit dans l'étape 2. Pour ce faire, Les auteurs de [Palumbo et al., 2018b] emploient TransE. L'apprentissage est basé sur l'ensemble des triplets du graphe de connaissances construit, *i.e.*

$$\{\langle s, p, o\rangle \,|\, s, o \in \mathrm{U} \cup \mathrm{I} \cup \textit{Caractéristiques}, \text{ et } p \in \textit{Propriétés} \cup \{\textit{like}\}\}$$

4. *Prédiction, ranking et recommandation* – A l'issue de l'étape précédente, nous disposons donc des représentations vectorielles pour chaque utilisateur u, chaque item i et la relation du type *préférence*, *i.e.* la relation *like*. De ce fait, étant donné l'utilisateur focal u et $\mathrm{L}_{candidats}(u)$ *i.e.* l'ensemble des items candidats pour l'utilisateur u, nous pouvons calculer, pour chaque triplet $\langle u, like, i\rangle$ avec $i \in \mathrm{L}_{candidats}(u)$, un score $s(u, like, i)$ quantifiant l'intérêt de u par rapport à l'item i. Pour ce faire, on peut mesurer par exemple la L_1-norme ou la L_2-norme du vecteur $\mathbf{u} + \boldsymbol{like} - \mathbf{i}$, *i.e.* $f(u, like, i) = ||\mathbf{u} + \boldsymbol{like} - \mathbf{i}||_{1/2}$. Notons que les normes mesurent des distances, or ici on souhaite trouver les top-N items i les proches de $\mathbf{u} + \boldsymbol{like}$, autrement dit, ceux ayant les plus petites normes. Ainsi, le score final pour l'item i est calculé par $s(u, like, i) = -f(u, like, i)$. Enfin, les N items ayant les meilleurs scores (prédictions) sont recommandés à l'utilisateur u.

Notons que les prédictions faites dans cette approche *i.e.* les scores prédits, n'ont pas pour objectif d'estimer la valeur de *rating* que mettrait l'utilisateur u pour l'item i. L'évaluation de cette approche ne peut donc pas reposer sur les métriques d'évaluation basées sur les prédictions des *ratings* (*i.e.* RMSE, MAE). Il s'agira ici d'un scénario qui évalue la liste des top-N items, comme présenté dans la section 2.1.6.3. L'évaluation de l'approche de [Palumbo et al., 2018b] a été basée sur deux jeux de données benchmark du domaine (*MovieLens* et *LibraryThing*), les résultats ont montré que l'approche proposée est meilleure que les modèles de l'état de l'art du filtrage collaboratif, *e.g.* SVD, NBCF, SVD++, etc. (voir section 2.1.3), en termes de précision, rappel, F1-mesure et MAP.

2.3 Conclusion de chapitre

Nous avons évoqué dans ce chapitre, les principes des systèmes de recommandation (SdR), outils efficaces qui facilitent la découverte d'items pouvant potentiellement nous intéresser, dans le contexte de la surcharge d'information auquel on doit faire face. Un SdR dispose de la capacité de prédire les goûts d'un utilisateur en fonction de son historique d'interaction avec le système. La prédiction peut se baser sur les goûts de l'utilisa-

teur focal (approche CBF) ou sur les goûts d'utilisateurs similaires (approche CF). Des approches hybrides ont été également proposées afin de bénéficier des avantages de chaque type d'approche et pallier leur inconvénients.

Initialement issus d'une vision orientée « *données* », les SdR peuvent être vus comme des cas particuliers de l'apprentissage automatique. En effet, la plupart des modèles de recommandation reposent sur une matrice de *ratings*, *i.e.* une structure de données classique pour l'apprentissage automatique. Le fait que cette matrice soit souvent creuse induit un problème de démarrage à froid qui peut être, au moins partiellement, résolu en prenant en compte la sémantique liée aux items. Les principes technologies d'ontologies et du Web sémantique possèdent un réel atout face aux inconvénients des SdR conventionnels.

Il est donc possible d'envisager une approche orientée « *connaissances* » ; les ontologies, le Web sémantique, *Linked Data* et les graphes de connaissances permettent de mieux représenter les utilisateurs ou/et les items et ainsi de mieux capturer les interactions entre utilisateurs et items.

Dans les chapitres suivants, nous allons suivre un chemin d'une vision orientée « *données* » à l'une autre orientée « *connaissances* », en découvrant l'amélioration de la performance des systèmes de recommandation, au travers de différents aspects des recommandations tels que l'*accuracy*, la diversité et l'explicabilité.

Chapitre 3

Recommandation plus pertinente ? Ajustement bayésien empirique des mesures de similarité pour améliorer les recommandations du filtrage collaboratif

Sommaire

3.1 Problématique 67

3.2 Contribution en termes de mesure de la similarité : EBCR 70

3.2.1 Prise en compte des spécificités des utilisateurs par un ratio de concordance : CR 70

3.2.2 Ajustement du ratio de concordance via le modèle bayésien empirique : EB 72

3.2.2.1 Inspiration : un exemple analogue en baseball traité par inférence bayésienne empirique 73

3.2.2.2 Bayésien empirique dans le contexte du ratio de concordance 75

3.3 Évaluation 77

3.3.1 Approches comparées 77

3.3.2 Jeux de données 77

3.3.3 Métriques d'évaluation 78

3.3.4 Protocole expérimental 78

3.3.4.1 Trois différents types de comparaison des méthodes 78

3.3.4.2 Découpage des jeux de données 79

3.3.4.3 Calibration des modèles comparés 80

3.4 Résultats et discussions 82

3.5 Exemple d'une étude de cas 87

3.6 Conclusion de chapitre 88

3.1 Problématique

Le filtrage collaboratif est une technique populaire dans le domaine des systèmes de recommandation. Comme présenté dans la section 2.1.3 du chapitre précédent, on distingue deux types de filtrage collaboratif (CF) : le CF basé sur le voisinage (NBCF) et le CF basé sur les modèles. Dans ce chapitre, nous allons nous concentrer sur les approches de CF basées sur le voisinage.

Comme montré dans la section 2.1.3.1, la procédure du filtrage collaboratif basé sur le voisinage est constituée de deux étapes consécutives : tout d'abord l'*identification du voisinage* puis la *prédiction des ratings*. Au cours de la première étape, l'objectif est d'identifier un ensemble d'individus (*utilisateurs* ou *items*) proches de l'utilisateur focal (respectivement proches de l'item). Ensuite les *ratings* associés aux individus du voisinage identifié sont utilisés pour prédire la valeur de *rating* en question.

Afin d'identifier les k plus proches voisins d'un utilisateur focal u, on utilise les mesures de similarité entre les vecteurs de notations (*ratings*) de l'utilisateur u et ceux des autres utilisateurs. Pour cela, des métriques de calcul de similarité telles que COS (cf. équation (2.1)), PCC (cf. équation (2.3)) et MSD (cf. équation (2.8)) sont employées. Rappelons que, pour simplifier la lecture, on ne mentionne ici que l'utilisateur u, mais l'identification des k plus proches voisins peut également être effectuée pour l'item i (dans l'approche NBCF basée sur les items, notamment).

Bien qu'elles soient largement utilisées dans le contexte du filtrage collaboratif basé sur le voisinage (aussi désigné comme basé sur la mémoire), ces mesures de similarité (*i.e.*, COS, PCC, MSD) possèdent des inconvénients majeurs [Mahara et al., 2016]. L'approche PCC (équation (2.3)), par exemple, estime la corrélation entre deux vecteurs correspondant à deux utilisateurs : u, l'utilisateur focal et v, un autre utilisateur quelconque. Pour cela, elle positionne d'abord chaque paire d'observation $(r_{u,i}, r_{v,i}), \forall i \in \mathrm{I}_{u,v}$ dans un espace à 2 dimensions, puis calcule une corrélation entre les utilisateurs u et v dans cet espace. Rappelons que $\mathrm{I}_{u,v}$ représente l'ensemble des items co-notés par les utilisateurs u et v. Mais ce procédé utilisé par le PCC pour mesurer la similarité entre u et v pose un problème de fiabilité lorsque $|\mathrm{I}_{u,i}|$ est petite, autrement dit, la similarité déduite n'est pas fiable lorsque deux utilisateurs n'ont co-noté que très peu d'items en commun.

Considérons l'exemple suivant : étant donné trois utilisateurs u, v et w et leur vecteurs de *ratings* respectifs $\overrightarrow{u} = (1,3,2,\emptyset,1)$, $\overrightarrow{v} = (1,\emptyset,\emptyset,5,\emptyset)$ et $\overrightarrow{w} = (1,2,2,\emptyset,1)$, qui représentent les goûts des utilisateurs u, v et w pour les 5 mêmes items. Avec l'approche de

la corrélation de *Pearson* (PCC), nous déduisons les similarités suivantes :

$$sim_{\text{PCC}}(u,v) = 1 > sim_{\text{PCC}}(u,w) = 0.905.$$

Autrement dit, l'utilisateur v jouera un rôle plus important que l'utilisateur w dans l'algorithme du filtrage collaboratif pour estimer les *ratings* de l'utilisateur u, puisqu'on considère ici que v est plus similaire à l'utilisateur focal que w. Cependant, cette estimation peut paraître contre-intuitive au vu des vecteurs de *ratings*. Nous constatons que la similarité estimée par le PCC entre u et v n'est basée que sur une seule observation (*i.e.* le premier item qui est le seul co-noté par u et v), alors que la similarité estimée entre u et w est basée sur 4 observations (*i.e.* l'ensemble des items ont été co-notés, sauf le quatrième). De ce fait, on peut être assez confiant sur le fait que les utilisateurs u et w ont des goûts assez similaires, alors que l'on ne dispose pas vraiment d'information sur le fait que u et v aient ou non des goûts similaires. Par conséquent, pour prédire la valeur de *rating* que u pourrait attribuer au quatrième item, il semble plus raisonnable d'utiliser les *ratings* de w plutôt que ceux de v.

Le même phénomène est également observé avec les métriques COS et MSD ; sur ce même exemple, nous avons en effet :

$$sim_{\text{COS}}(u,v) = 1 > sim_{\text{COS}}(u,w) = 0.9798$$

et

$$sim_{\text{MSD}}(u,v) = 1 > sim_{\text{MSD}}(u,w) = 0.80.$$

Ce phénomène vient du fait que ces métriques conventionnelles sont restreintes aux items co-notés par deux utilisateurs lors du calcul de similarité, tout en ignorant le nombre de ces items co-notés. Cette limitation rend les estimations de similarité assez peu fiables pour les utilisateurs qui ont peu d'items notés en commun, *e.g.* les utilisateurs u et v dans l'exemple précédent. En d'autres termes, en négligeant le nombre d'items co-évalués par les utilisateurs, ces mesures de similarité négligent le fait que la fiabilité de l'estimation de similarité entre utilisateurs augmente avec le nombre d'items qu'ils ont co-évalués et pose le problème de la gestion de l'incertitude qui en résulte. Prendre en compte cette incertitude pourrait rendre les calculs de voisinage plus fiables et les prédictions plus *justes*. Par ailleurs, il existe une autre forme d'incertitude : les items peuvent être notés différemment par un utilisateur ou un autre en fonction de leurs connaissances ou de leur caractère personnel. Il est donc également nécessaire de tenir compte de la disparité des distributions de *ratings* entre les utilisateurs [Herlocker et al., 2002].

On pourrait être tenté de pallier cet inconvénient lié à un ensemble d'items co-notés $\mathrm{I}_{u,i}$ très petit, en pénalisant les valeurs de similarité estimées pour les utilisateurs ayant noté peu d'items en commun. Dans les travaux de [Herlocker et al., 2002], les auteurs proposent une approche nommée facteur de pondération significative – ou *Significance*

Weighting (SW), visant à pénaliser la similarité entre deux utilisateurs donnés u et v quand le nombre d'items co-notés par u et v (*i.e.* $|\mathrm{I}_{u,v}|$) est inférieur à un seuil t préalablement défini. Formellement, étant donné $sim(u,v)$, la similarité estimée par une métrique et t, ce seuil de pénalisation, l'approche SW propose d'utiliser la similarité $sim_t(u,v)$ définie par l'équation (3.1).

$$sim_t(u,v) = \begin{cases} sim(u,v) \times \frac{|\mathrm{I}_{u,v}|}{t}, & \text{si } |\mathrm{I}_{u,v}| < t, \\ sim(u,v), & \text{sinon.} \end{cases} \tag{3.1}$$

Les résultats de l'approche proposée par [Herlocker et al., 2002] ont montré qu'en pénalisant les similarités estimées pour des utilisateurs ayant co-noté moins de t items, on améliore l'*accuracy* des prédictions en termes de MAE et de RMSE dans le contexte du filtrage collaboratif basé sur le voisinage. L'inconvénient de cette approche reste son manque potentiel de généricité. En effet, les auteurs n'ont évalué leur approche que pour la métrique PCC, en se basant sur un seul jeu de données. On ne sait pas comment il fonctionnerait avec d'autres mesures de similarité sur d'autres jeux de données. En outre, la détermination de la valeur du seuil t n'est pas triviale. Les auteurs conseillent de la fixer à 50 de façon empirique, après avoir essayé différentes valeurs de t. Là encore, la généricité peut être questionnée.

Les auteurs de [Polatidis and Georgiadis, 2016] ont étendu l'approche de [Herlocker et al., 2002] en proposant une pénalisation plus fine basée sur une fonction multi-niveaux (ML pour *Multi-Level*). Formellement, étant donné $sim_{\mathrm{PCC}}(u,v)$ la similarité entre u et v estimée par la métrique PCC, leur similarité sim_{ML} est définie par l'équation (3.2) suivante :

$$sim_{\mathrm{ML}}(u,v) = \begin{cases} sim_{\mathrm{PCC}}(u,v) + x_1, & \text{si } |\mathrm{I}_{u,v}| \geq t_1 \text{ et } sim_{\mathrm{PCC}}(u,v) \geq y, \\ sim_{\mathrm{PCC}}(u,v) + x_2, & \text{si } t_1 > |\mathrm{I}_{u,v}| \geq t_2 \text{ et } sim_{\mathrm{PCC}}(u,v) \geq y, \\ sim_{\mathrm{PCC}}(u,v) + x_3, & \text{si } t_2 > |\mathrm{I}_{u,v}| \geq t_3 \text{ et } sim_{\mathrm{PCC}}(u,v) \geq y, \\ sim_{\mathrm{PCC}}(u,v) + x_4, & \text{si } t_3 > |\mathrm{I}_{u,v}| \geq t_4 \text{ et } sim_{\mathrm{PCC}}(u,v) \geq y, \\ 0, & \text{sinon.} \end{cases} \tag{3.2}$$

où t_1, t_2, t_3, t_4 représentent les seuils utilisés pour chaque niveau ; x_1, x_2, x_3 et x_4 sont quatre valeurs pré-définies et y représente un seuil de similarité qui détermine la valeur de similarité minimum pour $sim_{\mathrm{PCC}}(u,v)$, autrement dit toute similarité inférieure est ignorée (sa valeur vaut 0). Les auteurs de [Polatidis and Georgiadis, 2016] ont fixé empiriquement les valeurs des paramètres comme suit : $(t_1, t_2, t_3, t_4) = (50, 20, 10, 5)$, $(x_1, x_2, x_3, x_4) = (0.5, 0.375, 0.25, 0.125)$ et $y = 0.33$. Les résultats de leur approche ont montré qu'elle permet d'améliorer la prédiction des *ratings*. Pourtant, cette approche possède le même inconvénient que celle proposée par [Herlocker et al., 2002] : son manque de généricité. En effet, l'approche proposée par [Polatidis and Georgiadis, 2016] se restreint à une seule

métrique de similarité dans son évaluation, bien qu'elle soit basée sur des jeux de données différents. Par conséquent, on ne connaît pas sa performance par rapport aux autres métriques de similarité telles que COS et MSD.

3.2 Contribution en termes de mesure de la similarité : EBCR

En réponse aux problématiques introduites dans la section précédente, *i.e.* (1) la disparité entre les distributions de *ratings* des utilisateurs ; (2) la non-prise en compte du nombre d'items co-notés lors de l'estimation de la similarité entre utilisateurs avec les mesures usuelles, pouvant entraîner un manque de fiabilité des similarités estimées pour des paires d'utilisateurs ayant co-noté peu d'items et (3) la non-généricité des méthodes proposées dans la littérature, nous présentons dans cette section une approche d'ajustement des similarités entre utilisateurs dans le contexte du NBCF. Nous proposons et détaillons tout d'abord la notion de *ratio de concordance* (*concordance ratio*), notée CR par la suite. Ensuite, nous décrivons une approche d'ajustement basée sur une méthode d'inférence bayésienne empirique (*Empirical Bayes*), notée EB. En combinant ces deux parties, CR et EB, nous obtenons la nouvelle approche que nous proposons : EBCR pour *Empirical Bayes Concordance Ratio*. Cette méthode a fait l'objet d'un article publié dans la conférence LFA [Du et al., 2019a] ainsi que la revue PLOS ONE [Du et al., 2021c].

3.2.1 Prise en compte des spécificités des utilisateurs par un ratio de concordance : CR

En général dans le contexte du filtrage collaboratif, deux utilisateurs sont considérés comme similaires s'il existe une forte concordance entre leur profils, *i.e.* entre leur *ratings*. Or l'expression des goûts peut être très différente d'une personne à une autre, un utilisateur "indulgent" mettant rarement le *rating* le plus bas même s'il n'aime pas un item et à contrario, un utilisateur "sévère" mettant rarement le meilleur *rating*, même pour un item qu'il apprécie [Herlocker et al., 2002]. L'information relative au goût semble donc difficile à capturer et donc à représenter. Ceci motive la discrétisation des *ratings* dans un espace de petite dimension (2 ou 3). Nous proposons donc tout d'abord de discrétiser les *ratings* des utilisateurs en trois catégories : *aime, n'aime pas* et *neutre*, reflétant le goût d'un utilisateur pour un item donné, comme défini dans la définition 1. Pour résoudre le problème lié à la disparité des distributions de *ratings* entre utilisateurs, nous réalisons cette discrétisation en nous basant sur le *z-score* (cf. section 2.1.3.1.2) comme détaillé dans l'équation (3.3). Le paramètre $a \in [0,1]$ utilisé dans l'équation (3.3) est un hyper-paramètre, qui permet de gérer la taille de la classe (zone) *neutre*.

Définition 1. (**Discrétisation des goûts des utilisateurs**). *Soit* $a \in [0,1]$, *le goût de l'utili-*

sateur u pour l'item i, discrétisé par l'opérateur T (pour *Taste*), *est défini comme suit* :

$$\mathrm{T}(r_{u,i}) = \begin{cases} \textit{aime}, & \text{si } \frac{r_{u,i}-\overline{r}_u}{\sigma_u} > a, \\ \textit{n'aime pas}, & \text{si } \frac{r_{u,i}-\overline{r}_u}{\sigma_u} < -a, \\ \textit{neutre}, & \text{sinon.} \end{cases} \tag{3.3}$$

Nous nous appuyons sur cette discrétisation pour définir la notion, ou le concept, de *concordance* entre les goûts des utilisateurs comme explicitée dans la définition 2. Le terme de *concordance* que nous proposons est lié aux *ratings* de deux utilisateurs pour un seul (même) item et nous utilisons le terme « *ratio de concordance* » pour la mesure de la concordance globale des profils des deux utilisateurs, telle que définie en définition 3.

Définition 2. (**Concordance des goûts**). *Soient $u \in \mathrm{U}$ et $v \in \mathrm{U}$ deux utilisateurs et $i \in \mathrm{I}_{u,v}$ l'un des items co-notés par u et v. La paire de ratings $(r_{u,i}, r_{v,i})$ est définie comme concordante, si et seulement si $\mathrm{T}(r_{u,i}) = \mathrm{T}(r_{v,i})$. L'ensemble des items associés aux ratings concordants pour les utilisateurs u et v est dénoté :*

$$\mathrm{C}_{u,v} = \{i \in \mathrm{I}_{u,v} \mid \mathrm{T}(r_{u,i}) = \mathrm{T}(r_{v,i})\}.$$

Cette définition de la *concordance* est proche de celle de [Lathia et al., 2007], dans laquelle les auteurs utilisent la notion de concordance relativement à la problématique de confidentialité (*privacy*) dans les systèmes de recommandation. En effet, dans l'approche de NBCF classique, les utilisateurs sont tenus de partager entre eux leur historique de notation afin de calculer leur similarité, cela peut causer des problèmes de confidentialité.

Définition 3. (**Ratio de concordance**). *En considérant les utilisateurs $u \in \mathrm{U}$ et $v \in \mathrm{U}$, le ratio de concordance* $\mathrm{CR}(u,v)$ *correspond à la proportion des items co-notés d'une manière concordante par u et v parmi tous les items qu'ils ont co-notés :*

$$\mathrm{CR}(u,v) = \frac{|\mathrm{C}_{u,v}|}{|\mathrm{I}_{u,v}|}.$$

Le ratio de concordance défini ci-dessus pour les utilisateurs u et v peut être interprété comme la probabilité que u et v aient la même appréciation pour un nouvel item. Nous proposons ainsi de nouvelles mesures de similarité nommées sim_{CR}, définies comme le produit du ratio de concordance calculé avec l'une des mesures de similarité présentées précédemment, *i.e.* PCC (cf. équation (2.3)), COS (cf. équation (2.1)) et MSD (cf. équation (2.8)). Cette métrique $sim_{\mathrm{CR}}(u,v)$ (cf. équation (3.4)) correspond à une pondération de la

similarité originale $sim(u,v)$, en fonction de la concordance des goûts entre l'utilisateur u et l'utilisateur v :

$$sim_{\text{CR}}(u,v) = \text{CR}(u,v) \times sim(u,v) \tag{3.4}$$

L'avantage de ce ratio de concordance est qu'il permet de pallier le problème lié à la disparité des comportements de notation entre différents utilisateurs, puisque la discrétisation des goûts est conduite de la même façon pour tous les utilisateurs en considérant leurs distributions spécifiques des *ratings*. Par exemple, en considérant une échelle de notation entre 1 et 5, un *rating* de 3 peut traduire des goûts différents (*aime, n'aime pas* ou *neutre*) selon la distribution des *ratings* de l'utilisateur en question. Ainsi, pour un utilisateur optimiste u, $r_{u,i} = 3$ peut exprimer qu'il *n'aime pas* l'item i (ou qu'il n'a pas d'avis, *i.e.* qu'il reste *neutre* vis à vis de cet item) étant donné qu'il donne très souvent de bonnes évaluations aux items, alors que pour un autre utilisateur v plutôt pessimiste, $r_{v,i} = 3$ peut traduire qu'il *aime* l'item i.

3.2.2 Ajustement du ratio de concordance via le modèle bayésien empirique : EB

Bien que le ratio de concordance proposé dans la section précédente, $\text{CR}(u,v)$, permette de prendre en compte la disparité des distributions des *ratings* des utilisateurs, cette façon de modéliser et de mesurer la concordance entre deux utilisateurs pose toujours un problème de fiabilité. En effet, les valeurs de $\text{CR}(u,v)$ ne sont pas toujours fiables, comme en témoigne l'exemple suivant. Considérons les utilisateurs u, v et w, et les ratios de concordance correspondant à leurs évaluations :

$$\text{CR}(u,v) = \frac{|\text{C}_{u,v}|}{|\text{I}_{u,v}|} = \frac{3}{3} = 1 \text{ et } \text{CR}(u,w) = \frac{|\text{C}_{u,w}|}{|\text{I}_{u,w}|} = \frac{290}{300} = 0.97.$$

Les utilisateurs u et v semblent avoir des goûts plus similaires que ceux des utilisateurs u et w, puisque $1 > 0.97$. Autrement dit, les profils de u et v semblent plus concordants que ceux de u et w. Cependant, u et v n'ont partagé leurs avis que sur trois items, et il est donc fort possible que cette valeur de ratio élevée soit uniquement le fruit du hasard et que ces deux utilisateurs aient une opinion complètement discordante concernant un quatrième item. Tout calcul de CR entre eux est donc nécessairement très incertain et pourrait profiter d'un ajustement (lissage). De ce fait, pour prédire les *ratings* de l'utilisateur u, il semble préférable de se baser sur les *ratings* de w plutôt que sur ceux de v. Même si la valeur de $\text{CR}(u,w)$ est un peu plus faible que celle de $\text{CR}(u,v)$, son estimation est beaucoup plus fiable puisqu'elle repose sur des centaines d'évaluations. Cet exemple souligne la nécessité d'un ajustement des ratios de concordance, qui devrait tenir compte du nombre d'items co-évalués par les utilisateurs afin de pouvoir *pénaliser* les ratios de concordance basés sur un petit nombre $|\text{I}_{u,v}|$ d'items co-notés.

3.2.2.1 Inspiration : un exemple analogue en baseball traité par inférence bayésienne empirique

Supposons que vous êtes chargé de sélectionner un joueur de baseball parmi deux candidats en fonction de leurs nombres respectifs de coups réussis dans le passé. Le premier joueur a réussi 4 coups en 10 essais, le deuxième a réussi 300 coups en 1000 essais. Même si le premier joueur dispose d'une proportion de réussite plus élevée que le deuxième, *i.e.* $0.4 > 0.3$, le choix reste difficile : il est connu qu'un joueur de niveau moyen a tendance à réussir un coup dans 26% des cas, la proportion de $\frac{4}{10}$ de ce premier joueur pourraient donc être sur-évaluée par hasard. Le deuxième joueur, en revanche, ayant un nombre de succès important sur un nombre élevé d'essais, apporte plus de preuves de sa valeur technique.

[Brown, 2008] propose une approche visant à résoudre le problème de la non-fiabilité des estimations du taux de réussite des joueurs dans le contexte du baseball. L'objectif de cette approche est de corriger les proportions estimées, pour les joueurs qui n'ont pas beaucoup joué pendant une saison, en s'appuyant sur la méthode de bayésien empirique qui procède en deux étapes, comme suit :

1. *Estimation d'un apriori bayésien empirique à partir de l'ensemble des proportions* – En considérant la proportion de $\frac{\textit{nombre de tirs réussis}}{\textit{nombre total de tirs}}$ comme la probabilité qu'un joueur de baseball réussisse son tir, on dispose de l'ensemble des probabilités estimées pour chaque joueur de la base de données. Nous considérons ensuite que l'ensemble des proportions observées $\mathscr{X}$ suit une distribution de la loi $Beta(\alpha_0, \beta_0)$, *i.e.* la loi de probabilité classique d'une proportion, qui peut être utilisée comme apriori :

$$\mathscr{X} \sim Beta(\alpha_0, \beta_0)$$

   où α_0 et β_0 représentent deux hyper-paramètres qui déterminent sa forme (cf. figure 3.1). Dans notre cas, les valeurs de α_0 et β_0 sont estimées empiriquement à partir de l'ensemble des observations (probabilités) dont on dispose, *i.e.* $\mathscr{X}$. Autrement dit, on cherche à déterminer les α_0 et β_0 qui permettent d'obtenir une distribution qui représente le mieux possible les données observées.

   D'un point de vue statistique, $\frac{\alpha_0}{\alpha_0+\beta_0}$ correspond à l'espérance de la distribution $Beta(\alpha_0, \beta_0)$. Cette information peut être considérée comme une connaissance *a priori*. Autrement dit, pour un joueur de baseball quelconque, avant qu'il ne joue, nous pouvons penser qu'il y a $\frac{\alpha_0}{\alpha_0+\beta_0}$ chances qu'il réussisse son tir. Cette distribution *Beta* des probabilités observées est considérée comme un *prior*.

2. *Utilisation de l'apriori bayésien empirique pour mettre à jour chaque estimation* – Dans la deuxième étape, on met-à-jour, joueur par joueur, chaque proportion observée de tirs réussis dans le jeu de données. Pour cela, on s'appuie sur la distribution $Beta(\alpha_0, \beta_0)$ obtenue à l'étape précédente. Cette mise à jour est effectuée par

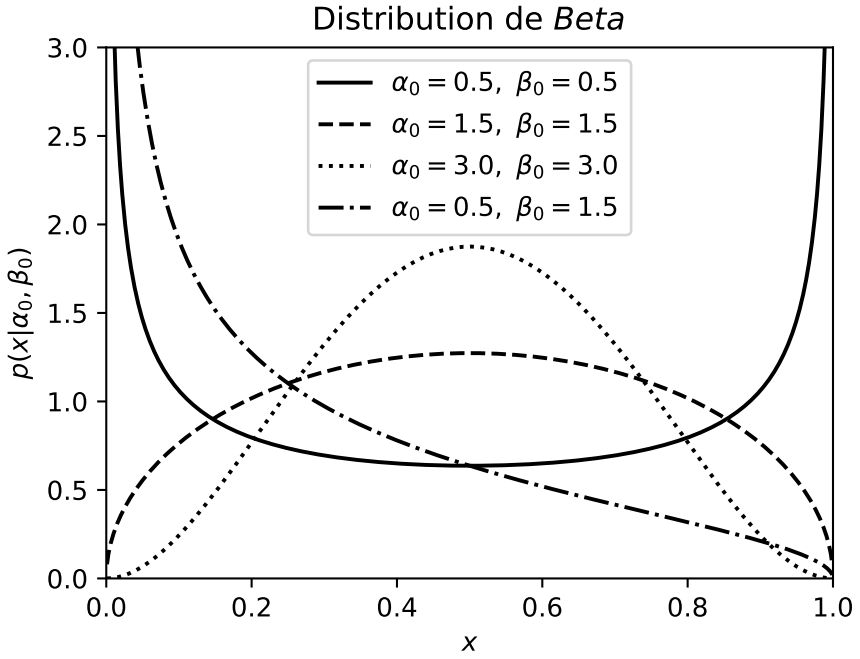


FIGURE 3.1 – Exemple des distributions de *Beta* avec des hyper-paramètres différents (Source : Wikipedia)

l'équation suivante :

$$\frac{\textit{nombre de tirs réussis}}{\textit{nombre total de tirs}} \Longrightarrow \frac{\textit{nombre de tirs réussis} + \alpha_0}{\textit{nombre total de tirs} + \alpha_0 + \beta_0}.$$

Revenons sur l'exemple mentionné au début de cette section. Supposons que $(\alpha_0, \beta_0) = (78.7, 224.9)$ pour la distribution des probabilités des tirs réussis des joueurs de baseball. L'espérance de la distribution est donc $\frac{78.7}{78.7+224.9} = 0.259$. De ce fait, la mise à jour des probabilités de tirs réussis pour les deux joueurs est faite comme suit. Pour le premier joueur :

$$\frac{4}{10} \Longrightarrow \frac{4 + 78.7}{10 + 78.7 + 224.9} = 0.264$$

On remarque que, dans ce cas (peu d'observations observées), l'estimation de sa probabilité de réussite (0.264) a été fortement diminuée par rapport à sa valeur initiale (0.4). Pour le deuxième joueur, on obtient :

$$\frac{300}{1000} \Longrightarrow \frac{300 + 78.7}{1000 + 78.7 + 224.9} = 0.29$$

Dans ce cas (proportion basée sur un grand nombre d'observations), l'estimation de la probabilité de réussite du joueur est restée quasiment identique (0.3 v.s. 0.29). Elle est supérieure à celle du premier joueur et ce joueur pourraient donc, effectivement, être un meilleur choix pour un recruteur. Nous retiendrons donc que lorsque la proportion est basée sur peu d'observations, elle sera fortement corrigée au mo-

ment de l'ajustement bayésien empirique ; elle va tendre vers l'espérance de la distribution *Beta* estimée à partir de l'ensemble des données, alors que si la proportion repose sur de nombreuses observations, elle restera presque inchangée.

3.2.2.2 Bayésien empirique dans le contexte du ratio de concordance

Le ratio de concordance $\mathrm{CR}(u,v)$ peut être interprété comme la probabilité que deux utilisateurs u et v aient le même avis concernant un item donné. Afin de prendre en compte l'incertitude et la non-fiabilité des $\mathrm{CR}(u,v)$ basés sur peu d'items co-notés, ces valeurs des ratios $\mathrm{CR}(u,v)$ nécessitent un lissage (ou un ajustement). Le lissage de *Laplace*, aussi appelé *lissage additif* [Chen and Goodman, 1999], est largement utilisé dans le domaine de l'apprentissage automatique afin de lisser les estimations de probabilité multinomiales en fonction de la taille de l'échantillon considéré. Dans le contexte de l'ajustement des ratios de concordance, la méthode du lissage par *Laplace* est définie par l'équation (3.5), où α est le paramètre de lissage. Par exemple $\alpha = 0$ signifie qu'il n'y pas de lissage sur la probabilité.

$$Laplace(u,v) = \frac{|\mathrm{C}_{u,v}| + \alpha}{|\mathrm{I}_{u,v}| + 2 \cdot \alpha} \tag{3.5}$$

D'un point de vue bayésien, *Laplace* correspond à l'espérance *a posteriori* de la proportion considérée en utilisant une distribution *a priori* $Beta(\alpha,\alpha)$. Autrement dit, il correspond à la mise à jour d'une probabilité *Beta* étant donné une distribution *a priori* non-informative (*e.g.* uniforme en prenant $\alpha = 1$).

Comme dans l'approche de [Brown, 2008] qui met à jour les probabilités de tirs réussis pour les joueurs de baseball, nous proposons d'utiliser toutes les informations contenues dans l'ensemble de l'échantillon pour déterminer les paramètres α_0 et β_0 à utiliser lors de l'ajustement des ratios de concordance. Autrement dit, au lieu d'employer α et $2 \cdot \alpha$ pour mettre à jour les ratios, comme le fait le lissage de *Laplace* ci-dessus, nous déterminons les valeurs de α_0 et de β_0 à partir de tous les ratios observés dans un jeu de données. Par analogie avec les travaux de [Brown, 2008], nous proposons d'ajuster de la même manière les ratios de concordance entre les utilisateurs :

$$\frac{\textit{nombre d'items co-notés de façon concordante}}{\textit{nombre total d'items co-notés}} \Longleftrightarrow \frac{\textit{nombre de tirs réussis}}{\textit{nombre total de tirs}}.$$

Spécifiquement, nous supposons que les ratios de concordance observés sur l'ensemble des paires d'utilisateurs du jeu de *ratings*, *i.e.* $\mathrm{CR_U} = \{\mathrm{CR}(u,v) | (u,v) \in \mathrm{U}^2, v \neq u, \mathrm{I}_{u,v} \neq \emptyset\}$ suivent une distribution $Beta(\alpha_0,\beta_0)$, qui peut être utilisée comme une distribution *a priori*. α_0 et β_0 sont les deux hyper-paramètres de la distribution, qui déterminent sa forme. L'estimation des paramètres α_0 et β_0 peut être effectuée par la méthode du *maximum de vraisemblance* : $(\alpha_0,\beta_0) = \arg\max_{\alpha>0,\beta>0} \mathrm{L}\Big((\alpha,\beta)|\mathrm{CR_U}\Big)$, où L est la fonction de vraisemblance de (α,β) sachant les données de concordance observées $\mathrm{CR_U}$.

Ensuite, chaque valeur de ratio de concordance est mise à jour (ou ajustée), en se basant sur la distribution *a priori Beta*(α_0, β_0). La distribution *a posteriori* correspondante déplace plus ou moins le ratio vers la valeur moyenne théorique des proportions observées, *i.e.* l'espérance de la distribution observée. De ce fait, un ratio CR(u, v) basé sur peu d'items co-notés, *e.g.* $\frac{3}{3}$, sera largement corrigé et tendra vers l'espérance de la distribution *Beta*, *i.e.* $\frac{\alpha_0}{\alpha_0+\beta_0}$, alors qu'un ratio CR($u, w$) basé sur de nombreux items co-noté, *e.g.* $\frac{290}{300}$ restera pratiquement inchangé.

Cette approche d'ajustement des ratios de concordance permet de prendre en compte le nombre $|I_{u,v}|$ d'items co-notés par deux utilisateurs, une information généralement ignorée par les métriques de similarité usuelles. En outre, cet ajustement tire parti des informations relatives aux goûts de tous les autres utilisateurs (lors de l'estimation de α_0 et β_0), ce qui pourrait être considéré comme une étape collaborative supplémentaire. Ainsi, nous définissons le *Ratio* de *Concordance* ajusté de façon *Bayésienne Empirique* (EBCR), comme la version lissée ou ajustée de CR(u, v) selon de manière bayésienne empirique :

$$\text{EBCR}(u, v) = \frac{|C_{u,v}| + \alpha_0}{|I_{u,v}| + \alpha_0 + \beta_0} \tag{3.6}$$

Enfin, n'importe quelle mesure de similarité pourra être ajustée en la multipliant avec le ratio de concordance ajusté, *i.e.* EBCR (comme pondération). Par exemple, la similarité EBCR_COS dénote la mesure ajustée pour la métrique de similarité cosinus (équation (2.1)), comme illustré par l'équation (3.7).

$$sim_{\text{EBCR_COS}}(u, v) = \text{EBCR}(u, v) \times sim_{\text{COS}}(u, v) \tag{3.7}$$

Notons que l'estimation de α_0 et de β_0 est une étape de pré-traitement qui n'est effectuée qu'une seule fois. Autrement dit, une fois déterminées à partir des données de l'ensemble des utilisateurs, les valeurs de α_0 et β_0 sont utilisées pour ajuster l'ensemble des similarités, sans avoir besoin de les re-calculer à chaque fois. Notons également que $|I_{u,v}|$ doit être identifié pour pouvoir calculer $sim_{\text{COS}}(u, v)$, $|C_{u,v}|$ est calculé avec une complexité en $\mathcal{O}(|I_{u,v}|)$, comme c'est également le cas pour $sim_{\text{COS}}(u, v)$. Par conséquent, le calcul de la similarité entre u et v possède la même complexité en termes de temps de calcul, qu'il intègre ou non la pondération de EBCR.

Dans ce qui suit, nous allons présenter l'évaluation de l'approche proposée (EBCR), pour ajuster des similarités entre utilisateurs dans le contexte du filtrage collaboratif basé sur le voisinage (NBCF). Notons que cet ajustement peut également être appliqué aux similarités entre items ($sim(i, j)$) dans une approche de NBCF basée sur les items de manière à pénaliser les cas où peu d'utilisateurs ont noté à la fois deux items donnés i et j.

3.3 Évaluation

3.3.1 Approches comparées

Cette section présente brièvement les approches comparées. Tout d'abord, nous souhaitons évaluer l'impact de l'ajustement proposé (*i.e.* EBCR) appliqué aux mesures de similarité conventionnelles (*e.g.* cosinus) pendant la phase de prédiction des *ratings* dans le cadre du filtrage collaboratif basé sur la voisinage (NBCF). Ainsi, nous comparons chaque méthode de NBCF utilisant des mesures de similarité conventionnelles avec celle qui utilise les mesures de similarité ajustées par EBCR.

Ensuite, nous comparons la méthode EBCR proposée avec d'autres méthodes d'ajustement de similarité présentées précédemment, c'est-à-dire l'approche du *facteur de pondération significative* (*significance weighting*) (cf. équation (3.1)) de [Herlocker et al., 2002], l'approche de [Polatidis and Georgiadis, 2016] dans laquelle l'ajustement est basé sur une fonction multi-niveau (cf. équation (3.2)) et l'approche de lissage de *Laplace* ou *additif* (cf. équation (3.5)).

Troisièmement, nous souhaitons également élargir notre évaluation en comparant les performances prédictives de l'approche proposée de CF basée sur le voisinage avec celles des approches basées sur les modèles. Ces dernières s'appuient principalement sur des techniques de réduction de dimensions, telles que la factorisation matricielle, et ont été montrées pertinentes et flexibles, notamment lorsque la matrice de *ratings* est creuse [Koren, 2010]. À cette fin, nous avons choisi de compléter les expérimentations d'évaluation en considérant les modèles suivants : les modèles *baseline* (voir équation (2.19)), SVD (voir équation (2.23)), SVD++ (voir équation (2.25)) et le modèle NeuMF (*Neural Matrix Factorization*) [He et al., 2017], qui est basé sur des réseaux de neurones profonds.

3.3.2 Jeux de données

L'évaluation de l'approche d'ajustement proposée repose sur trois jeux de données benchmark du domaine des systèmes de recommandation : 1) le jeu de données *MovieLens-100K*, qui contient 100 000 *ratings* de 943 utilisateurs sur 1 682 films ; 2) le jeu de données *MovieLens-1M*, qui représente une collection d'un million de *ratings* attribués par 6 040 utilisateurs à 3 900 films et 3) le jeu de données *Jester*, qui contient plus de 1.7 million de *ratings* de 140 blagues attribués par 59 132 utilisateurs anonymes. Les deux premiers jeux de données (*ratings* des films), dans lesquels les *ratings* sont compris entre 1 et 5, ont été collectés par le centre de recherche *GroupLens* de l'Université du *Minnesota*[1]. Le troisième jeu de données de blagues *Jester*, dans lequel les *ratings* sont spécifiés sur une échelle continue de -10 à 10, a été dérivé du système de recommandation de blagues *Jester*[2].

[1] https://grouplens.org/
[2] https://goldberg.berkeley.edu/jester-data/

Ces trois jeux de données sont des références de la communauté des SdR [Harper and Konstan, 2015]. Les caractéristiques de ces jeux de données sont récapitulées dans le tableau 3.1.

TABLEAU 3.1 – Caractéristiques des jeux de données pour l'évaluation de l'approche EBCR

Jeu de données	**#utilisateurs**	**#items**	***#ratings***	**échelle de *rating***	**densité**	**domaine**
MovieLens-100K	943	1 682	100K	[1, 5]	6.30%	Movie
MovieLens-1M	6 040	3 900	1M	[1, 5]	4.47%	Movie
Jester	59.1K	140	1.7M	[-10, 10]	20.53%	Joke

La densité d'un jeu de données, *i.e.* l'inverse de la rareté (*sparsity*), représente le pourcentage de cases non-vides dans la matrice de *ratings*

3.3.3 Métriques d'évaluation

Comme introduit dans le chapitre précédent, l'évaluation de l'*accuracy* des systèmes de recommandation est généralement effectuée par deux familles de métriques : celles basées sur la prédiction des *ratings* et celles basées sur la liste des top-N recommandations. La méthode des ratios de concordance ajustés de façon bayésienne empirique (EBCR) qu'on propose ici appartient à la famille des approches de filtrage collaboratif basé sur le voisinage. Son objectif est d'améliorer l'*accuracy* des prédictions des *ratings* que les utilisateurs mettraient pour les items candidats. De ce fait, nous avons donc adopté deux métriques d'évaluation classiques des problèmes de régression : le RMSE (cf. équation (2.30)) et le MAE (cf. équation (2.31)).

3.3.4 Protocole expérimental

Cette section détaille le protocole expérimental utilisé pour évaluer l'approche EBCR.

3.3.4.1 Trois différents types de comparaison des méthodes

Comme mentionné au début de la section 3.3.1, l'évaluation de l'approche EBCR est réalisée selon les trois critères suivants :

1. *L'impact de l'ajustement par EBCR pour différentes mesures de similarité usuelles au cours du processus de NBCF.* — Pour ce faire, nous comparons les approches de filtrage collaboratif utilisant les métriques de similarité suivantes sim_{COS} (cf. équation (2.1)), sim_{MSD} (cf. équation (2.8)) et sim_{NormPCC} (cf. équation (3.8)) avec ou sans l'ajustement par EBCR. Notons ici que contrairement aux métriques sim_{COS} et sim_{MSD} dont les valeurs sont comprises entre [0, 1], le domaine de la fonction sim_{PCC} (cf. équation (2.3)) est entre -1 et 1. Cette différence pourrait impacter l'évaluation de la procédure d'ajustement. De ce fait, nous choisissons de normaliser linéairement le *coefficient de corrélation de Pearson* pour obtenir des valeurs dans l'intervalle de

[0, 1] (voir équation (3.8)). La mesure résultante est nommée *NormPCC* (pour coefficient de corrélation de Pearson normalisé) :

$$sim_{\text{Norm}PCC}(u,v) = \frac{sim_{PCC}(u,v)+1}{2} \tag{3.8}$$

Nous adoptons les noms des métriques de similarité pour dénoter les approches basiques (*e.g. COS_KNN* pour l'approche de type NBCF avec similarité cosinus). Notons que le terme « *KNN* » (*k-nearest neighbors*) est combiné avec le label de la similarité puisque différentes tailles de voisinages sont considérées dans l'évaluation. Respectivement, le terme « EBCR » est ajouté pour dénoter les approches intégrant l'ajustement par EBCR (*e.g. EBCR_COS_KNN*). Ce type de comparaison permet d'évaluer la pertinence et la généricité de l'approche d'ajustement proposée par la multiplicité des jeux de données et mesures de similarité considérés.

2. *Comparaison avec d'autres méthodes d'ajustement de similarité proposées dans l'état de l'art.* — Pour ce faire, nous comparons l'approche EBCR avec trois méthodes d'ajustement existantes : (1) l'approche dite de facteur de pondération significative (*significance weighting factor*, cf. équation (3.1)) ; (2) l'approche d'ajustement multi-niveau (*Multi-Level collaborative filtering*, cf. équation (3.2)) et (3) le lissage additif (*Laplace Smoothing*, cf. équation (3.5)). Ces trois approches sont respectivement dénotées *SW* (*e.g. SW_COS_KNN*), *MLCF* (*e.g. MLCF_COS_KNN*) et *LS* (*e.g. LS_COS_KNN*).

3. *Comparaison avec les approches de CF basées sur les modèles.* — Pour cela, nous comparons l'approche EBCR avec les modèles de l'état de l'art suivants : *baseline*, SVD, SVD++ et NeuMF. Les détails de ces modèles sont présentés dans le chapitre précédent (cf. section 2.1.3.2). Pour ce type de comparaison, nous dénotons l'approche d'ajustement proposée par le terme « EBCR ».

3.3.4.2 Découpage des jeux de données

Rappelons que dans le chapitre précédent (cf. section 2.1.6.4) nous avons évoqué trois méthodes classiquement utilisées pour découper un jeu de données en jeu d'entraînement (S_{train}) et jeu de test (S_{test}) : *hold-out*, *leave-one-out* et *validation croisée*. Ici, pour l'évaluation de l'approche EBCR associée à un NBCF, nous adoptons un découpage basé sur une validation croisée à 5-couches. Pour chacun des trois jeux de données considérés, nous avons séparé aléatoirement le jeu de données en 5 sous-échantillons de même taille et en avons sélectionné un successivement comme échantillon de test (S_{test}), tandis que les quatre autres ont été utilisés comme jeu d'entraînement (S_{train}). Nous avons obtenu ainsi 5 valeurs de MAE et et de RMSE différentes. Toutes les approches comparées ont été entraînées sur les mêmes jeux d'entraînement et évaluées sur les mêmes jeux de test. Les

moyennes des 5 valeurs de MAE et de RMSE ont été finalement considérées pour chaque modèle comparé.

3.3.4.3 Calibration des modèles comparés

L'approche EBCR proposée comporte deux hyper-paramètres principaux : 1) la taille k du voisinage (*i.e.* nombre de voisins dont les *ratings* sont utilisées pour la prédiction des *ratings* de l'utilisateur focal) et 2) le paramètre a de l'équation (3.3), qui configure la taille de la zone *neutre* dans la modélisation des goûts des utilisateurs.

La taille du voisinage est un paramètre utilisé tant pour les approches de NBCF basiques (*e.g. COS_KNN*) que pour les approches intégrant un ajustement (*SW_COS_KNN*, *MLCF_COS_KNN* et *LS_COS_KNN*). Pour chaque jeu de données, nous avons considéré 8 tailles de voisinage différentes : {5, 10, 20, 40, 60, 80, 100, 200} pour les deux jeux de *ratings* de films et {5, 10, 15, 20, 25, 30, 35, 40} pour le jeu de données de blagues (*Jester*). Comme nous allons discuter dans le paragraphe suivant, cette différence est liée au fait que nous avons considéré des approches de NBCF basée sur les utilisateurs pour les jeux de données de films (943 utilisateurs pour MovieLens-100k et 6,040 utilisateurs pour MovieLens-1M) et des approches NBCF basées sur les items pour le jeu de *ratings* de blagues (140 items au total).

Pour la valeur du paramètre a de l'approche proposée (cf. équation (3.3)), nous l'avons fixé empiriquement à 0.5 après avoir observé qu'il avait un impact négligeable sur les résultats. Plus précisément, pour déterminer la valeur du paramètre a dans l'équation (3.3), des expérimentations ont été menées sur le jeu de données MovieLens-1M avec différentes valeurs de a pour différentes tailles de voisinage. D'après les résultats obtenus (cf. figure 3.2), nous avons observé que la variation du paramètre a ne semble pas avoir un impact significatif sur les performances prédictives des modèles et la valeur de $a = 0.5$ semble à peu près optimale lorsque la taille du voisinage considérée est petite.

Un autre paramètre des approches de filtrage collaboratif NBCF basées sur la mémoire est lié à la manière de construire la matrice de similarité, c'est-à-dire basée sur les utilisateurs ou sur les items. Pour le jeu de données de blagues *Jester*, les expérimentations ont été menées avec un filtrage collaboratif basé sur les items afin de faciliter les calculs. Comme le jeu de données de blagues ne contient que 140 items comparés à 59.1K utilisateurs, il est plus judicieux d'adopter l'approche basée sur les items, la taille de la matrice de similarité (140×140) étant beaucoup plus petite dans ce cas que celle construite pour l'approche basée sur les utilisateurs ($59.1K \times 59.1K$) ce qui entraîne une optimisation de l'espace mémoire nécessaire pour le traitement. Pour les jeux de données de films, le ratio entre le nombre d'items et le nombre d'utilisateurs étant inversé, les expériences ont été menées en utilisant l'approche basée sur les utilisateurs.

En ce qui concerne les hyper-paramètres des approches d'ajustement comparées, c'est-à-dire l'ajustement par *significance weighting* (SW, cf. équation (3.1)), l'ajustement

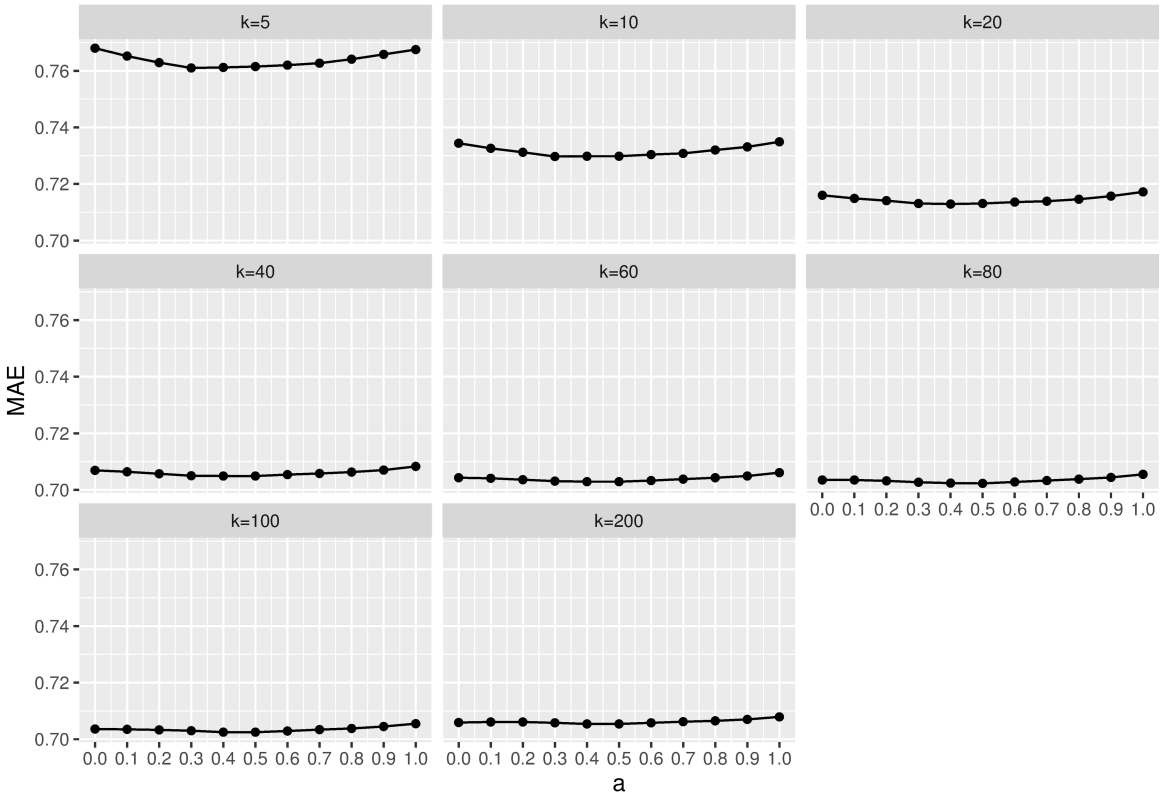


FIGURE 3.2 – Performances prédictives (MAE) de la méthode EBCR pour différentes valeurs du paramètre a (cf. équation (3.3)) et différentes tailles de voisinage (k)

.

basé sur la fonction multi-niveaux (MLCF, cf. équation (3.2)) et l'ajustement par l'approche *Laplace Smoothing* (LS, cf. équation (3.5)), nous avons suivi les recommandations fournies par les auteurs dans leurs travaux correspondants [Herlocker et al., 2002; Polatidis and Georgiadis, 2016; Russel and Norvig, 2013]. Ainsi, le seuil t utilisé dans l'approche SW a été fixé à 50 et le paramètre de lissage dans l'approche LS a été fixé à $\alpha = 1$ (ce qui est équivalent à utiliser une loi uniforme comme distribution *a priori*). Pour les paramètres de l'approche MLCF (cf. équation (3.2)), ils sont fixés comme suit : $(t_1, t_2, t_3, t_4) = (50, 20, 10, 5)$, $(x_1, x_2, x_3, x_4) = (0.5, 0.375, 0.25, 0.125)$ et $y = 0.33$, comme choisis par [Polatidis and Georgiadis, 2016] dans leur travaux.

Enfin, concernant les hyper-paramètres et la mise en œuvre des approches de filtrage collaboratif basées sur des modèles, nous avons suivi les configurations recommandées dans les travaux originaux correspondants. Pour ce faire, nous avons adopté des *packages* et des implémentations existants en *open-source*. Plus précisément, les modèles *baseline*, SVD et SVD++ ont été implémentés en utilisant le *package* Python « Surprise »[3], ce qui est aussi le cas pour toutes les approches basées sur le voisinage envisagées, y compris l'approche EBCR proposée. Ce *package* est dédié au développement et à l'évaluation des algorithmes de filtrage collaboratif dans un cadre homogène. Pour le modèle NeuMF qui

[3] http://surpriselib.com

repose sur des réseaux de neurones, nous avons utilisé une implémentation existante[4] réalisée avec le *framework PyTorch*[5], un framework couramment utilisé pour implémenter les modèles basés sur des réseaux de neurones.

Afin de garantir la reproductibilité de nos analyses, toutes les ressources sont librement accessibles (code source de la méthode, jeux de données utilisés, fichiers de résultats, etc.), via le répertoire GitHub https://github.com/lgi2p/EBCR ou le répertoire Zenodo suivant https://doi.org/10.5281/zenodo.5013115.

3.4 Résultats et discussions

Les résultats des trois types de comparaison sont présentés dans cette section.

Premièrement, les résultats concernant la généricité de l'ajustement EBCR proposé sont illustrés dans la figure 3.3. Chaque mesure de similarité présente dans la figure est associée à deux courbes de même couleur : les courbes en lignes pointillées pour les mesures originales (*e.g. MSD_KNN*, en bleu avec des rectangles) et les courbes en lignes pleines pour leurs variantes intégrant l'ajustement EBCR (*e.g. EBCR_MSD_KNN*). Comme nous pouvons le constater, les courbes en lignes pleines sont systématiquement en dessous des celles en pointillés. Par exemple, pour l'approche *COS_KNN* avec une taille de voisinage égale à 100 pour le jeu de données MovieLens-1M, l'ajustement par EBCR a conduit à une amélioration des performances prédictives de 7.83% en termes de MAE (de 0.766 à 0.706) et de 7.32% en termes de RMSE (de 0.970 à 0.899). Les résultats obtenus ont confirmé également que l'intégration de la méthode EBCR est pertinente pour améliorer la performance globale des méthodes de filtrage collaboratif testées. Enfin, ils confirment la généricité de la contribution car l'intégration de EBCR a amélioré l'*accuracy* des prédictions de *ratings* pour tous les jeux de données testés et toutes les mesures de similarité considérées.

Deuxièmement, les résultats concernant la comparaison de l'ajustement EBCR proposé avec les autres approches d'ajustement telles que SW, MLCF et LS sont illustrés par la figure 3.4 et par le tableau 3.2. Comme pour la figure 3.3, la figure 3.4 illustre les résultats obtenus en associant chaque mesure de similarité à trois courbes de même couleur : les lignes qui alternent points et pointillés désignent les mesures de similarité pondérées par le facteur de SW, les lignes en pointillés désignent les mesures pondérées par l'approche MLCF et les lignes pleines les mesures qui intègrent l'approche de pondération EBCR proposée. Le tableau 3.2 représente les résultats en termes de MAE et de RMSE des approches *Laplace Smoothing* (LS) et EBCR. Notons que ces deux approches sont basées sur le même principe bayésien : l'utilisation d'une loi *a priori* dont la mise à jour (à partir de données) permet de corriger les proportions observées. L'approche LS considère une loi uniforme (non-informative) comme *a priori* alors que l'approche EBCR utilise l'en-

[4] https://github.com/guoyang9/NCF
[5] https://pytorch.org/

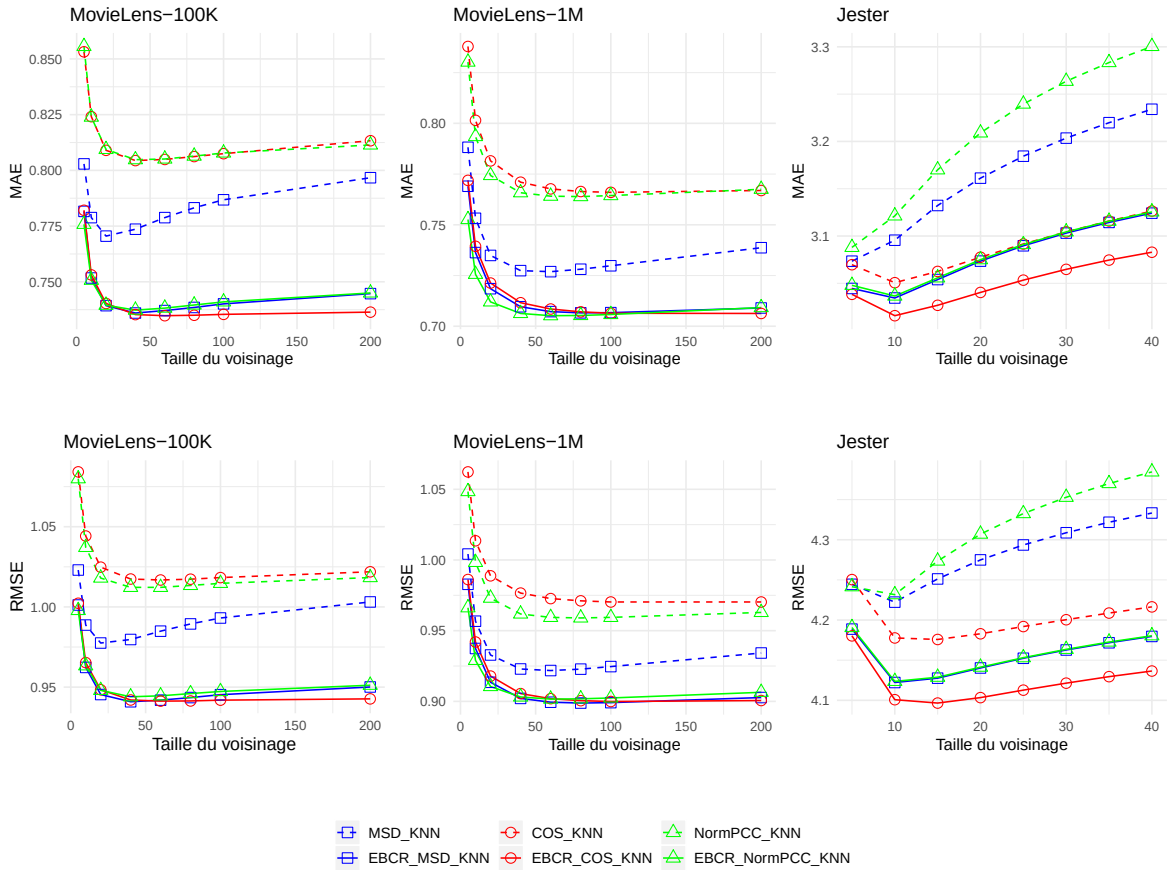


FIGURE 3.3 – Comparaison des résultats obtenus en termes de MAE et RMSE en fonction de la taille du voisinage en utilisant respectivement les mesures de similarité usuelles (courbes en lignes pointillés) et leurs variantes intégrant les ratios EBCR (courbes en lignes pleines de même couleur), sur trois jeux de données de référence : MovieLens-100K, MovieLens-1M et Jester.

semble des données de l'échantillon pour estimer ou *fitter* la distribution *a priori Beta* de façon à ce qu'elle soit au plus près de la distribution "empirique" des données. Comme des résultats similaires ont été observés pour différentes tailles de voisinage, dans le tableau 3.2, nous n'illustrons que les résultats pour 4 tailles de voisinage différentes pour les trois jeux de données testés.

D'après la figure 3.4, pour les 72 conditions testées (3 jeux de données × 3 mesures de similarités × 8 tailles du voisinage), l'ajustement EBCR a conduit à de meilleures prédictions en termes de MAE et RMSE que celles obtenues avec les approches SW et MLCF. Spécifiquement, nous observons dans la figure 3.4 que les courbes concernant l'approche EBCR (lignes pleines) se trouvent systématiquement en-dessous des courbes des deux autres approches (lignes pointillées). D'autre part, comme le montre le tableau 3.2, l'approche EBCR a conduit à des résultats systématiquement meilleurs que l'approche LS pour toutes les mesures de similarité et les tailles de voisinage considérées (même si la différence entre les résultats des deux approches est parfois minime). Ce dernier point montre que la prise en compte de la distribution des goûts dans l'ensemble de l'échan-

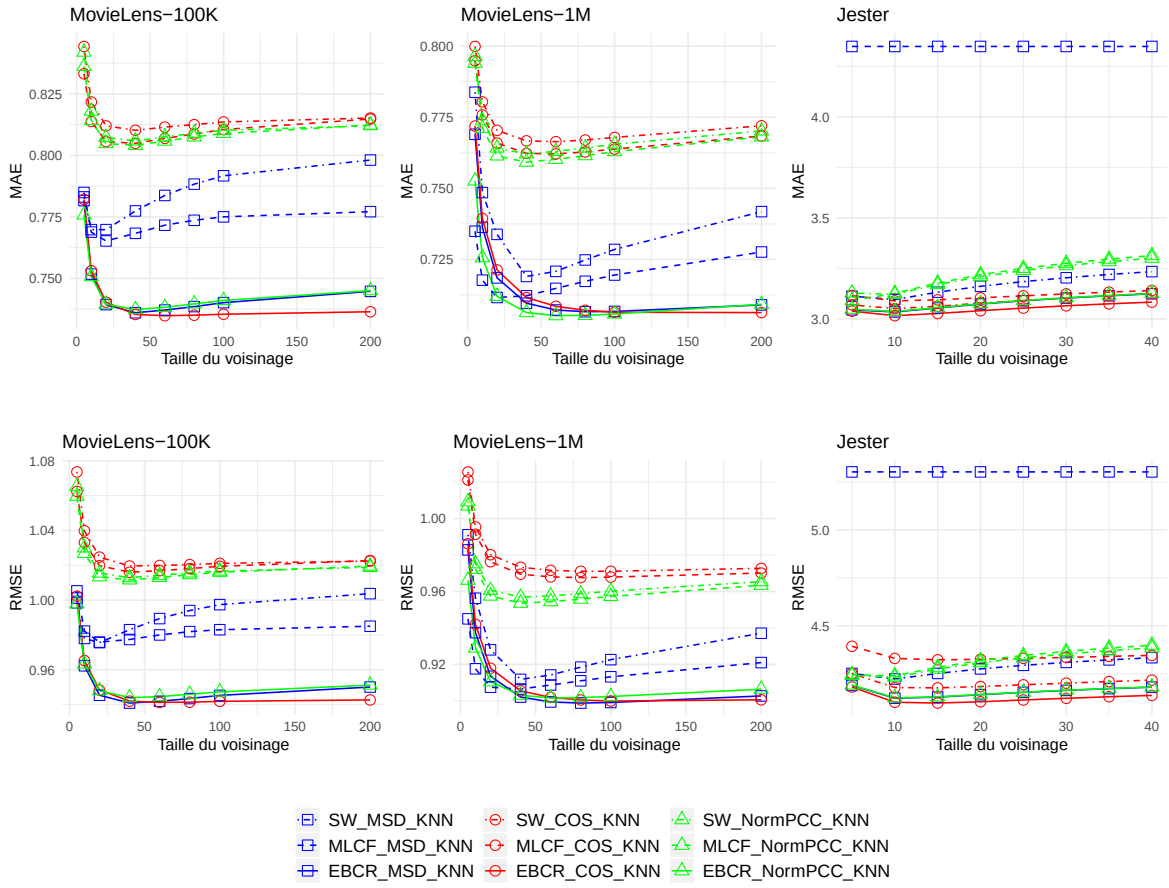


FIGURE 3.4 – Comparaison des résultats obtenus en termes de MAE et RMSE en fonction de la taille du voisinage en utilisant respectivement l'approche de *significance weighting* (SW), l'approche de *multi-level collaborative filtering* (MLCF) et l'approche EBCR sur trois jeux de données de référence : MovieLens-100K, MovieLens-1M et Jester.

tillon (comme c'est le cas pour EBCR) semble préférable à l'utilisation d'*apriori* non-informatif (comme dans l'approche LS).

Enfin, troisièmement, les résultats concernant la comparaison entre l'approche d'ajustement EBCR proposée et les approches de filtrage collaboratif basées sur les modèles, incluant les modèles *baseline*, SVD, SVD++ et NeuMF, sont récapitulés dans le tableau 3.3. Les jeux de données présentés dans ce tableau sont énumérés par ordre croissant de densités (cf. tableau 3.1). Rappelons que la densité d'un jeu de données de *ratings* correspond au pourcentage de cases complétées dans la matrice de *ratings*. Cette caractéristique des jeux de données a un impact connue sur les performances des différentes approches de système de recommandation [Adomavicius and Tuzhilin, 2005; Cacheda et al., 2011]. En termes d'*accuracy* des prédictions des *ratings* en général, les approches basées sur les modèles sont plus performantes que celles basées sur la mémoire pour des matrices de *ratings* de faible densité. Au contraire, les approches de CF basées sur le voisinage (*i.e.* les approches NBCF) ont tendance à être plus performantes sur des matrices denses. En effet, lorsque la matrice de *ratings* contient très peu d'informations, les mesures de similarité

TABLEAU 3.2 – Résultats comparatifs entre les approches *Laplace Smoothing* (LS) et EBCR sur trois jeux de données de référence (les meilleurs prédictions sont mises en gras).

Jeu de données	Mesure de similarité	Métrique d'évaluation (LS, EBCR)	Taille du voisinage			
			5	10	20	40
MovieLens-100K	MSD	MAE	(0.790, **0.782**)	(0.758, **0.752**)	(0.743, **0.739**)	(0.738, **0.736**)
		RMSE	(1.010, **1.001**)	(0.969, **0.962**)	(0.949, **0.945**)	(0.943, **0.941**)
	COS	MAE	(0.788, **0.782**)	(0.757, **0.753**)	(0.743, **0.740**)	(0.737, **0.735**)
		RMSE	(1.009, **1.002**)	(0.970, **0.965**)	(0.951, **0.948**)	(0.944, **0.942**)
	NormPCC	MAE	(0.786, **0.776**)	(0.756, **0.751**)	(0.743, **0.740**)	(0.739, **0.737**)
		RMSE	(1.008, **0.998**)	(0.969, **0.963**)	(0.951, **0.948**)	(0.945, **0.944**)
MovieLens-1M	MSD	MAE	(0.782, **0.769**)	(0.747, **0.736**)	(0.727, **0.719**)	(0.717, **0.710**)
		RMSE	(0.997, **0.983**)	(0.949, **0.937**)	(0.923, **0.914**)	(0.909, **0.902**)
	COS	MAE	(0.781, **0.772**)	(0.747, **0.739**)	(0.727, **0.721**)	(0.716, **0.712**)
		RMSE	(0.997, **0.986**)	(0.950, **0.942**)	(0.924, **0.918**)	(0.909, **0.906**)
	NormPCC	MAE	(0.775, **0.753**)	(0.742, **0.726**)	(0.724, **0.712**)	(0.714, **0.706**)
		RMSE	(0.990, **0.966**)	(0.945, **0.929**)	(0.922, **0.911**)	(0.910, **0.903**)
Jester	MSD	MAE	(3.045, 3.045)	(3.035, 3.035)	(3.074, 3.074)	(3.125, **3.124**)
		RMSE	(4.189, 4.189)	(4.123, **4.122**)	(4.141, **4.140**)	(4.180, 4.180)
	COS	MAE	(3.039, **3.038**)	(3.017, **3.016**)	(3.041, **3.040**)	(3.083, 3.083)
		RMSE	(4.181, **4.180**)	(4.101, 4.101)	(4.104, **4.103**)	(4.137, 4.137)
	NormPCC	MAE	(3.049, **3.048**)	(3.037, 3.037)	(3.075, 3.075)	(3.126, 3.126)
		RMSE	(4.191, 4.191)	(4.124, 4.124)	(4.142, **4.141**)	(4.181, **4.180**)

entre utilisateurs/items sur lesquelles les approches de NBCF reposent sont peu fiables, ce qui peut mener à des erreurs prédictives. Alors que les approches de CF basées sur les modèles gèrent mieux le faible nombre d'interactions utilisateur-item grâce à l'utilisation d'espaces latents permettant d'obtenir des prédictions plus justes. En revanche, lorsque la matrice des *ratings* est dense, elle contient plus d'informations explicites potentiellement perdues par les filtrages collaboratifs basés sur les modèles lors de la réduction de dimension de la matrice initiale. Les approches basées sur le voisinage peuvent, dans ce cas, s'avérer plus pertinentes.

TABLEAU 3.3 – Comparaison des MAE et RMSE obtenus par les approches du filtrage collaboratif basé sur les modèles et EBCR, respectivement sur 3 jeux de données (les meilleures valeurs de MAE et RMSE pour chaque jeu de données sont indiquées en gras et les secondes sont soulignées).

	Jeu de données (densité)					
Approche	MovieLens-1M (4.47%)		MovieLens-100k (6.30%)		Jester (20.53%)	
	MAE	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
baseline	0.7195	0.9088	0.7484	0.944	3.3982	4.3134
SVD	0.6863	<u>0.8743</u>	0.7376	<u>0.9358</u>	3.3713	4.5004
SVD++	**0.6729**	**0.8625**	**0.7214**	**0.9203**	3.6209	4.9042
NeuMF	<u>0.6773</u>	0.8765	0.7437	0.9363	<u>3.0375</u>	<u>4.1376</u>
EBCR	0.7052	0.9016	<u>0.7348</u>	0.9413	**3.0158**	**4.1008**

Le tableau 3.3 montre que lorsque la densité du jeu de données augmente, l'approche EBCR, approche de CF basée sur la mémoire, pourrait dépasser les approches basées sur des modèles. Pour le jeu de données MovieLens-1M (densité = 4.47%), la meilleure méthode est l'approche SVD++ (MAE = 0.6729, RMSE = 0.8625), suivie par les approches SVD

et NeuMF. Pour le jeu de données MovieLens-100K, qui est légèrement plus dense (densité=6.30%), le modèle SVD++ reste la meilleure méthode (MAE=0.7214, RMSE=0.9203), mais les résultats de l'approche EBCR arrivent en deuxième position en termes de MAE, devant les autres modèles (0.7348). Enfin, sur le jeu de données le plus dense Jester (densité=20.53%), l'approche EBCR a conduit aux meilleurs résultats en termes de prédiction de *ratings* (MAE=3.0158, RMSE=4.1008), légèrement devant le modèle NeuMF (MAE=3.0375, RMSE=4.1376) et nettement devant les modèles SVD (MAE=3.3713, RMSE=4.5004) et SVD++ (MAE=3.6209, RMSE=4.9042).

Il est intéressant de noter que les résultats présentés dans le tableau 3.3 suggèrent que les modèles de filtrage collaboratif classiques[6], par exemple SVD, SVD++ et EBCR, peuvent donner de meilleurs résultats en termes d'*accuracy* que des modèles basés sur des réseaux de neurones profonds (DNN) tels que NeuMF, bien que ce dernier soit plus récent et reconnu dans la communauté des SdR. Il est cependant possible que d'autres approches DNN soient plus performantes. De plus, comme c'est le cas pour de nombreux autres modèles basés sur les DNN évalués dans [Ferrari Dacrema et al., 2019], le modèle NeuMF a été conçu pour traiter des *ratings* d'une façon implicite et pour être évalué avec des métriques basées sur la liste des top-N recommandations, *e.g.*, HR (cf. équation (2.39)), NDCG (cf. équation (2.38)) etc. Comme le protocole d'évaluation adopté pour EBCR était basé sur des *ratings* explicites en termes de prédictions des *ratings* (*i.e.* RMSE et MAE), l'entraînement et l'optimisation du modèle NeuMF ont été adaptés en se basant sur des fonctions objectif appropriées (voir *L1Loss* et *MSELoss* dans le framework PyTorch[7]) afin de traiter la tâche de prédiction des *ratings*.

Cependant, [He et al., 2017; Xue et al., 2017] ont montré que les modèles basés sur les DNN ont clairement surpassé les modèles classiques quand l'évaluation est réalisée sur les top-N recommandations. C'est pourquoi nous avons également effectué des expérimentations qui comparent des approches de filtrage collaboratif usuelles avec celle basée sur les réseaux de neurones, dans un contexte d'évaluation de listes de top-N recommandations. Pour ce faire, nous avons modifié la fonction objectif de l'approche NeuMF pour pouvoir traiter les *ratings* implicites (*cross-entropy loss*, cf. équation (2.27)) et mettre à jour ses paramètres au moment de la rétropropagation du gradient pendant l'apprentissage. Pour les approches de filtrage collaboratif classiques, rien n'a été modifié. Nous avons ensuite adopté les métriques de HR et NDCG pour évaluer l'*accuracy* des listes de top-10 recommandations proposées par chaque modèle. Les résultats obtenus, synthétisés dans le tableau 3.4, confirment que l'approche basée sur les réseaux de neurones (NeuMF) est nettement meilleure que toutes les autres approches pour la prédiction des listes de top-N recommandations. On voit bien avec ces résultats que selon l'évaluation utilisée, les approches DNN peuvent sembler surpasser très nettement les autres approches (en termes

[6] les modèles de CF basés sur le voisinage ou sur la factorisation matricielle classique sont considérés comme des approches classiques (méthodes proposées avant 2010) tandis que les modèles de CF basés sur des réseaux de neurones sont des approches plus récentes et modernes.

[7] https://pytorch.org/docs/stable/nn.html#loss-functions

de top-N recommandations), ou au contraire faire à peine mieux, voire légèrement moins bien (en termes de prédiction des *ratings*, cf. tableau 3.3).

TABLEAU 3.4 – Comparaison entre les résultats obtenus par des approches classiques de filtrage collaboratif (*i.e. baseline*, SVD, SVD++, EBCR) et l'approche basée sur les réseaux de neurones profonds (NeuMF), dans le contexte des top-N recommandations.

Approach	MovieLens 100k		MovieLens 1M		Jester	
	HR	NDCG	HR	NDCG	HR	NDCG
baseline	0.3786	0.2061	0.4238	0.2304	0.2855	0.1667
SVD	0.3701	0.1964	0.4589	0.2618	0.2186	0.1187
SVD++	0.3849	0.2128	0.4660	0.2713	0.2352	0.1366
NeuMF	**0.811**	**0.550**	**0.863**	**0.621**	**0.832**	**0.685**
EBCR	0.2768	0.1143	0.3674	0.1710	0.2762	0.1589

3.5 Exemple d'une étude de cas

Cette section illustre un exemple réel extrait du jeu de données MovieLens-100K pour montrer la contribution de la méthode EBCR proposée. Chacune des approches comparées doit prédire la valeur de *rating* de l'utilisateur focal 903 (903 étant l'ID de l'utilisateur), pour l'item 106 (106 étant l'ID de l'item). Notons que la valeur de *rating* réelle de $r_{u_903,i_106} = 2$. Le tableau 3.5 montre les résultats obtenus. Pour les méthodes de filtrage collaboratif basées sur la mémoire, la taille du voisinage est fixée à 5 et la mesure de cosinus est utilisée pour calculer la similarité entre les utilisateurs. Comme le montre le tableau, le voisinage de l'utilisateur focal calculé par l'approche *COS_KNN* est complètement différent de celui calculé par l'approche *EBCR_COS_KNN*. Plus précisément, on remarque que pour l'approche *COS_KNN*, le nombre d'items co-notés par l'utilisateur cible et ses voisins est faible (entre 1 et 3), par exemple les voisins u_36 et u_33 ont seulement co-évalué 2 items avec l'utilisateur cible. Alors que dans le cas de l'approche *EBCR_COS_KNN*, les utilisateurs inclus dans le voisinage de l'utilisateur cible ont plus d'items co-cotés avec l'utilisateur cible (entre 4 et 18), par exemple les voisins u_556 et u_8 ont co-coté respectivement 13 et 18 items avec l'utilisateur cible.

Ces observations illustrent le fait que les utilisateurs ayant co-noté peu d'items avec l'utilisateur cible sont exclus de son voisinage lors de l'ajustement par EBCR, même s'ils ont partagé les mêmes *ratings* pour les quelques items co-notés. En ce qui concerne la valeur de *rating* prédite, le tableau 3.5 montre que la méthode *EBCR_COS_KNN* conduit à une prédiction de *rating* plus juste que la méthode *COS_KNN*, à savoir 2.306 contre 3.195 (alors que la valeur réelle de la note est de 2). En outre, dans cet exemple, les résultats montrent que l'approche EBCR permet d'obtenir la meilleure prédiction par rapport à toutes les autres méthodes comparées.

TABLEAU 3.5 – Exemple d'une étude de cas extrait du jeu de données MovieLens-100K.

Étude de cas : prédire le *rating* de u_903 (l'utilisateur focal) pour l'item i_106 (le *rating* réel est : 2)		
Approche	Voisins (# d'items co-notés avec l'utilisateur focal)	*rating* prédit
COS_KNN (k=5)	u_36 (2) ; u_33 (2) ; u_240 (3) ; u_61 (1) ; u_173 (2)	3.195
EBCR_COS_KNN (k=5)	u_556 (13) ; u_8 (18) ; u_898 (4) ; u_563 (7) ; u_609 (4)	**2.306**
baseline		3.122
SVD		2.925
SVD++		<u>2.780</u>
NeuMF		2.934

La meilleure prédiction est en gras et la seconde est soulignée.

3.6 Conclusion de chapitre

Dans ce chapitre, une nouvelle approche est proposée qui vise à affiner l'estimation de la similarité entre utilisateurs/items dans le contexte du filtrage collaboratif basé sur le voisinage. Dans un premier temps, le ratio de concordance (CR) est proposé pour représenter la concordance entre les *ratings* d'une paire d'utilisateurs. Ce ratio permet de prendre en compte la disparité potentielle entre leurs comportements de notation. Ensuite, nous proposons d'ajuster chaque ratio de concordance en tirant parti de l'inférence Bayésienne Empirique (EB), qui tient compte de la distribution de tous les ratios présents dans le jeu de données. De plus, l'ajustement réalisé par l'approche EB prend en compte le nombre d'items co-évalués par les deux utilisateurs comparés, facteur généralement ignoré par les mesures de similarité classiques. Enfin, nous proposons de pondérer les similarités classiques par le ratio de concordance ajusté (EBCR). L'évaluation sur trois jeux de données de référence a confirmé la pertinence du modèle EBCR en termes d'*accuracy* des prédictions des *ratings*.

L'approche EBCR proposée présente deux avantages : sa simplicité et sa généricité. En effet, le ratio ajusté (EBCR), utilisé comme un lissage (*smoothing*) des mesures de similarité classiques, est facile à intégrer et n'augmente pas la complexité calculatoire. De plus, la contribution de l'approche EBCR semble générique puisqu'elle permet d'améliorer la performance du filtrage collaboratif basé sur le voisinage, pour toutes les mesures de similarité et tailles de voisinage considérées.

Les résultats de la comparaison entre la méthode EBCR proposée et les approches de CF basées sur des modèles ont également mis en évidence la pertinence de l'ajustement par EBCR, notamment pour les jeux de données denses. Par exemple, l'approche proposée est plus performante que le modèle basé sur les réseaux de neurones profonds pour le jeu de données Jester pour la prédiction des *ratings*. Néanmoins, comme indiqué à la fin de la section 3.4, le choix du modèle optimal de filtrage collaboratif est largement dépendant de la forme d'évaluation considérée (l'*accuracy* selon l'ensemble des *ratings* prédits ou la pertinence des listes de top-N recommandations). Plus précisément, lorsqu'il s'agit de prédiction de *ratings* (évaluation par item), les modèles de CF classiques semblent plus appropriés que les modèles basés sur les réseaux de neurones profonds. Ce type de mo-

dèle apparaît cependant être le meilleur choix dans un contexte d'évaluation des top-N recommandations (évaluation par liste d'items).

Notons que dans certaines applications réelles, les prédictions des *ratings* sur des items individuels pourraient être plus pertinentes, comme par exemple les recommandations pour un groupe d'utilisateurs [Boratto et al., 2010] où il est difficile de fournir des top-N meilleures recommandations en traitant le groupe d'utilisateurs comme un ensemble.

Une autre application pourrait être d'afficher aux utilisateurs les *ratings* prédits pour chaque item, et ainsi les aider ainsi à prendre des décisions [Orso et al., 2017; Wexelblat and Maes, 1999]. De plus, les recommandations générées par les NBCF sont plus faciles à expliquer que les modèles boîtes-noires basés sur des facteurs latents. Dans un tel contexte, l'explication des recommandations issues des approches NBCF pourrait être « les utilisateurs qui ont acheté l'item i comme vous, ont également acheté l'item j ». Néanmoins, les approches de NBCF basées sur les mesures de similarité classiques ne permettent pas de distinguer si les recommandations sont basées ou non sur des avis de voisins fiables. A ce niveau, la méthode EBCR proposée pourrait permettre de renforcer la confiance des utilisateurs dans leurs recommandations, car les items recommandés sont basés sur des voisins plus fiables car ayant co-évalué un nombre important d'items, ce qui pourrait augmenter la confiance des utilisateurs en leurs recommandations et donc au SdR.

Chapitre 4

Recommandation plus diversifiée ? Vers une meilleure compréhension de la diversité dans les approches de recommandation

Sommaire

4.1 Motivation 91

4.1.1 Problématique et questions de recherche 93

4.1.2 Solutions proposées 93

4.2 Diversité des recommandations 94

4.2.1 Définition et métrique d'évaluation 94

4.2.1.1 Diversité individuelle (*absolue*) & diversité agrégée 95

4.2.1.2 Diversité personnalisée (*relative*) 96

4.2.2 Différentes fonctions objectifs pour la diversification 98

4.2.2.1 Fonction objectif classique (non-personnalisée) pour la diversité 98

4.2.2.2 Fonction objectif pour la diversité personnalisée 99

4.2.3 Heuristique gloutonne d'optimisation 100

4.2.4 Autres approches de diversification 101

4.3 Protocole d'évaluation 102

4.3.1 Les systèmes de recommandation considérés 102

4.3.1.1 Approche basée sur la popularité des items (*TopPopular*) . 102

4.3.1.2 Approche basée sur le contenu et les données liées (*CBF*) . 102

4.3.1.3 Approche par filtrage collaboratif basé sur les items (*IBCF*) 103

4.3.1.4 Approche dite *singular value decomposition* (*SVD*) 103
4.3.1.5 Approche basée sur les réseaux de neurones profonds (*DNN*) 104
4.3.1.6 Approche hybride basée sur les *embeddings* de graphe de connaissances (*KGE*) 104
4.3.1.7 Approche hybride combinant la popularité et le contenu des items (*CBF-TopPopular*) 105
4.3.2 Jeux de données & pré-traitement 105
4.3.3 Construction des graphes de connaissances 107
4.3.4 Construction des modèles de recommandation 107
4.3.5 Métriques d'évaluation 111
4.3.5.1 Métriques d'*accuracy* 111
4.3.5.2 Métriques de diversité 111
4.3.5.3 Similarité sémantique 112
4.3.6 Configuration choisie pour l'étude de la diversification 113
4.4 Résultats et discussions 113
4.4.1 Performances en termes d'*accuracy* et de diversité des SdR testés . 114
4.4.2 Impact du post-traitement de la diversification 117
4.4.3 Impact du nombre d'items candidats du post-traitement glouton sur la diversité et l'*accuracy* 120
4.4.4 Limitation, discussion et implication 122
4.5 Conclusion de chapitre 124

4.1 Motivation

Les systèmes de recommandation (SdR) sont des outils efficaces pour aider les utilisateurs à trouver ce qui leur correspond dans le contexte actuel de surcharge d'informations. Le chapitre 2 a introduit différentes familles de SdR telles que les approches basées sur la matrice de *ratings* (NBCF, SVD, SVD++, réseaux de neurones, etc.), les approches basées sur le contenu des items, les approches tirant parti de l'ingénierie des connaissances (*e.g.* Linked Data, KGE) et les approches hybrides. Bien que les algorithmes et les stratégies derrière ces SdR soient différents, ils partagent un même objectif : mieux prédire les préférences des utilisateurs pour des items qu'ils ne connaissent pas encore, c'est-à-dire avoir la meilleure performance en termes d'*accuracy* des recommandations.

L'évaluation des performances de ces approches de recommandation est généralement focalisée sur leur capacité à prédire, le plus précisément possible, les interactions

entre les utilisateurs et les items. Les systèmes de recommandation les plus récents, *e.g.* les modèles basés sur les *embeddings* de graphe ou sur les réseaux de neurones, sont capables d'améliorer largement l'*accuracy* des recommandations par rapport aux approches plus *classiques, e.g.* les modèles de type filtrage collaboratif ou basés sur le contenu. Cependant, les SdR les plus précis, selon les mesures d'*accuracy* standards, ne sont pas nécessairement les plus utiles pour les utilisateurs [McNee et al., 2006]. Par exemple, imaginons un utilisateur qui aime regarder des films dans lesquels Brad Pitt est acteur. Si un SdR lui fournit une liste de films avec Brad Pitt, tous les films recommandés peuvent être précis (les recommandations correspondent aux goûts de cet utilisateur) mais la liste entière a une utilité limitée car l'utilisateur a probablement déjà vu la plupart d'entre-eux. Il s'agit d'un problème typique de manque de diversité dans les recommandations, car les items de la liste sont très similaires les uns avec les autres (*i.e.*, les items partagent beaucoup d'attributs).

La diversité des recommandations est depuis quelques années reconnue comme un critère important pour évaluer les listes de recommandations [McNee et al., 2006; Vargas and Castells, 2011; Kunaver and Požrl, 2017; Meymandpour and Davis, 2020]. L'évaluation de l'*accuracy* des SdR a été étudiée de manière rigoureuse, et les différentes approches de recommandation ont été comparées selon ce critère. En revanche, peu d'études ont comparé et analysé leurs performances en termes de diversité.

La plupart des méthodes proposées visant à améliorer la diversité des recommandations considèrent l'optimisation de la diversité comme une étape de post-traitement, indépendante du système qui génère les recommandations [Ziegler et al., 2005; Adomavicius and Kwon, 2012; Aytekin and Karakaya, 2014; Sha et al., 2016; Kunaver and Požrl, 2017]. Dans ces approches de diversification des recommandations, le SdR est d'abord utilisé pour obtenir un large ensemble d'items candidats (par exemple, les 100 meilleurs items en termes d'*accuracy* selon les prédictions du SdR) ; cette liste est ensuite affinée par le post-traitement de diversification pour construire une liste de recommandations plus concise et diversifiée, qui est proposée à l'utilisateur. Comme les SdR utilisent des approches et des algorithmes de prédiction différents, ils peuvent renvoyer des listes d'items candidats qui représentent différents niveaux de diversité. Par conséquent, appliquer la même stratégie de post-traitement de diversification peut ne pas conduire au même niveau de diversification selon le SdR utilisé.

De manière intuitive : 1) les SdR basés sur le contenu (CBF) ont la plus faible capacité à recommander des items diversifiés, car ils favorisent les items dont le contenu est similaire à ceux aimés par l'utilisateur cible ; 2) les SdR basés sur le filtrage collaboratif (CF) peuvent conduire à des listes d'items plus diversifiées, car ils explorent les profils des autres utilisateurs et 3) comme les approches hybrides prennent en compte à la fois les informations collaboratives et les contenus, elles pourraient conduire à des recommandations avec des niveaux de diversité intermédiaires. Augmenter aveuglément la diversité des recommandations (par un post-traitement) semble donc être, pour le moins, discu-

table. Par exemple, il pourrait être raisonnable d'augmenter la diversité des recommandations pour les CBF, mais cela pourrait être risqué pour les CF dont les recommandations pourraient déjà être suffisamment diversifiées.

Les travaux existants, tels que [Bradley and Smyth, 2001; Ziegler et al., 2005; Adomavicius and Kwon, 2012; Aytekin and Karakaya, 2014; Wu et al., 2018; Yigit-Sert et al., 2020], ont principalement tenté d'augmenter la diversité de la liste recommandée tout en maintenant l'*accuracy* des recommandations de la liste. Cependant, cela pourrait s'avérer inutile ou même contre-productif pour les SdR basés sur le filtrage collaboratif (CF). De plus, le niveau optimal de diversité pourrait varier d'un utilisateur à l'autre [Di Noia et al., 2014; Wu et al., 2018; Meymandpour and Davis, 2020]. Par exemple, une liste de recommandations très diversifiée peut plaire aux utilisateurs ayant des goûts relativement éclectiques, mais elle peut diminuer la satisfaction d'autres utilisateurs plus réfractaires au changement. Il semble donc raisonnable de comparer et d'analyser les comportements de ces SdR à la fois en termes de diversité *absolue* et de diversité *relative*. Pour un utilisateur donné, la diversité *absolue* fait référence à la quantité de diversité présente dans sa liste de recommandations, tandis que la diversité *relative* mesure l'adéquation entre la diversité *absolue* de la liste recommandée et les attentes de l'utilisateur en termes de diversité.

4.1.1 Problématique et questions de recherche

Les principales questions de recherche (QR) abordées dans ce chapitre sont les suivantes :

- QR1 : Comment les principales familles de systèmes de recommandation se comportent-elles en termes de diversité *absolue* et *relative* ?
- QR2 : L'optimisation de la diversité induite par la méthode de post-traitement est-elle aussi efficace pour tous les types de systèmes de recommandation ? Autrement dit, quelles seraient les performances de ces systèmes de recommandation, en termes de diversité et d'*accuracy*, s'ils étaient combinés avec le même post-traitement de diversification ?
- QR3 : Quel est l'impact de la prise en compte des besoins de diversité des utilisateurs individuels lors du post-traitement optimisant la diversification pour différents types de SdR ?

4.1.2 Solutions proposées

Pour mener à bien cette étude et répondre aux questions de recherche précédentes, nous proposons d'analyser et de comparer la capacité des principales familles de SdR à fournir des recommandations diversifiées. À cette fin, nous comparons sept modèles (systèmes) de recommandation différents qui représentent quatre types différents de SdR.

Nous considérons trois jeux de données liés à trois domaines différents de recommandation : le cinéma, la littérature et les *animés* (dessins animés japonais inspirés des mangas).

Les 7 SdR étudiés dans ce chapitre sont les suivants : une approche non-personnalisée basée sur la popularité des items [Ferrari Dacrema et al., 2019] ; pour les SdR basés sur le contenu (CBF), nous considérons une approche basée sur les données liées (*Linked Data*) [Di Noia et al., 2012b] ; pour les SdR basés sur le filtrage collaboratif (CF), nous considérons une approche basée sur les plus proches voisins [Sarwar et al., 2001], une approche de CF basée sur les modèles [Koren et al., 2009] et une approche de CF basée sur les réseaux de neurones profonds [He et al., 2017] ; enfin, pour les SdR hybrides, nous considérons une approche basée sur les *embeddings* de graphes de connaissances [Palumbo et al., 2018a] et une approche qui combine les modèles basés sur le contenu et la popularité des items. Les principes de ces SdR seront brièvement introduits dans la présentation du protocole d'évaluation de notre étude en section 4.3.1.

Nous proposons d'analyser les résultats obtenus en combinant chacun de ces SdR avec le même post-traitement de diversification. Les recommandations résultantes sont évaluées selon leur *accuracy* et leurs diversités *absolue* et *relative*. Le post-traitement de diversification peut facilement être adapté pour optimiser la diversité *relative* plutôt que la diversité *absolue*. Nous décrivons cette extension et évaluons l'impact de l'ajustement de la diversité des recommandations par rapport aux besoins individuels des utilisateurs.

Le reste de ce chapitre est organisé comme suit : La section 4.2 introduit les notions de diversité relatives aux SdR, présente les objectifs du post-traitement de diversification et détaille les méthodes de diversification comparées dans cette étude. Dans la section 4.3, nous détaillons le protocole d'évaluation. Nous rappelons les principes des SdR comparés dans notre étude ainsi que les métriques d'évaluation utilisées. Dans la section 4.4, nous présentons et commentons les résultats expérimentaux et discutons des limites et des implications de cette étude. Ce travail a fait l'objet d'un article publié dans la revue *Information Processing & Management* (IPM) [Du et al., 2021a].

4.2 Diversité des recommandations

4.2.1 Définition et métrique d'évaluation

Selon le dictionnaire Larousse, le concept de diversité est défini comme : « *Ensemble des personnes qui diffèrent les unes des autres par leur origine géographique, socio-culturelle ou religieuse, leur âge, leur sexe, leur orientation sexuelle, etc.* ». De manière plus générale, la *diversité* peut être définie comme *l'ensemble des choses qui diffèrent les unes des autres par leur caractéristiques* et ainsi considérée comme l'inverse de la *similarité* [Bradley and Smyth, 2001]. Notons que la notion de similarité est généralement relative à une paire d'éléments alors que la diversité est généralement considérée pour une liste d'éléments. Dans le contexte des systèmes de recommandation, les chercheurs adoptent sou-

vent la métrique de diversité intra-liste – ou *Intra-List-Diversity* (ILD) pour mesurer la diversité d'une liste de recommandations, qui est définie comme la distance (dis-similarité) moyenne par paires d'items recommandés (voir équation (4.1))[Bradley and Smyth, 2001; Vargas and Castells, 2011; Castagnos et al., 2013; Di Noia et al., 2014; Ekstrand et al., 2014; Jannach et al., 2015; Jugovac et al., 2017; Wang et al., 2019]. Étant donné la liste d'items recommandés L, l'ILD de L est définie formellement comme suit :

$$\mathrm{ILD}(\mathrm{L}) = \frac{2}{|\mathrm{L}| \times (|\mathrm{L}| - 1)} \sum_{\forall 1 \leq j < k \leq |\mathrm{L}|} (1 - sim(\mathrm{L}[j], \mathrm{L}[k])) \tag{4.1}$$

où $sim(.,.)$ est une fonction qui mesure la similarité (conventionnellement normalisée entre [0, 1]) pour une paire d'items donnée. Par exemple, cette mesure de similarité peut se baser sur les proximités entre items, estimées en fonction de leur contenu (caractéristiques) ou en fonction de la matrice des *ratings* des utilisateurs.

4.2.1.1 Diversité individuelle (*absolue*) & diversité agrégée

La diversité d'un SdR peut être mesurée au niveau individuel ou au niveau du système (dite *diversité agrégée*). Dans le premier cas, la diversité individuelle mesure la diversité de la liste de recommandations proposée à l'individu (l'utilisateur) en question. La mesure de l'ILD présentée dans la sous-section précédente fait généralement référence à la diversité individuelle, car la liste d'items sur laquelle l'ILD est mesurée est fournie pour un individu/utilisateur particulier. Alors que l'ILD individuel mesure la diversité d'une recommandation particulière, la diversité agrégée mesure la capacité globale d'un SdR à recommander des items variés [Adomavicius and Kwon, 2012]. La diversité agrégée peut être calculée comme la proportion des items du catalogue qui peuvent être recommandés aux utilisateurs du SdR. Un SdR ayant des performances élevées en termes de diversité agrégée devrait par exemple être capable de fournir aux utilisateurs des items de niche (ceux moins populaires). Ces deux mesures sont complémentaires. En effet, considérons une liste de 10 items très diversifiée. Un SdR qui proposera cette même liste à tous les utilisateurs aura une diversité individuelle forte, mais une ILD agrégée très faible. Inversement, un SdR renvoyant à chaque utilisateur une liste d'items différente pourra avoir une ILD agrégée forte, même si chacune des listes recommandées a une diversité individuelle faible.

Ce chapitre vise à analyser et comparer les performances de différentes familles de systèmes de recommandation, en termes de diversité individuelle (ILD). Comme nous envisageons d'étudier l'impact de la prise en compte des besoins de diversité des utilisateurs individuels au cours du post-traitement de la diversification (*i.e.* QR3 de la section 4.1.1), la diversité agrégée ne correspond pas à l'objectif de la présente étude. Néanmoins, la comparaison de la diversité agrégée entre divers types d'approches de recommandation aurait également ses propres intérêts, en particulier pour le commerce électronique,

car la popularité des items incite généralement les utilisateurs à choisir des items populaires plutôt que ceux de niche. Par exemple, [Lee and Hosanagar, 2014] ont démontré que les SdR basés sur le CF ont généralement une faible diversité agrégée.

Nous employons le terme de diversité *absolue* pour faire référence à l'ILD de la liste d'items recommandée à un utilisateur particulier, par opposition à la diversité *relative* abordée dans la sous-section suivante.

4.2.1.2 Diversité personnalisée (*relative*)

La diversité personnalisée des recommandations a récemment attiré l'attention des chercheurs de la communauté des SdR. L'hypothèse principale est que différents utilisateurs peuvent avoir des besoins différents en termes de diversification des recommandations. Contrairement à la diversité *absolue* (non-personnalisée) qui mesure la quantité de diversité (ILD) dans une liste d'items recommandés, la diversité *relative* (personnalisée) mesure la correspondance entre la diversité *absolue* et les besoins de diversité de l'utilisateur cible. Comme la diversité *relative* dépend du profil de chaque utilisateur, il faut dans un premier temps quantifier leur besoin en diversité.

La façon la plus intuitive de quantifier les besoins en diversité d'un utilisateur est de calculer la valeur de l'ILD de la liste des items qu'il a évalués, c'est-à-dire les items de son profil [Jugovac et al., 2017; Meymandpour and Davis, 2020]. De manière alternative, on peut aussi quantifier, sur ce même profil, les besoins en diversité de l'utilisateur en mesurant l'entropie de *Shannon*, par rapport à certaines catégories d'items. Par exemple, pour la recommandation de films, les auteurs de [Di Noia et al., 2014] proposent d'étudier la propension des utilisateurs individuels à solliciter de la diversité en calculant, à partir de leur profil, l'entropie basée sur différents attributs : le genre, les acteurs, l'année de sortie, etc. Ces approches reposent principalement sur les *ratings* passés des utilisateurs, c'est-à-dire leurs évaluations des items. Les auteurs de [Wu et al., 2018] supposent que la personnalité des utilisateurs peut influencer leurs besoins en termes de diversité. Ils proposent de quantifier les besoins de diversité des utilisateurs à l'aide d'un modèle de personnalité. Par exemple, on peut s'attendre à ce qu'un utilisateur ayant une grande ouverture d'esprit (*Openness*) aspire à une diversité élevée. Ils ont donc mené une étude auprès des utilisateurs pour mieux cerner leur personnalité. Ensuite ils ont construit un modèle de régression linéaire pour analyser l'impact de la personnalité d'un utilisateur sur ses besoins en matière de diversité (diversité désirée).

Disposant de la diversité estimée sur les items du profil utilisateur $\mathrm{D}iv(p_u)$ (diversité *désirée*) et de celle calculée sur ses items recommandés $\mathrm{D}iv(r_u)$, la diversité *relative* peut être définie en comparant $\mathrm{D}iv(p_u)$ et $\mathrm{D}iv(r_u)$ grâce à des métriques classiquement utilisées pour quantifier une erreur d'estimation. Les travaux de [Jugovac et al., 2017] et de [Wu et al., 2018] proposent de calculer, pour chaque utilisateur u, l'erreur absolue entre la diversité du profil de u et celle de la liste de recommandations de u, *i.e.* $|\mathrm{D}iv(p_u) - \mathrm{D}iv(r_u)|$.

De même, [Meymandpour and Davis, 2020] utilise l'erreur quadratique moyenne de diversité (RMSDE) pour représenter la diversité *relative* (cf. équation (4.2)).

$$\text{RMSDE} = \sqrt{\frac{\sum_{u \in \text{U}} (\text{D}iv(p_u) - \text{D}iv(r_u))^2}{|\text{U}|}} \tag{4.2}$$

En statistique, le coefficient de détermination (R^2) est une métrique courante pour mesurer la corrélation de deux ensembles de valeurs. Dans le contexte de la diversité *relative*, on peut utiliser cette métrique (cf. équation (4.3)) pour mesurer, sur l'ensemble des utilisateurs, à quel point les valeurs de $\text{D}iv(r_u)$ (*i.e.* diversités sur les items recommandés) sont adaptées aux valeurs de $\text{D}iv(p_u)$ (*i.e.* besoins de diversité des utilisateurs estimés sur leur profil). Le terme $\overline{\text{D}iv(p_u)}$ fait référence à la diversité moyenne des profils de l'ensemble des utilisateurs U.

$$\text{R}^2 = 1 - \frac{\sum_{u \in \text{U}} (\text{D}iv(p_u) - \text{D}iv(r_u))^2}{\sum_{u \in \text{U}} (\text{D}iv(p_u) - \overline{\text{D}iv(p_u)})^2} \tag{4.3}$$

Le R^2 est une version normalisée de la métrique RMSDE, dans laquelle les valeurs varient de 0 à 1. Une telle normalisation peut être utile en termes d'interprétation. Ainsi, une valeur de $\text{R}^2 = 1$ signifie que les diversités proposées par le système de recommandation répondent parfaitement aux besoins des utilisateurs en termes de diversité.

D'autres façons de mesurer la diversité *relative* considèrent des métriques basées sur des distributions de probabilité, telles que la divergence de *Kullback-Leibler* (KL). Dans ce type d'approche, les items sont classés dans différentes catégories et l'on cherche à savoir si les items du profil d'un utilisateur et ceux qui lui sont recommandés se distribuent de la même manière au sein des différentes catégories. Par exemple, [Steck, 2018] propose la métrique de *calibration* $\text{C}_{\text{KL}}(p, q)$, basée sur la divergence de KL de deux distributions de probabilité p et q, p étant, dans son étude, la distribution des genres dans les films du profil utilisateur et q étant la distribution des genres dans les films recommandés (cf. équation (4.4)).

$$\text{C}_{\text{KL}}(p, q) = \sum_{g \in \text{genres}} p(g|u) \log \frac{p(g|u)}{q(g|u)} \tag{4.4}$$

Des valeurs faibles de $\text{C}_{\text{KL}}(p, q)$ correspondent à des recommandations bien calibrées. Étant donné que $\text{C}_{\text{KL}}(p, q)$ diverge dès que $q(g|u) = 0$ et $p(g|u) > 0$, (c'est-à-dire dès que le genre g est présent dans le profil de u mais pas dans les recommandations qui lui sont faites), Steck propose de remplacer le terme $q(g|u)$ par $(1 - a) \cdot q(g|u) + a \cdot p(g|u)$ dans l'équation (4.4), avec une petite valeur de $a > 0$ (par exemple 0.01).

Le travail de [Steck, 2018] vise notamment à fournir à l'utilisateur une liste de films dans laquelle la proportion de chaque genre correspond au mieux à celle de son profil. Bien que liées, la notion de recommandations *calibrées* est, néanmoins, assez différente

de la notion de diversité *relative*. D'un côté les recommandations *calibrées* peuvent sembler plus riches puisque, par exemple dans le cas des genres, elles ne cherchent pas uniquement à renvoyer une liste contenant différents genres, mais à s'assurer du fait que les différents genres proposés sont prioritairement ceux appréciés par l'utilisateur. D'un autre côté, les recommandations calibrées se limitent à considérer un attribut particulier tel que le *genre* d'un film, alors que pour la mesure de diversité *relative* l'ensemble des attributs peut être considéré. Le fait de se limiter à une seule propriété est en général très réducteur puisque les items (films) disposent en général de nombreuses caractéristiques (par exemple, les réalisateurs, les acteurs, etc.).

4.2.2 Différentes fonctions objectifs pour la diversification

Cette section aborde les principales fonctions objectifs associées à la diversification.

4.2.2.1 Fonction objectif classique (non-personnalisée) pour la diversité

En général, l'augmentation de la diversité est associée à une diminution de l'*accuracy* des recommandations [Zhou et al., 2010; Adomavicius and Kwon, 2012; Wang et al., 2019]. L'équilibre entre *accuracy* et diversité (*i.e.* l'identification du meilleur compromis) représente un réel défi pour les systèmes de recommandation. D'une part, le fait de toujours fournir aux utilisateurs des recommandations en se focalisant sur l'*accuracy*, quelle que soit la diversité, peut les lasser et donc réduire leur satisfaction. Par exemple, une liste ne contenant que des films d'un seul genre peut sembler répétitive même pour un utilisateur qui apprécie ce type de film. Ce problème est connu sous le nom de sur-spécialisation, et il est notamment fréquemment observé pour les SdR basés sur le contenu, qui fournissent aux utilisateurs des items similaires à leurs items les mieux notés [Lops et al., 2011]. D'autre part, la prise en compte excessive de la diversité de la liste des items recommandés risquerait de sacrifier la pertinence, ce qui n'est pas non plus acceptable. Par exemple, recommander, au profit de cette diversité, des films d'horreur et d'action à un utilisateur qui n'apprécie que des films comiques pourrait diminuer sa satisfaction et remettre en cause son utilisation ultérieure du système. Il faut donc tenir compte à la fois de la pertinence (*accuracy*) et de la diversité. Néanmoins, l'*accuracy* doit rester l'objectif premier des systèmes de recommandation, car une perte significative de pertinence ne serait pas acceptable dans la plupart des applications réelles [Adomavicius and Kwon, 2012]. En d'autres termes, on vise à recommander des items permettant un gain considérable en diversité tout en conservant un excellent niveau d'*accuracy*.

Cette problématique de compromis entre diversité et *accuracy* des recommandations peut être traitée comme un problème d'optimisation bi-critères pondérés, dans lequel on cherche à maximiser la pertinence globale d'une liste de recommandations tout en minimisant la redondance entre les items de cette liste. Une fonction objectif largement utilisée pour résoudre cette problématique est fournie par l'équation (4.5) [Bradley and

Smyth, 2001] :

$$f_{obj}(\mathrm{L},\alpha) = (1-\alpha) \times \frac{\sum_{\forall i \in \mathrm{L}} rel(i)}{|\mathrm{L}|} + \alpha \times \mathrm{ILD}(\mathrm{L}) \tag{4.5}$$

où $rel(i)$ représente le score de pertinence (*relevance* en anglais) de l'item i et le paramètre $\alpha \in [0,1]$ est un facteur de diversification équilibrant le compromis entre l'*accuracy* (estimée via la pertinence moyenne des items de L) et la diversité de L (mesurée via son ILD).

4.2.2.2 Fonction objectif pour la diversité personnalisée

Nous pouvons constater que cette fonction objectif (équation (4.5)) est indépendante des utilisateurs puisqu'elle essaye de maximiser la diversité (ILD) de la liste L quel que soit l'utilisateur ciblé. Elle ne tient donc pas compte des besoins individuels en termes de diversité. Comme indiqué dans la section 4.2.1.2, certaines métriques prenant en compte les besoins individuels des utilisateurs en termes de diversification ont été proposées. Ainsi, on peut envisager des variantes de l'équation (4.5) remplaçant l'ILD par une métrique évaluant l'adéquation entre la diversité souhaitée par un utilisateur et celle de la liste L qui lui est proposée.

Généralement considérée comme un problème d'optimisation bi-critères concernant à la fois l'*accuracy* et la diversité, la personnalisation de la diversité s'appuie principalement sur la partie droite de l'équation (4.5) car la partie gauche (partie *pertinence*), qui dépend uniquement des SdR en soit, est indépendante des utilisateurs considérés.

Différentes approches ont été proposées pour traiter la partie droite de la fonction objectif classique (équation (4.5)). Les auteurs de [Di Noia et al., 2014] proposent de pondérer les valeurs de similarité entre paires d'items en fonction du besoin de diversité d'un utilisateur donné (qui est mesuré par l'entropie de Shannon). Ainsi, pour un utilisateur ayant une valeur d'entropie élevée (c'est à dire ayant un besoin de diversité élevé), leur fonction objectif pénalise les scores des items candidats qui sont similaires à ceux déjà présents dans la liste recommandée. Les auteurs de [Wu et al., 2018] modélisent les personnalités des utilisateurs pour mesurer leurs besoins de diversification $\mathrm{D}iv(p_u)$. La partie droite de la fonction objectif dans leur approche est définie par $-|\mathrm{D}iv(p_u) - \mathrm{D}iv(r_u)|$, c'est-à-dire la différence absolue entre la quantité de diversité du profil de l'utilisateur et la diversité de la liste recommandée. Notez que la négation (signe $-$) de cette différence est prise en compte puisque le but est de maximiser le score global de la fonction objectif. De même, l'approche de recommandation calibrée proposée par [Steck, 2018] adopte la divergence de *Kullback-Leibler* pour mesurer la distance entre deux distributions (voir équation (4.4)). Par conséquent, dans la partie droite de leur fonction objectif l'ILD est remplacée par la divergence C_{KL}, de manière à recommander des items dont la diversité correspond au mieux au profil utilisateur.

Bien qu'elles soient différentes, toutes les fonctions objectifs discutées précédemment

ont un objectif commun : minimiser la différence entre la diversité mesurée dans les profils utilisateurs et celle des items recommandés. Dans la présente étude, comme souligné par QR3 (cf. section 4.1.1), nous visons également à étudier l'impact de la personnalisation de la diversité sur les performances de différentes familles de SdR. Une façon simple et intuitive d'atteindre cet objectif est d'optimiser la variante de la fonction objectif classique (équation (4.5)) donnée dans l'équation (4.6) :

$$f_{obj}(u, \mathrm{L}, \alpha) = (1-\alpha) \times \frac{\sum\limits_{\forall i \in \mathrm{L}} rel(i)}{|\mathrm{L}|} + \alpha \times (-|\mathrm{ILD}(\mathrm{L}) - \mathrm{ILD}_p(u)|) \tag{4.6}$$

où $\mathrm{ILD}_p(u)$ représente l'ILD des items notés par l'utilisateur u (le profil de u). L'idée ici est de recommander une liste L dans laquelle les items seraient pertinents et l'ILD de L serait aussi proche que possible de l'ILD du profil de l'utilisateur, plutôt que d'être aussi élevé que possible.

4.2.3 Heuristique gloutonne d'optimisation

L'optimisation gloutonne est une approche de post-traitement couramment utilisée, dans laquelle les items candidats préalablement ordonnés par leurs scores de pertinence sont ultérieurement reclassés [Bradley and Smyth, 2001]. Plus précisément, l'approche gloutonne procède en n étapes séquentielles, n étant la longueur de la liste de recommandations à construire. A chaque itération, elle ajoute un nouvel item à cette liste de recommandations. Pour cela, elle recherche parmi les items candidats celui qui, une fois ajouté aux items déjà sélectionnés lors des étapes précédentes, maximisera la fonction objectif bi-critères considérée pour combiner les scores d'*accuracy* et de diversité. Formellement, considérons $\mathrm{L}(u)$ comme la liste des items non-évalués pour l'utilisateur u, dans laquelle les items sont ordonnés par leurs scores de pertinence. Supposons que nous voulons recommander une liste de n items à l'utilisateur u, notée $\mathrm{L}_{rec}(u)$. Sans tenir compte de l'optimisation de la diversité, les n premiers items de $\mathrm{L}(u)$ seraient renvoyés à u, ce qui correspondrait à un scénario typique de système de recommandation. Au contraire, l'optimisation gloutonne illustrée par l'algorithme 1 sélectionne à chaque étape l'item maximisant la fonction objectif f_{obj} (lignes 4 et 6), qui prend en compte à la fois la diversité et l'*accuracy*. Notez que différentes fonctions objectifs peuvent être utilisées en fonction de la valeur du paramètre booléen *personnalisé*.

Comme le soulignent les auteurs de [Bradley and Smyth, 2001], cet algorithme est coûteux en termes de complexité temporelle car la taille de $\mathrm{L}(u)$ peut être très grande (*e.g.* $|\mathrm{L}(u)| \gg n$). Ainsi, les auteurs ont proposé une approche *bornée*, qui consiste à appliquer simplement l'algorithme aux premiers m éléments de $\mathrm{L}(u)$ (avec $|\mathrm{L}(u)| \gg m > n$). Le temps de calcul est ainsi largement réduit sans que cela n'ait un grand impact sur le score de la fonction objectif de la liste recommandée, vu que les éléments ignorés ont un score de pertinence faible et donc peut de chances d'être retenus de toutes façons. Cette version

Algorithme 1 : Optimisation de diversité par heuristique gloutonne

Entrée : $\mathrm{L}(u)$, α, *personnalisé* (booléen), n
Sortie : $\mathrm{L}_{rec}(u)$

$\mathrm{L}_{rec}(u) \leftarrow \emptyset$;
tant que $|\mathrm{L}_{rec}(u)| < n$ **faire**
  si *personnalisé* **alors**
    $i^* \leftarrow \underset{i \in \mathrm{L}(u) \setminus \mathrm{L}_{rec}(u)}{\operatorname{argmax}} f_{obj}(u, \mathrm{L}_{rec}(u) \cup \{i\}, \alpha)$;
  sinon
    $i^* \leftarrow \underset{i \in \mathrm{L}(u) \setminus \mathrm{L}_{rec}(u)}{\operatorname{argmax}} f_{obj}(\mathrm{L}_{rec}(u) \cup \{i\}, \alpha)$;
  fin
  $\mathrm{L}_{rec}(u) \leftarrow \mathrm{L}_{rec}(u) \cup \{i^*\}$;
fin
retourner $\mathrm{L}_{rec}(u)$

bornée de l'algorithme est couramment adoptée dans la littérature [Ziegler et al., 2005; Di Noia et al., 2014; Sha et al., 2016; Wu et al., 2018]. Nous désignons ici par $\mathrm{L}_{candidats}(u)$ la liste des m premiers items de $\mathrm{L}(u)$ qui constituent l'entrée de l'algorithme glouton borné.

4.2.4 Autres approches de diversification

Comme mentionné précédemment, la plupart des approches existantes d'optimisation de la diversité des recommandations sont des approches de post-traitement (*i.e.* elles interviennent *après* la phase de prédiction des *ratings*). Cependant, certains travaux récents [Cheng et al., 2017; Li et al., 2017] envisagent le problème différemment et considèrent la diversité *pendant* la phase prédictive en utilisant des approches d'*apprentissage d'ordonnancement* (LTR pour *Learning To Rank*). L'idée principale de ces approches est d'entraîner des modèles d'apprentissage supervisé sur un ensemble d'instances d'entraînement déterminé de façon empirique. La méthode de descente de gradient peut être utilisée pour optimiser une fonction de perte (*Loss*) qui prend en compte à la fois la diversité et l'*accuracy*. Il convient de noter que la méthode *LTR* est souvent liée à un système de recommandation spécifique, par exemple le modèle de factorisation matricielle comme dans [Cheng et al., 2017; Li et al., 2017]. Contrairement aux approches de post-traitement, la méthode *LTR* n'est donc pas généralisable à l'ensemble des systèmes de recommandation. En outre, l'approche d'optimisation par post-traitement glouton est une méthode classique qui est largement et continuellement utilisée dans la littérature de par sa simplicité [Bradley and Smyth, 2001; Ziegler et al., 2005; Vargas and Castells, 2011; Adomavicius and Kwon, 2012; Di Noia et al., 2014; Sha et al., 2016; Jugovac et al., 2017; Wu et al., 2018; Steck, 2018; Wang et al., 2019].

De ce fait, nous choisissons d'omettre les approches *LTR* dans les expérimentations de cette étude et nous concentrons sur les heuristiques gloutonnes de post-traitement pour l'optimisation de la diversité des recommandations. Cela nous permet de comparer les

performances en termes de diversité pour différents systèmes de recommandation et de mesurer l'intérêt d'y ajouter un post-traitement.

4.3 Protocole d'évaluation

Dans cette section, nous présentons le protocole d'évaluation sur lequel s'appuie la présente étude.

4.3.1 Les systèmes de recommandation considérés

Comme présenté dans la section 4.1.2, les systèmes de recommandation que nous considérons dans cette étude incluent : (1) l'approche basée sur la popularité des items (*TopPopular*) ; (2) l'approche de filtrage basé sur la description du contenu des items (*CBF*) ; (3) l'approche de filtrage collaboratif basé sur la proximité entre items (*IBCF*) ; (4) l'approche de CF basé sur les modèles (*SVD*) ; (5) l'approche de CF basé sur les réseaux de neurones profonds (*DNN*) ; (6) l'approche hybride basée sur les *embeddings* de graphes de connaissances (*KGE*) et (7) l'approche hybride qui combine la popularité et les contenus des items (*CBF-TopPopular*). Dans ce qui suit, nous rappelons brièvement les principes de ces SdR.

4.3.1.1 Approche basée sur la popularité des items (*TopPopular*)

La popularité des items est connue pour être un facteur important de recommandation [Abdollahpouri, 2019]. En tant qu'approche non personnalisée, le modèle fournit à chaque utilisateur, parmi les items qu'il n'a pas encore notés, les plus populaires. La popularité $\mathrm{P}op(i)$ d'un item donné i est définie comme le nombre d'utilisateurs ayant noté cet item. Formellement, si on note U l'ensemble des utilisateurs, I_u l'ensemble des items du profil de l'utilisateur $u \in \mathrm{U}$, alors la popularité d'un item i, notée $\mathrm{P}op(i)$ est définie par :

$$\mathrm{P}op(i) = |\{u \in \mathrm{U} | i \in \mathrm{I}_u\}| \tag{4.7}$$

Malgré sa simplicité, une étude récente [Ferrari Dacrema et al., 2019] a montré que le modèle *TopPopular* pouvait conduire à des recommandations plus pertinentes sur certains jeux de données que des modèles récents basés sur des réseaux de neurones profonds, comme par exemple le modèle CMN [Ebesu et al., 2018].

4.3.1.2 Approche basée sur le contenu et les données liées (*CBF*)

Le principe d'un système de recommandation basé sur le contenu est de proposer aux utilisateurs les items qui sont, d'après la description de leur contenu, similaires à ceux qu'ils

ont aimés (cf. section 2.2.2.1). Pour cela, un tel système commence par construire un espace vectoriel dans lequel les profils utilisateurs et les items sont représentés. Des approches plus robustes exploitent les caractéristiques structurées des items à l'aide d'ontologies et de données liées pour mieux représenter les items.

Pour prédire la pertinence d'un item non-évalué par un utilisateur, l'approche CBF considérée ici s'appuie sur l'équation (4.8). Dans cette formule, $profile(u)^+$ représente les items aimés par l'utilisateur u et la fonction de similarité $sim(i,j)$ fait référence à la similarité sémantique entre les items i et j. Le calcul de cette similarité sémantique est basé sur le graphe de connaissances DBpedia, comme dans [Di Noia et al., 2012b]. Les détails de ce calcul sont fournis en section 4.3.5.3.

$$rel(u,i) = \frac{\sum_{j \in profile(u)^+} sim(i,j)}{|profile(u)^+|} \tag{4.8}$$

4.3.1.3 Approche par filtrage collaboratif basé sur les items (*IBCF*)

Comme présenté dans la section 2.1.3.1, il existe deux types d'approche de CF basé sur le voisinage : le premier est basé sur les utilisateurs (*UBCF*, cf. équation (2.15)) et le second sur les items (*IBCF*, cf. équation (2.16)). Schématiquement, étant donné un utilisateur u et un item i ($i \in \mathrm{L}_{candidats}(u)$), ces deux types d'approche de CF basé sur le voisinage identifient d'abord un ensemble d'utilisateurs ou d'items similaires et ensuite utilisent les *ratings* des voisins concernés pour prédire le score $\hat{r}_{u,i}$.

Il convient de noter que dans la présente étude, nous avons omis *UBCF* pour trois raisons. Premièrement, des travaux [Sarwar et al., 2001; Ekstrand et al., 2014] ont montré que *IBCF* surpasse généralement *UBCF*, en termes d'*accuracy* des recommandations et de satisfaction des utilisateurs. Deuxièmement, *IBCF* est plus flexible que *UBCF* car cette méthode nécessite moins de mémoire pour construire la matrice de similarités pour les items que pour les utilisateurs. Ceci s'explique par le fait que le nombre d'items dans un système de recommandation réel, par exemple Amazon, est généralement beaucoup plus faible que le nombre d'utilisateurs ($|\mathrm{I}| \ll |\mathrm{U}|$). Enfin, l'étude réalisée par [Ekstrand et al., 2014] a montré qu'il n'y a pas de différence significative entre *IBCF* et *UBCF* en termes de diversité des recommandations.

4.3.1.4 Approche dite *singular value decomposition* (*SVD*)

Le modèle SVD [Koren et al., 2009] est l'un des modèles de CF les plus populaires. Il applique des techniques de factorisation matricielle pour mettre en correspondance les utilisateurs et les items dans un espace factoriel latent de dimension réduite d. Les représentations apprises sont ensuite utilisées pour prédire les *ratings*. Formellement, chaque item i est associé à un vecteur $\mathbf{q}_i \in \mathbb{R}^d$ et chaque utilisateur u est associé à un vecteur $\mathbf{p}_u \in \mathbb{R}^d$. La prédiction de note est obtenue par le produit scalaire de ces deux vecteurs, *i.e.*

$\widehat{r}_{u,i} = \mathbf{q}_i^{\mathrm{T}} \mathbf{p}_u$. Pour apprendre ces vecteurs de facteurs, SVD minimise une fonction objectif représentant l'erreur quadratique régularisée entre les scores prédits et les scores réels (cf. équation (4.9)). Pour réaliser cette optimisation, la descente de gradient stochastique (SGD) est souvent utilisée (cf. section 2.1.3.2.2).

$$Loss = \sum_{r_{u,i} \in \mathrm{R}_{train}} (r_{u,i} - \mathbf{q}_i^{\mathrm{T}} \mathbf{p}_u)^2 + \lambda(||\mathbf{q}_i||^2 + ||\mathbf{p}_u||^2) \tag{4.9}$$

4.3.1.5 Approche basée sur les réseaux de neurones profonds (*DNN*)

En général, les SdR basés sur les réseaux de neurones prennent la matrice des *ratings* comme couche d'entrée et génèrent des scores pour chaque paire utilisateur-item à l'issue de la couche de sortie. Les couches cachées, au sein d'une architecture neuronale multi-couches, mettent en lumière les structures latentes des interactions entre utilisateurs et items. L'approche *DNN* considérée, proposée dans [He et al., 2017], est l'une des approches de recommandation basées sur les réseaux de neurones profonds les plus citées [Ferrari Dacrema et al., 2019]. Plus précisément, DNN concatène deux réseaux de neurones : GMF (*Generalized Matrix Factorization*) et MLP (*Multi-Layer Perceptron*). L'idée principale de GMF est de généraliser le modèle de factorisation matricielle (équation (4.9)) dans lequel i) les facteurs latents sont traités de la même manière (le poids étant identique pour tous les facteurs latents), et ii) une fonction linéaire (un produit scalaire) est utilisée pour modéliser l'interaction utilisateur-item. Le réseau GMF, quant à lui, peut i) apprendre automatiquement différents poids pour chaque facteur latent et ii) prendre en compte de manière non linéaire les interactions entre utilisateurs et items. Le réseau MLP permet d'apprendre des interactions utilisateur-item plus subtiles en ajoutant des couches cachées au vecteur concaténé de $\mathbf{p}_u$ et $\mathbf{q}_i$. Enfin, les dernières couches cachées du GMF et du MLP sont concaténées pour générer le score final d'une paire utilisateur-item donnée (voir section 2.1.3.2.4 pour plus de détails).

4.3.1.6 Approche hybride basée sur les *embeddings* de graphe de connaissances (*KGE*)

Dans le contexte des systèmes de recommandation, deux graphes de connaissances (KG pour *Knowledge Graph*) peuvent être construits : le KG des Contenus des Items *ICKG* et le KG des Préférences des Utilisateurs *UPKG*. Concrètement, ICKG contient des triplets qui décrivent les connaissances liées aux items (*e.g.* le directeur d'un film) et UPKG contient des triplets décrivant les connaissances relatives aux préférences des utilisateurs. En combinant ces deux graphes de connaissances, on peut construire un graphe hybride HybridKG (voir figure 2.10 pour un exemple).

La prédiction du score de pertinence pour chaque item i non-évalué par l'utilisateur u est basée sur les représentations vectorielles latentes (*i.e.* les plongements) apprises par un modèle d'*embedding* de graphe de connaissances. Dans ce cadre, la recommandation peut être considérée comme une tâche de complétion du graphe de connaissances ou

une tâche de prédiction des triplets du type $\langle u, \text{aime}, i \rangle$. Par exemple, pour les modèles d'*embeddings* basés sur les distances (comme TransE [Bordes et al., 2013]), nous pouvons utiliser la fonction $score(u,i) = \|\mathbf{u} + \mathbf{aime} - \mathbf{i}\|$ avec **u**, **aime** et **i** les vecteurs d'*embedding* de l'entité (utilisateur) u, de la relation *aime* et de l'entité (item) i, respectivement (cf. section 2.2.2.2.2).

4.3.1.7 Approche hybride combinant la popularité et le contenu des items (*CBF-TopPopular*)

Les recommandations basées sur la popularité des items sont non-personnalisées et privilégient les items populaires par rapport aux items de niche, tandis que les recommandations basées sur le contenu sont généralement sur-spécialisées et favorisent les items de niche (très spécifiques). Nous envisageons ainsi une approche hybride simple qui combine les modèles CBF et TopPopular présentés précédemment. Formellement, étant donné un utilisateur u et l'un des items non-notés i, le score de pertinence du modèle CBF-TopPopular est défini par l'équation (4.10) comme suit :

$$rel(u,i) = \omega * \text{TopPopular}(i) + (1-\omega) * \text{CBF}(u,i) \tag{4.10}$$

où $\text{TopPopular}(i)$ et $\text{CBF}(u,i)$ représentent respectivement la popularité de l'item i (cf. équation (4.7)) et le score de pertinence $rel(u,i)$ calculé par le modèle CBF (cf. équation (4.8)). L'hyper-paramètre ω pondère l'importance de chacun des deux modèles combinés dans cette approche hybride.

4.3.2 Jeux de données & pré-traitement

Les expérimentations sont basées sur trois jeux de données issus de corpus réels venant de trois différents domaines de recommandation : *MovieLens-1M*[1] pour le cinéma, *Anime*[2] pour le domaine des *animés* (dessins animés japonais inspirés des mangas) et *LibraryThing*[3] pour la littérature. *MovieLens-1M* est un jeu de données de référence, très utilisé, qui contient 1 million de *ratings* attribués par 6 040 utilisateurs sur 3 900 films. Le jeu de données *Anime* est un jeu de données public Kaggle contenant 1 597 830 *ratings* de 37 100 utilisateurs sur 12 294 animés. En plus de ces 1 597 830 *ratings* explicites, ce jeu de données contient également des valeurs de préférence implicites qui signifie que l'utilisateur a consulté l'animé sans le noter. Ces évaluations implicites ont été exclues de nos expérimentations. *LibraryThing* est un jeu de données de recommandation de livres qui contient 626 000 *ratings* de 7 112 utilisateurs sur 37 231 livres.

La base de connaissances DBpedia est un élément central de notre étude. Nous l'avons utilisée pour construire le modèle CBF de recommandation (cf. section 4.3.1.2), pour

[1] https://grouplens.org/datasets/movielens/1m/
[2] https://www.kaggle.com/CooperUnion/anime-recommendations-database
[3] https://www.librarything.com/

construire les graphes de connaissances *ICKG* et *UPKG* (voir leurs définitions dans la section 4.3.1.6) utilisés dans le modèle de recommandation KGE et pour calculer les similarités sémantiques des items afin d'estimer la diversité d'une liste d'items (cf. équation (4.1)). Afin d'exploiter cette gigantesque source de connaissances, les items doivent être mis en correspondance (alignés) avec les entités correspondantes dans DBpedia, identifiées à travers leurs URI. Les films dans *MovieLens-1M* et les livres dans *LibraryThing* ont été mis en correspondance par des travaux précédents [Di Noia et al., 2012b] et ces alignements sont librement accessibles. Cependant, le graphe de connaissances DBpedia a évolué depuis que ces alignements ont été réalisés. Certains alignements sont désormais invalides, et *pointent* vers des URI inexistants. Nous avons donc corrigé ces alignements manuellement. Pour le jeu de données Anime, nous avons d'abord extrait toutes les entités DBpedia du type *dbo:Manga* et leurs étiquettes en anglais en utilisant des requêtes SPARQL à partir du point d'accès de DBpedia[4]. Nous avons ensuite comparé les étiquettes (labels) de manga de DBpedia avec les noms d'animes fournis dans les méta-données du jeux de données, pour mesurer une similarité entre les items d'Anime et les entités DBpedia, en termes de labels. Un item de la base de données Anime est mis en correspondance avec une entité DBpedia si la similarité de Levenshtein normalisée entre leurs étiquettes est supérieure à 0.95.

Cette procédure d'alignement nous a permis d'obtenir 3 301 items dans MovieLens (85%), 656 items dans Anime (5%) et 11 695 items dans LibraryThing (31%) alignés avec des entités DBpedia correspondants. Nous avons ensuite supprimé les items non alignés, ainsi que les *ratings* associés, dans les jeux de données correspondants. Il convient de noter que ce pré-traitement des jeux de données entraîne la suppression de nombreux items (en particulier pour le jeux de données Anime) ; cependant, la plupart des items supprimés ont, de toute façon, peu d'évaluations côté utilisateur. Les items les plus populaires sont souvent ceux présents dans DBpedia et ayant reçu le plus d'évaluations, tandis que les items moins connus, souvent absents de DBpedia, ont tendance à recevoir peu d'évaluations. Par conséquent, ce pré-filtrage des jeux de données n'est pas aussi radical qu'il n'y paraît. Même pour le jeu de données Anime, dans lequel seulement 5% des items sont conservés, 63% des *ratings* sont préservées.

Le tableau 4.1 résume les principales caractéristiques des trois jeux de données prétraités. La rareté des données (*sparsity*) dans un jeu de données représente la proportion de *ratings* manquants dans l'ensemble de la matrice d'évaluation utilisateur-item. Enfin, nous divisons aléatoirement chaque jeu de données en un jeu d'apprentissage (80% des *ratings*) et un jeu de test (20% des *ratings*).

[4] https://dbpedia.org/sparql

TABLEAU 4.1 – Principales caractéristiques des jeux de données (filtrés)

Jeux	domaine	# utilisateurs (%préservé)	# items (%préservé)	# *ratings* (%préservé)	*sparsity*
MovieLens-1M	film	6 040 (100%)	3 301 (85%)	948 840 (95%)	95.09%
Anime	animé	27 454 (74%)	656 (5%)	1 007 384 (63%)	94.41%
LibraryThing	livre	6 789 (95%)	11 695 (31%)	403 860 (65%)	99.49%

4.3.3 Construction des graphes de connaissances

Pour chaque jeu de données, nous avons construit deux graphes de connaissances, ICKG et UPKG. Pour construire les trois instances ICKG, nous avons sélectionné un sous-ensemble adapté de propriétés de l'ontologie DBpedia, comme proposé dans [Palumbo et al., 2018b]. Les propriétés sélectionnées pour chaque jeu de données sont présentées dans le tableau 4.2. Pour construire les trois instances UPKG, la relation « *aime* » doit être établie entre les entités de type utilisateur et item. Suivant les auteurs de [Di Noia et al., 2016; Palumbo et al., 2018b], nous avons considéré qu'un utilisateur u aime un item i si $r_{u,i} \geq 4$ pour le jeu de données *MovieLens-1M* (dans lequel les *ratings* varient entre 1 et 5) et $r_{u,i} \geq 8$ pour les jeux de données *Anime* et *LibraryThing* (dans lesquels les *ratings* varient entre 1 et 10). Le tableau 4.3 fournit le nombre de triplets déduits dans les instances ICKG et UPKG construites pour chaque jeu de données.

TABLEAU 4.2 – Propriétés de l'ontologie DBpedia prises en compte pour chaque jeu de données

Jeu de données	Propriétés considérées
MovieLens-1M	dbo:director, dbo:starring, dbo:distributor, dbo:writer, dbo:musicComposer, dbo:producer, dbo:cinematography, dbo:editing, dct:subject
Anime	dbo:author, dbo:publisher, dbo:magazine, dbp:director, dbp:genre, dbo:illustrator, dbp:writer, dct:subject
LibraryThing	dbo:author, dbo:publisher, dbo:literaryGenre, dbo:mediaType, dbo:subsequentWork, dbo:previousWork, dbo:series, dbo:country, dbo:language, dbo:coverArtist, dct:subject

TABLEAU 4.3 – Nombre de triplets composant les deux graphes de connaissances construits pour chaque jeu de données.

Jeu de données	ICKG (#triplets)	UPKG (#triplets)
MovieLens-1M	93 269	442 838
Anime	12 266	490 427
LibraryThing	134 333	207 884

4.3.4 Construction des modèles de recommandation

Nous avons suivi la proposition de [Steck, 2013] selon laquelle tous les items du catalogue peuvent être recommandés (cf. *allUnratedItems* de la section 2.1.6.4). Par conséquent, dans nos expérimentations, l'ensemble L(u) (cf. algorithme 1) contient tous les items que

u n'a pas notés auparavant, qu'ils soient dans le jeu d'entraînement ou dans le jeu de test. Dans nos expérimentations, nous comparons sept modèles de recommandation de quatre types différents, listés dans le tableau 4.4. Pour chaque jeu de données, nous procédons comme suit pour construire les modèles de recommandation : 1) l'optimisation des hyper-paramètres est faite pour chacun des SdR en optimisant leur *accuracy* et 2) les modèles sont entraînés avec les hyper-paramètres optimaux trouvés.

TABLEAU 4.4 – Récapitulation des SdR testés

SdR (Référence)	Type de SdR	Description brève
CBF [Di Noia et al., 2012b]	Filtrage basé sur le contenu	Méthode basée sur la similarité sémantique des items
TopPopular [Ferrari Dacrema et al., 2019]	Non-personnalisé	Méthode basée sur la popularité des items
CBF-TopPopular (équation (4.10))	Hybride	Méthode combinant CBF et TopPopular
IBCF [Sarwar et al., 2001]	Filtrage collaboratif	Méthode de NBCF basée sur les items
SVD [Koren et al., 2009]	Filtrage collaboratif	Méthode basée sur les facteurs latents (factorisation matricielle)
KGE [Palumbo et al., 2018b]	Hybride	Méthode basée sur les plongements de graphes de connaissances
DNN (NeuMF) [He et al., 2017]	Filtrage collaboratif	Méthode de CF basée sur les réseaux de neurones profonds

Optimisation des hyper-paramètres En général, l'optimisation des hyper-paramètres des modèles d'apprentissage automatique est une étape visant à trouver la combinaison d'hyper-paramètres conduisant aux meilleures performances. Il s'agit d'une étape cruciale pour les modèles de recommandation modernes, car le comportement des SdR peut être fortement influencé par le jeu de données et les hyper-paramètres utilisés pendant l'entraînement [Jannach et al., 2015].

Nous avons adopté l'optimisation bayésienne (BO) [Snoek et al., 2012] pour optimiser les hyper-paramètres des SdR comparés. Par rapport aux méthodes d'optimisation par recherche aléatoire et par recherche sur grille, qui prennent généralement beaucoup de temps, l'optimisation bayésienne est une procédure itérative qui prend en compte les performances des valeurs d'hyper-paramètre évaluées lors des précédentes itérations pour déterminer la combinaison d'hyper-paramètres à évaluer lors de l'itération suivante. Le caractère itératif de l'optimisation bayésienne (l'utilisateur peut choisir le nombre d'itérations souhaité) et le fait que les espaces des hyper-paramètres n'ont pas à être entièrement balayés réduisent considérablement le temps de calcul. Dans nos expérimentations, le *package BayesianOptimization* de [Nogueira, 2014] a été utilisé pour optimiser les hyper-paramètres des modèles de recommandation comparés.

Parmi les 7 SdR comparés dans la présente étude, les modèles TopPopular et CBF n'ont pas d'hyper-paramètres à optimiser. Plus précisément, pour le modèle TopPopular, le score de pertinence de chaque item i, parmi l'ensemble d'items non-évalués par l'utilisateur u, *i.e.* $\mathrm{L}(u)$, est calculé à l'aide de l'équation (4.7). Pour le modèle CBF, le score de pertinence d'une paire utilisateur-item donnée, c'est-à-dire $rel(u,i)$, est calculé à l'aide de l'équation (4.8). Dans l'équation (4.8), la similarité sémantique entre deux items est estimée en mesurant la similarité cosinus entre leurs vecteurs d'*embeddings* appris à partir du graphe de connaissances correspondant lié aux contenus des items (ICKG). Par conséquent, des scores de pertinence élevés sont attribués aux items qui sont sémantiquement

proches de ceux aimés par les utilisateurs.

Pour les cinq modèles de recommandation restants, à savoir CBF-TopPopular, IBCF, SVD, DNN et KGE, nous avons effectué 20 itérations d'optimisation bayésienne (BO) sur chacun des trois jeux de données, afin d'optimiser les hyper-paramètres de chaque modèle sur chaque jeu de données d'entraînement. À cette fin, nous avons d'abord choisi de diviser chaque jeux d'entraînement (représentant 80% du jeu de données) en deux ensembles : un sous-ensemble d'entraînement (80% de l'ensemble d'entraînement initial) et un ensemble de validation (20% de l'ensemble d'entraînement initial). Pour un modèle et un jeu de données particulier, la procédure, schématisée par la figure 4.1, est donc la suivante. Le processus de BO utilise le sous-ensemble d'entraînement pour entraîner le modèle, et ses hyper-paramètres sont optimisés sur l'ensemble de validation correspondant de manière à maximiser la métrique de *Mean average precision* (MAP) (cf. équation (2.35)). Après l'étape de BO, la configuration optimale des hyper-paramètres résultante est utilisée pour entraîner le modèle sur le jeu d'entraînement complet (80% du jeu de données correspondant) et le modèle est finalement testé sur chaque jeu de test.

Les espaces de variation des hyper-paramètres que nous avons optimisés pour chaque modèle sont les suivants :

- CBF-TopPopular : le facteur de pondération ω. Dans l'équation (4.10), nous avons choisi de normaliser le score de popularité des items, c'est-à-dire de transposer les valeurs de $TopPopular(i)$ dans l'intervalle [0, 1] afin qu'il ait la même échelle que CBF(u, i).

- IBCF : le nombre k de voisins de l'item cible i, c'est-à-dire $|N_i|$ dans l'équation (2.16).

- SVD : le nombre de facteurs $n_{factors}$, le nombre d'époques (*epoch* en anglais) d'apprentissage de la descente de gradient stochastique, le taux d'apprentissage η et le terme de régularisation λ (cf. section 2.1.3.2.2).

- KGE : nous avons entraîné le modèle TransE sur HybridKG (hybridation de ICKG et UPKG, cf. section 4.3.1.6) pour le jeu de données correspondant. TransE est un modèle de référence d'*embeddings* pour les graphes de connaissances. [Palumbo et al., 2018a] a démontré que le modèle de recommandation basé sur TransE était plus performant que ceux basés sur d'autres modèles d'*embeddings*. Les hyper-paramètres de KGE sont constitués de la dimension des vecteurs d'*embeddings* d, de la marge de perte γ, du taux d'apprentissage η, de la taille du *mini-batch* s_{batch} et du nombre d'époques d'apprentissage.

- DNN : la taille du *mini-batch* s_{batch}, le nombre de facteurs $n_{factors}$, le nombre de couches cachées dans le réseau MLP n_{layers}, le taux d'apprentissage η, le nombre d'époques et le nombre d'échantillons négatifs pour l'entraînement n_{neg}.

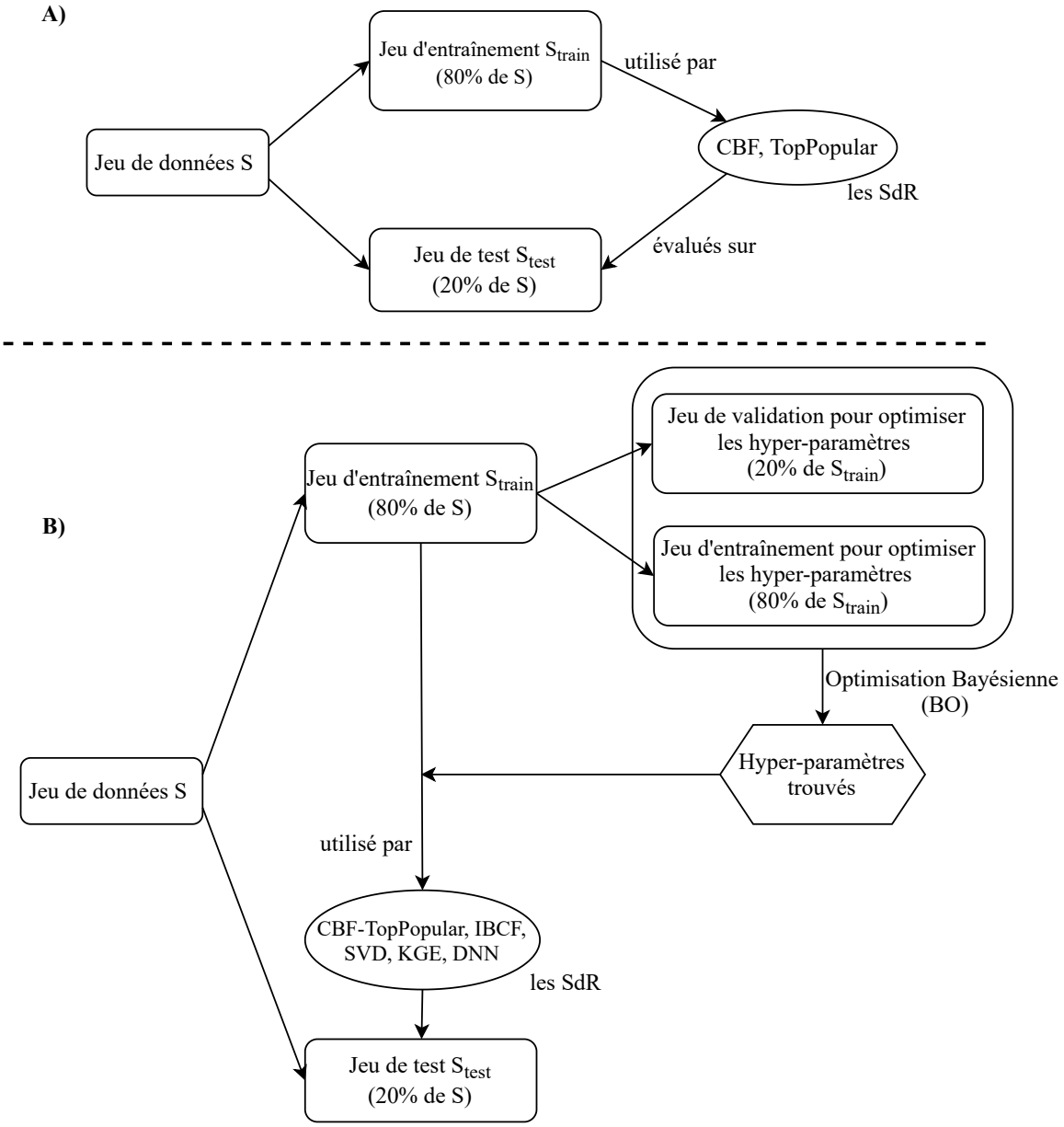


FIGURE 4.1 – Procédure de construction des SdR pour : A) les SdR ne disposant pas d'hyper-paramètres et B) les SdR disposant des hyper-paramètres.

L'implémentation de ces modèles de recommandation a été réalisée à l'aide de *packages open-source*. Plus précisément, la librairie Python *Surprise*[5] de [Hug, 2020] a été utilisée pour entraîner les modèles IBCF et SVD. Pour le modèle DNN, nous avons utilisé une implémentation disponible[6]. L'apprentissage du modèle KGE a été effectué avec le *package Pykeen* [Ali et al., 2019] qui est une référence pour l'apprentissage de plongements de graphes de connaissances.

Les détails de la configuration des hyper-paramètres ainsi que les hyper-paramètres optimaux trouvés pour chaque jeu de données sont fournis dans les tableaux A.1 et A.2 de l'annexe A. Toutes les ressources de notre étude (jeux de données, code source, *map-*

[5] https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html
[6] https://github.com/guoyang9/NCF

pings générés, *embeddings* appris, etc.) sont disponibles dans le répertoire GitHub[7]. Cela permet de garantir la reproductibilité de notre étude et devrait faciliter les analyses ultérieures.

4.3.5 Métriques d'évaluation

Nous adoptons les métriques suivantes pour évaluer la qualité d'une liste de recommandations comprenant n items, en termes d'*accuracy* et de diversité.

4.3.5.1 Métriques d'*accuracy*

Nous évaluons l'*accuracy* des modèles de recommandation testés avec les métriques standards suivantes :

- **Précision** : représente la proportion d'items pertinents parmi les n items recommandés pour un utilisateur donné u (cf. équation (2.32)).
- **Rappel** : représente le pourcentage des items pertinents pour l'utilisateur u, au sein du catalogue, qui sont inclus dans sa liste de recommandations (cf. équation (2.33)).
- **F1-mesure** : désigne la moyenne harmonique de la précision et du rappel (cf. équation (2.34)).
- ***Mean Average Precision* (MAP)** représente la moyenne de toutes les valeurs d'*average precision* (AP) sur tous les utilisateurs (cf. équation (2.35)). Contrairement à la précision et au rappel, qui ne tiennent pas compte des positions des items pertinents dans la liste, la MAP est une mesure de précision qui met l'accent sur le fait que les items pertinents doivent être classés en tête des recommandations [Steck, 2013].

4.3.5.2 Métriques de diversité

La performance des modèles de recommandation en matière de diversité est évaluée à l'aide des métriques suivantes, basées sur l'ILD et l'ILD_p.

- $\overline{\textbf{ILD}}$ représente la moyenne des diversités intra-liste (ILD) calculée sur l'ensemble des utilisateurs. Nous appelons également cette mesure la diversité *absolue* par opposition à la mesure suivante qui évalue la diversité des listes de recommandations par rapport à la diversité des items présents dans les profils utilisateurs.
- $\mathbf{R}^2(\text{ILD}, \text{ILD}_p)$, dénommé diversité *relative* (cf. équation (4.3)), est le coefficient de détermination de deux listes de valeurs : 1) des valeurs d'ILD, c'est-à-dire les diversités des items recommandés aux utilisateurs par un SdR et 2) les valeurs d'ILD_p, c'est-à-dire la diversité des items dans les profils utilisateurs correspondants. Cette

[7] https://github.com/lgi2p/Rec_Sys_Diversity_Study

métrique permet d'évaluer dans quelle mesure la diversité des recommandations fournies par un SdR correspond aux besoins des utilisateurs.

4.3.5.3 Similarité sémantique

La fonction de similarité $sim(i, j)$ de l'équation (4.1) est cruciale pour mesurer l'ILD d'une liste d'items (à la fois pour les diversités *absolue* et *relative*). La similarité entre deux items peut être mesurée à partir de la matrice de *ratings* utilisateur-item. Cela présente l'avantage d'être toujours réalisable, peu importe le SdR, dès qu'on a accès à une matrice de *ratings*. Par ailleurs, les mesures de similarité basées sur la sémantique des items nécessitent que certaines caractéristiques sémantiques soient associées aux items, ce qui peut être difficile à obtenir (voir section 4.3.2). Cependant, les similarités basées sur la matrice des *ratings* présentent elles-mêmes certains inconvénients. Le premier est lié à la rareté des données dans la matrice (*sparsity*) : le fait d'avoir de nombreuses valeurs manquantes, comme c'est souvent le cas puisque la plupart des items sont généralement évalués par peu d'utilisateurs, peut conduire à des mesures de similarité peu fiables [Aytekin and Karakaya, 2014]. Si l'on considère un nouvel item, pour lequel il n'y a (presque) aucune notation disponible, sa similarité avec d'autres items sera mesurée de manière inexacte en se basant sur la matrice. Le deuxième inconvénient des similarités basées sur la matrice creuse est le biais de popularité qu'elles introduisent. Par rapport aux items de niche, les items populaires ont tendance à avoir plus d'utilisateurs en commun (*i.e.* utilisateurs ayant évalué les deux items) et donc à être considérés comme plus similaires entre eux [Hou et al., 2018].

Des études récentes ont étudié les fonctions de similarité des items au regard des jugements humains [Yao and Harper, 2018; Trattner and Jannach, 2020]. L'objectif principal de ces travaux est d'étudier les perceptions des utilisateurs sur la similarité entre les items. Les auteurs de [Yao and Harper, 2018] demandent aux utilisateurs d'évaluer la similarité entre un film cible et des films sélectionnés à partir de différentes fonctions de similarité (par exemple, basées sur la matrice de *ratings* ou sur le contenu). D'après leur étude, lorsque les films sont sélectionnés sur la base de leur contenu, ils sont généralement considérés comme plus similaires au film cible que ceux sélectionnés via une similarité entre *ratings*. Les auteurs de [Trattner and Jannach, 2020], quant à eux, ont étudié plus profondément les perceptions des utilisateurs sur les similarités des items. Leur étude s'est principalement appuyée sur les similarités basées sur le contenu, en essayant de comprendre comment différentes caractéristiques des items (par exemple, pour les films : le titre, l'intrigue, la qualité d'image, etc.) pouvaient déterminer la similarité des items perçue par les utilisateurs et comment ils évaluent l'utilité des recommandations obtenues à partir de différentes fonctions de similarité.

L'utilisation de similarités sémantiques semble donc préférable lorsque cela est possible (*i.e.* lorsque des données relatives au contenu des items sont disponibles). La simi-

larité sémantique a été largement étudiée dans le domaine de l'ingénierie des connaissances et de nombreuses métriques ont été proposées dans la littérature [Harispe et al., 2015]. Visant à représenter la relation sémantique entre les significations des concepts, elles ont été largement adoptées dans le domaine de la recherche d'information puis dans celui des systèmes de recommandation [Passant, 2010b; Carrer-Neto et al., 2012; Di Noia et al., 2014; Di Noia et al., 2016; Meymandpour and Davis, 2020; García-Sánchez et al., 2020]. La présente étude s'appuie également sur la similarité sémantique des items. Pour l'estimer, nous considérons la similarité cosinus entre les vecteurs d'*embeddings* des items (basés sur ICKG). Cette similarité sémantique entre paires d'items est utilisée dans l'approche CBF pour la prédiction de la pertinence des items non-notés (équation (4.8)), ainsi que pour l'estimation de l'ILD d'une liste d'items (équation (4.1)).

4.3.6 Configuration choisie pour l'étude de la diversification

Nous présentons ici la configuration adoptée pour la présente étude. Tout d'abord, chacun des modèles de recommandation est censé recommander une liste de $n = 10$ items ayant les scores de prédiction de pertinence les plus élevés. Ces SdR sont ensuite évalués en termes d'*accuracy* et de diversité (à la fois *absolue* et *relative*).

Ensuite, pour analyser l'impact du post-traitement de diversification sur chacun de ces modèles de recommandation, nous avons exécuté l'algorithme glouton borné (cf. algorithme 1) avec deux fonctions objectifs différentes : la classique (équation (4.5)) et la personnalisée (équation (4.6)). Pour le post-traitement glouton borné, la borne m a été fixée à 100 (n^2), c'est-à-dire que $\mathrm{L}_{candidates}(u)$ contient les 100 premiers items ayant les meilleures prédictions. Pour différents modèles de recommandation et jeux de données, les valeurs possibles de leurs scores prédits, que l'on retrouve sous le terme $rel(\cdot)$ dans les fonctions objectifs de diversification, sont assez différents. Pour éviter les biais possiblement induits par cette hétérogénéité, nous avons choisi de normaliser ces valeurs $rel(\cdot)$. Plus précisément, pour normaliser le score de pertinence prédit $rel(\cdot)$, nous avons appliqué la formule $\frac{rel(\cdot)-rel_{min}}{rel_{max}-rel_{min}}$ avec rel_{min} et rel_{max} les valeurs minimales et maximales de la fonction $rel(\cdot)$ pour une méthode et un jeu de données spécifiques. Enfin, pour le facteur de diversification α, nous avons testé les 10 valeurs suivantes : $\{0, 0.1, ..., 0.9\}$.

Enfin, nous avons étudié l'impact de la taille de la borne m du post-traitement glouton borné sur les performances des SdR. En effet, une borne plus élevée (par exemple, $m = 200$) peut permettre de trouver des items plus diversifiés, au risque de sacrifier pour cela une partie de l'*accuracy*.

4.4 Résultats et discussions

Dans cette section, nous présentons et discutons d'abord les performances des systèmes de recommandation (SdR) testés, en termes d'*accuracy*, de diversité *absolue* et de diver-

sité *relative* (QR1). Ensuite, nous étudions la sensibilité des SdR testés au post-traitement glouton de diversification (QR2 et QR3). Troisièmement, nous discutons de l'impact du nombre d'items candidats m sur la qualité des recommandations obtenues. Enfin, nous discutons des limites et des implications de notre étude.

4.4.1 Performances en termes d'*accuracy* et de diversité des SdR testés

Le tableau 4.5 résume les performances en termes d'*accuracy* obtenues par les sept SdR testés (de 4 familles différentes) sur les trois jeux de données considérés.

Considérons d'abord les deux jeux de données MovieLens et Anime. Les modèles basés sur des techniques plus récentes (*i.e.*, KGE pour *knowledge graph embeddings* et DNN pour *deep neural networks*) proposent toujours des recommandations plus pertinentes que les autres modèles plus standards. En effet, pour les 8 configurations (4 métriques × 2 jeux de données), DNN est classé 7 fois comme meilleur modèle et KGE une fois. Les modèles basés sur la popularité des items (TopPopular et CBF-TopPopular) sont légèrement moins pertinents que les meilleurs modèles, mais on note que CBF-TopPopular est toujours (au moins légèrement) meilleur que TopPopular. Notez qu'une simple hybridation de CBF et de TopPopular permet d'obtenir 6 fois (sur 8) le deuxième modèle le plus pertinent. Cette simple hybridation vient donc s'intercaler dans le classement entre les deux modèles les plus récents que sont KGE et DNN. Les modèles SVD et CBF ont des performances d'*accuracy* comparables sur ces deux jeux de données et l'approche IBCF a l'*accuracy* la plus faible. Par exemple, sur MovieLens, la MAP est de ~ 0.19 pour DNN, KGE et CBF-TopPopular, d'environ ~ 0.16 pour TopPopular, de ~ 0.09 pour SVD et CBF, et de ~ 0.04 pour IBCF. Le classement des performances reste inchangé lorsque l'on considère la MAP pour le jeu de données Anime même si la différence entre DNN (~ 0.43) et KGE / CBF-TopPopular (~ 0.35 / ~ 0.36) est plus importante. La même remarque peut être faite pour d'autres métriques sur le jeu de données Anime. Par exemple, en considérant la F1-mesure, on observe les résultats suivants : ~ 0.25 pour DNN contre ~ 0.18 pour KGE et CBF-TopPopular et seulement ~ 0.1 pour SVD, ~ 0.08 pour CBF et ~ 0.05 pour IBCF.

De manière assez surprenante, les résultats en termes d'*accuracy* sont très différents sur le troisième jeu de données LibraryThing. Les modèles de recommandation exploitant la sémantique des items, c'est-à-dire celui basé sur le contenu (CBF) ou les modèles hybrides (KGE et CBF-TopPopular), sont ici plus pertinents que les modèles basés uniquement sur le filtrage collaboratif (*i.e.* sur la matrice des *ratings*). En effet, sur ce jeu de données, les SdR hybrides sont toujours les meilleurs modèles : KGE arrive toujours en tête et CBF-TopPopular en deuxième position, pour les 4 métriques, suivis par CBF, qui est toujours classé troisième. Par exemple, sur ce jeu de données, KGE obtient la valeur de MAP la plus élevée (~ 0.20), suivi de CBF-TopPopular (~ 0.16) et de CBF (~ 0.10). L'approche basée sur les réseaux de neurones profonds n'arrive qu'en quatrième position (~ 0.09), suivie de TopPopular (~ 0.08), SVD (~ 0.07) et IBCF (~ 0.005). Les mauvaises performances,

TABLEAU 4.5 – Comparaison des performances des SdR en termes de l'*accuracy*

Métrique	Jeu de données	CBF	TopPopular	CBF-TopPopular	IBCF	SVD	KGE	DNN
Précision	MovieLens	0.0322	0.0808	0.0869	0.0175	0.0406	**0.0887**	*0.0857*
	Anime	0.0586	0.1308	0.1352	0.0356	0.0734	*0.1313*	**0.1786**
	LibraryThing	*0.0302*	*0.0302*	0.0512	0.0019	0.0217	**0.0695**	0.0299
Rappel	MovieLens	0.0259	0.0498	*0.0608*	0.0110	0.0278	0.0625	**0.0821**
	Anime	0.1256	0.3065	0.3159	0.0847	0.1772	*0.3114*	**0.4139**
	LibraryThing	*0.0539*	0.0434	0.0847	0.0024	0.0307	**0.1132**	0.0449
F1-Mesure	MovieLens	0.0287	0.0616	*0.0715*	0.0135	0.0330	0.0733	**0.0839**
	Anime	0.0799	0.1834	0.1894	0.0501	0.1038	*0.1847*	**0.2495**
	LibraryThing	*0.0387*	0.0356	0.0638	0.0021	0.0254	**0.0861**	0.0359
MAP	MovieLens	0.0849	0.1661	0.1889	0.0442	0.0951	*0.1860*	**0.1956**
	Anime	0.1591	0.3476	0.3635	0.1026	0.1593	*0.3498*	**0.4270**
	LibraryThing	*0.0980*	0.0819	0.1617	0.0045	0.0707	**0.1966**	0.0889

Pour chaque jeu de données et chaque métrique, le meilleur score est en gras, le deuxième meilleur est souligné et le troisième est en italique.

en termes d'*accuracy*, des modèles de CF sur le jeu de données LibraryThing peuvent être dues à la grande *sparsity* de la matrice de *ratings* (99.49%, voir tableau 4.1). Cela peut empêcher ces modèles de CF d'apprendre une représentation adéquate des vecteurs de facteurs pour les utilisateurs et les items. Dans un contexte de rareté de données aussi élevée, même une approche simple basée sur le contenu pourrait conduire à des recommandations plus pertinentes que l'approche DNN. Cela confirme que la prise en compte des connaissances et de la sémantique des items permet de soulager les problèmes de démarrage à froid et de rareté des données auxquels sont classiquement confrontées les approches de CF [Natarajan et al., 2020].

Le tableau 4.6 résume les valeurs de diversité obtenues par les sept SdR testés sur les trois jeux de données considérés. Examinons d'abord la diversité *absolue* ($\overline{\text{ILD}}$) des recommandations. Le modèle CBF, qui s'appuie exclusivement sur les connaissances des items, conduit à des recommandations beaucoup moins diversifiées que les autres modèles. Par exemple, sur le jeu de données MovieLens, l'ILD moyen de CBF n'est que de ~ 0.28 alors qu'il varie entre ~ 0.40 (CBF-PopPopular) et ~ 0.70 (IBCF) pour les autres modèles. La même tendance est observée pour les deux autres jeux de données. Cela confirme que les recommandations basées sur le contenu des items sont généralement sur-spécialisées. À l'autre extrémité du spectre, IBCF et SVD, les deux modèles traditionnels de CF, fournissent les recommandations les plus diversifiées. Les 4 autres modèles se situent entre ces deux extrêmes : sur les jeux de données de films et de livres, DNN (0.54/0.66) est légèrement plus diversifié que le modèle TopPopular basé sur la popularité (0.49/0.60) et que les deux modèles hybrides KGE (0.51/0.59) et CBF-TopPopular (0.40/0.40).

Si l'objectif est simplement de maximiser la diversité des recommandations, alors le filtrage collaboratif traditionnel en général, et le modèle IBCF en particulier, semblent être des choix optimaux. Cependant, on peut se demander comment comparer ces niveaux de diversité absolue aux attentes ou besoins des utilisateurs.

TABLEAU 4.6 – Comparaison des performances des SdR en termes de la diversité.

Métrique	Jeu de données	CBF	TopPopular	CBF-TopPopular	IBCF	SVD	KGE	DNN
$\overline{\text{ILD}}$	MovieLens ($\overline{\text{ILD}_p}$: 0.59)	0.2761	0.4953	0.3988	**0.6998**	0.6623	0.5144	*0.5417*
	Anime ($\overline{\text{ILD}_p}$: 0.75)	0.5952	*0.7406*	0.7039	**0.8057**	0.7786	0.7389	0.7260
	LibraryThing ($\overline{\text{ILD}_p}$: 0.75)	0.3283	0.6019	0.4006	**0.8405**	0.7110	0.5901	*0.6576*
R^2(ILD, ILD_p)	MovieLens	0.0605	0.0065	0.0362	0.0001	0.0073	*0.0485*	**0.1030**
	Anime	**0.0542**	0.0428	0.0057	0.0005	0.0093	*0.0205*	0.0103
	LibraryThing	0.1870	0.0001	**0.3119**	0.0001	0.0001	*0.1342*	0.0126

Pour chaque jeu de données et chaque métrique, le meilleur score est en gras, le deuxième meilleur est souligné et le troisième est en italique.

Confronter la diversité des recommandations ($\overline{\text{ILD}}$) à celle observée au sein des profils utilisateurs ($\overline{\text{ILD}_p}$) permet de mesurer l'adéquation entre le niveau de diversité proposé par ces SdR et celui attendu par les utilisateurs. D'après le tableau 4.6, les modèles récents KGE et DNN semblent être les plus performants en termes de diversité *relative*. Le niveau de diversité intermédiaire de leurs listes de recommandations est plus cohérent avec la diversité observée dans les profils utilisateurs. Par exemple, sur le jeu de données Anime où l'$\overline{\text{ILD}_p}$ est d'environ 0.75, l'$\overline{\text{ILD}}$ est de ~ 0.74 / ~ 0.73 pour KGE / DNN, alors qu'il dépasse 0.80 pour IBCF et est inférieur à 0.60 pour CBF. Selon ces valeurs moyennes de métrique, KGE et DNN semblent avoir des performances raisonnables et être proches des attentes des utilisateurs en termes de diversité. La figure 4.2 permet de visualiser, pour tous les jeux de données considérés, la distribution des valeurs d'ILD de chaque SdR testé ainsi que la distribution des valeurs d'ILD_p (courbes rouges). Cette représentation visuelle confirme qu'IBCF (courbes jaunes) conduit à des recommandations sur-diversifiées alors que les modèles basés sur le contenu tels que CBF et CBF-TopPopular (courbes vertes et oranges) ont tendance à produire des recommandations sous-diversifiées. Les modèles les plus récents (KGE et DNN) conduisent à des distributions d'ILD plus proches de celles d'ILD_p.

Cependant, même si la distribution des valeurs d'ILD ainsi que la valeur moyenne $\overline{\text{ILD}}$, pour certains SdR, sont assez proches de ceux des ILD_p (notamment pour le jeu de données Anime), leurs recommandations les plus diversifiées ne semblent pas être proposées aux bons utilisateurs. En effet, le résultat le plus frappant ici est probablement la forte divergence entre la diversité des profils utilisateurs (ILD_p) et la diversité des recommandations qui leur sont faites par le SdR (ILD). Comme le montre le tableau 4.6, les valeurs de R^2 pour ces deux ensembles de valeurs (indiquant dans quelle mesure elles sont linéairement corrélées) sont assez faibles pour les 21 configurations (3 jeux de données × 7 SdR). Toutes les valeurs de R^2 sont inférieures à 0.2 (sauf pour CBF-TopPopular sur LibraryThing, qui est de ~ 0.31 mais reste encore très faible). Des valeurs de corrélation aussi faibles indiquent que même DNN et KGE ne parviennent pas à proposer le bon niveau de diversité aux bons utilisateurs. Ceci est clairement illustré par la figure 4.3 qui

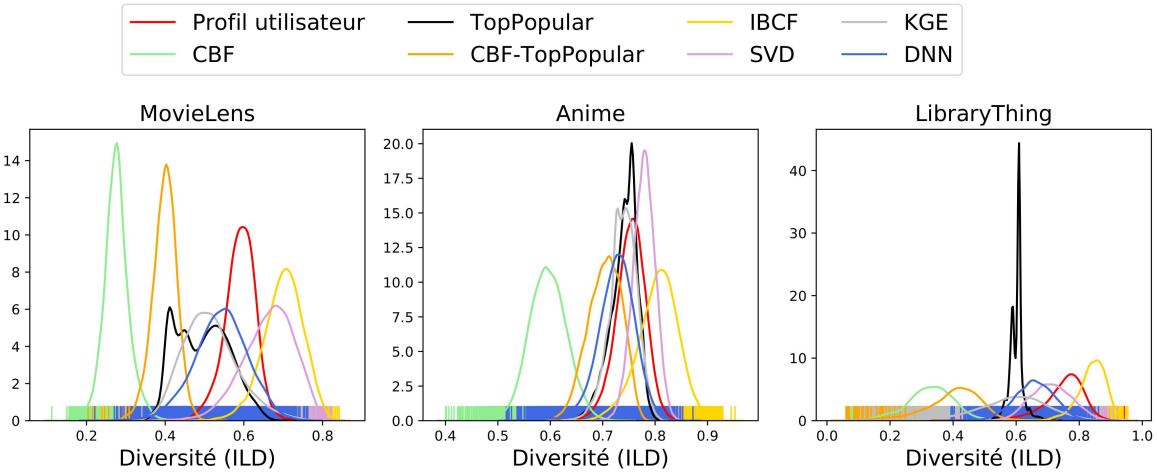


FIGURE 4.2 – Distributions des valeurs de diversité (ILD) fournies aux utilisateurs par différents SdR et des valeurs de diversités présentes dans les profils utilisateurs (ILD_p)

montre les nuages de points des valeurs d'ILD par rapport aux valeurs d'ILD_p.

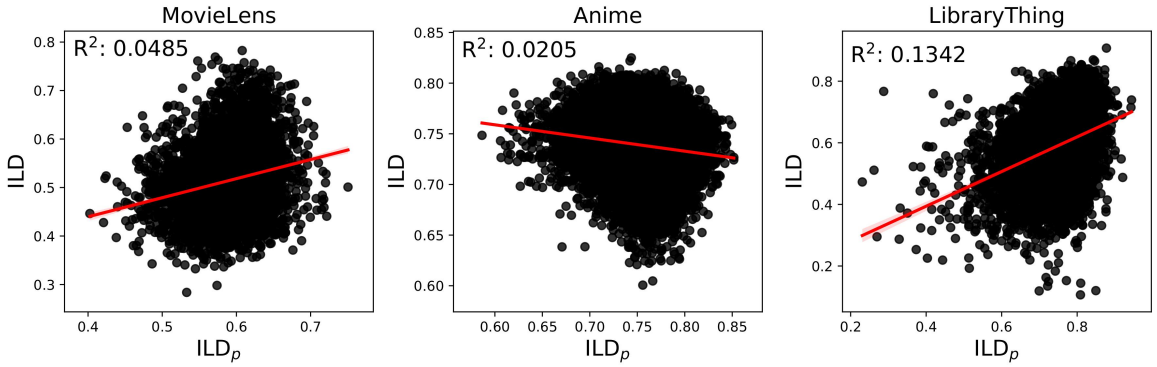


FIGURE 4.3 – Diagramme de dispersion des diversités au sein des recommandations par rapport aux diversités dans les profils utilisateurs (*i.e.*, ILD par rapport à ILD_p). Les recommandations sont basées sur le modèle KGE.

4.4.2 Impact du post-traitement de la diversification

Le post-traitement de diversification peut être considéré comme un processus de reclassement au cours duquel les items, initialement classés en fonction de leur score de pertinence, sont reclassés de sorte que les premiers items soient à la fois pertinents et diversifiés. Nous analysons ici les résultats de la méthode de post-traitement par l'heuristique gloutonne (cf. algorithme 1) pour deux fonctions objectifs différentes : la variante classique et la variante personnalisée, désignées respectivement par *glouton-cls* et *glouton-pers*. Les deux méthodes s'appuient sur un paramètre, appelé facteur de diversification α, pour ajuster le compromis entre pertinence (*accuracy*) et diversité.

Ces deux approches de post-traitement ont été testées en combinaison avec les sept modèles de recommandation sur les trois jeux de données considérés. Les figures 4.4, 4.5

et 4.7 fournissent respectivement, pour chacune de ces 42 combinaisons, une représentation graphique de l'évolution de la diversité *absolue* ($\overline{\text{ILD}}$), de la diversité *relative* (R^2) et de l'*accuracy* (F1-mesure) des recommandations, lorsque l'on augmente α. Notez que la seule mesure d'*accuracy* discutée dans cette sous-section est la F1-mesure, les mêmes tendances étant observées pour les autres mesures d'*accuracy*, c'est-à-dire la précision, le rappel et la MAP.

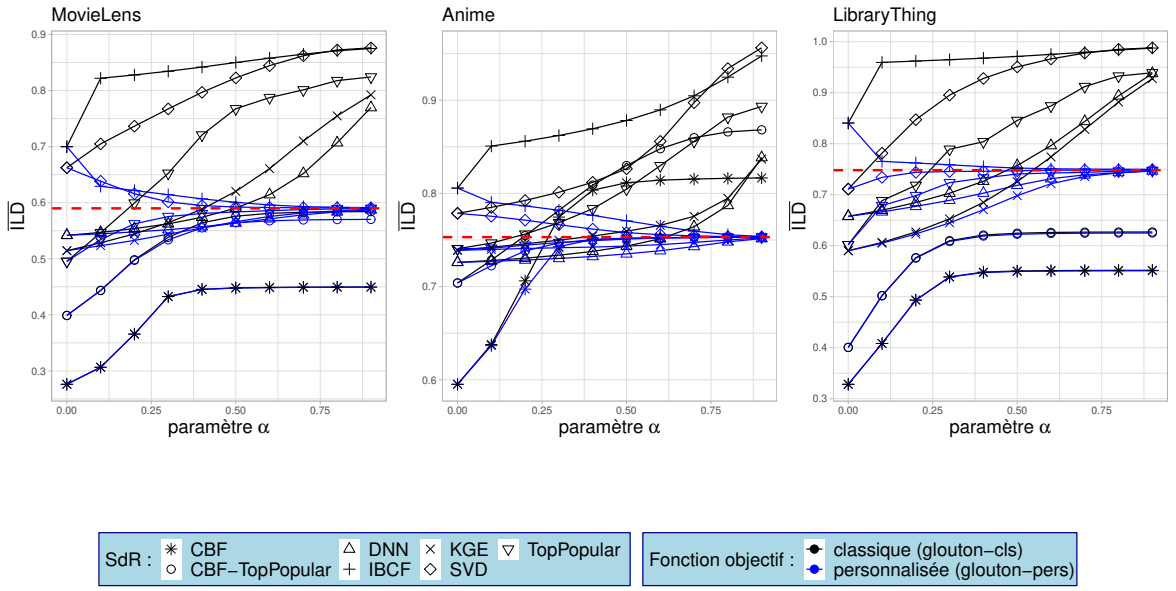


FIGURE 4.4 – Évaluation de la diversité *absolue* pour les fonctions objectifs classique et personnalisée dans le post-traitement glouton. Les lignes en pointillés rouges représentent la moyenne des diversités *désirées* par les utilisateurs, $\overline{\text{ILD}}_p$.

Comme prévu, le post-traitement *glouton-cls* augmente les niveaux d'$\overline{\text{ILD}}$ pour tous les SdR et tous les jeux de données. Plus les valeurs de α sont élevées, plus l'$\overline{\text{ILD}}$ résultant est grand (cf. figure 4.4). Les seules exceptions à cette règle sont observées pour les modèles de recommandation basés sur le contenu, c'est-à-dire CBF (∗) et CBF-TopPopular (∘), sur les jeux de données MovieLens ou LibraryThing où l'$\overline{\text{ILD}}$ atteint un plateau. Il semble que les SdR basés sur le contenu fournissent des listes de candidats (*i.e.* les 100 premiers items classés dans nos expérimentations) si peu diversifiées, que cela limite fortement le potentiel de diversification du post-traitement. En d'autres termes, les 100 premiers items renvoyés par les approches basées sur le contenu sont parfois si similaires qu'il n'y a aucun moyen d'en extraire un sous-ensemble hautement diversifié. Pour surmonter cette limite, il peut être utile d'augmenter le nombre des items candidats, comme nous le présenterons dans la prochaine sous-section. Dans tous les autres cas, lorsque le paramètre α de *glouton-cls* est augmenté, l'$\overline{\text{ILD}}$ est également largement augmenté, ignorant complètement les besoins de diversité des utilisateurs (*i.e.*, l'$\overline{\text{ILD}}_p$, représenté par les lignes pointillées rouges dans la figure 4.4). En revanche, lorsque le paramètre α de *glouton-pers* est augmenté, les valeurs d'$\overline{\text{ILD}}$ tendent vers les valeurs d'$\overline{\text{ILD}}_p$.

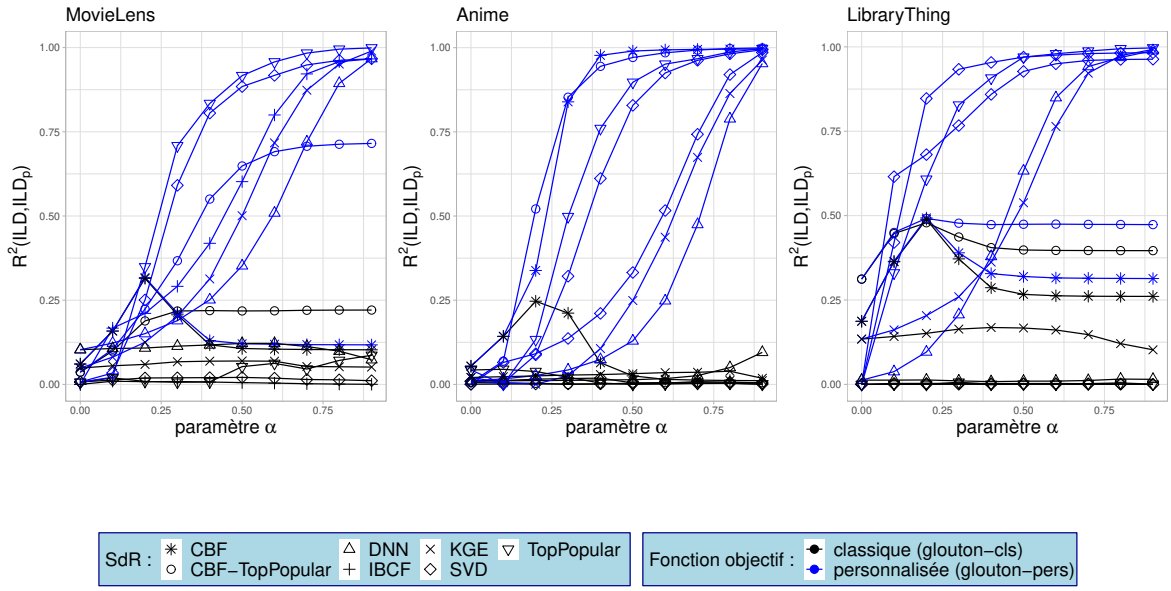


FIGURE 4.5 – Évaluation de la diversité *relative* pour les fonctions objectifs classique et personnalisée dans le post-traitement glouton.

Le graphique de la diversité relative (cf. figure 4.5) indique que, lors de l'utilisation de *glouton-pers*, l'augmentation de α améliore la cohérence des diversités observées dans les recommandations et dans les profils utilisateurs (les valeurs R^2 tendent vers 1). En revanche, en utilisant *glouton-cls*, l'augmentation de α conduit souvent à diminuer encore plus la corrélation entre la diversité du profil utilisateur et celle des recommandations qui lui sont proposées (les valeurs R^2 tendent vers 0). La différence entre ces deux post-traitements, concernant la corrélation entre les valeurs ILD et ILD$_p$, est clairement mise en évidence par la figure 4.6.

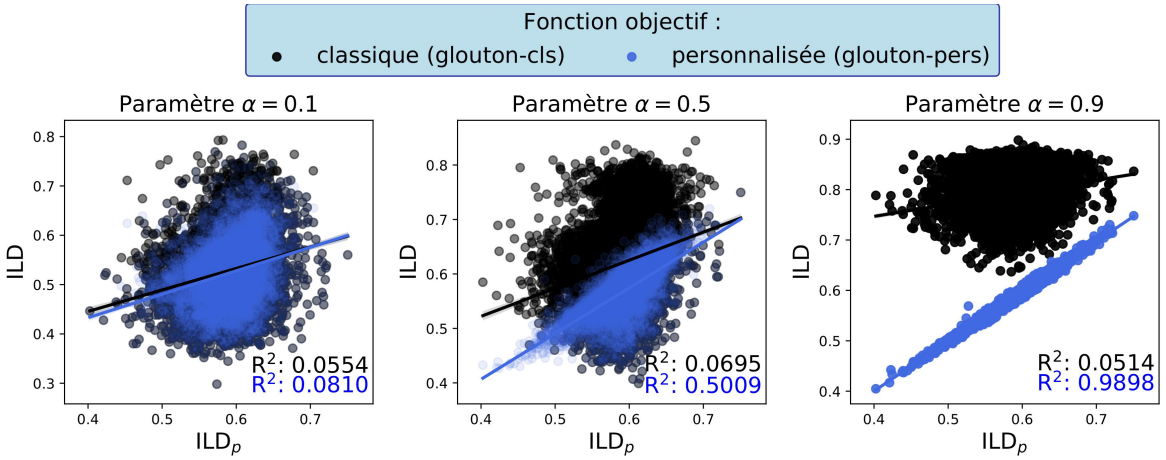


FIGURE 4.6 – Diagramme de dispersion des valeurs d'ILD (diversité recommandée) par rapport à celles d'ILD$_p$ (diversité du profil) pour trois valeurs de α avec deux fonctions objectifs différentes. Les recommandations sont basées sur le modèle KGE.

En ce qui concerne l'impact du paramètre α, les SdR basés sur le contenu ont, une

fois de plus, un comportement atypique. En effet, lorsqu'il est associé aux modèles CBF et CBF-TopPopular, le post-traitement *glouton-cls* permet également d'augmenter la métrique R^2 lorsque de faibles valeurs de α sont utilisées (*e.g.* $\alpha < 0.3$). Ceci est probablement dû à la faible diversité au sein de leurs recommandations de sorte que l'introduction d'une certaine diversité, grâce à un post-traitement (personnalisé ou non), ne présente que des avantages. En effet, CBF ($*$) et CBF-TopPopular ($\circ$) sont les deux seuls SdR testés pour lesquels il est possible d'améliorer simultanément la diversité *absolue* ($\overline{\text{ILD}}$), la diversité *relative* (R^2) et l'*accuracy* (F1-mesure). Une valeur de α proche de 0.25 semble optimale pour ces deux approches basées sur le contenu : la valeur d'$\overline{\text{ILD}}$ est bien meilleure qu'avec $\alpha = 0$ tandis que la F1-mesure est presque inchangée ($\alpha \in [0, 0.25]$) pour MovieLens et Anime et même améliorée pour LibraryThing. Pour tous les autres SdR, l'amélioration des mesures de diversité se fait toujours au prix d'une baisse d'*accuracy* (cf. figure 4.7). Cependant, l'optimisation *glouton-cls* entraîne une baisse de l'*accuracy* plus importante que l'optimisation *glouton-pers* (notamment lorsque α devient supérieur à 0.5). Le post-traitement basé sur *glouton-pers* tend à préserver des niveaux d'*accuracy* relativement élevés, même avec une valeur de α élevée, et ce sur les trois jeux de données.

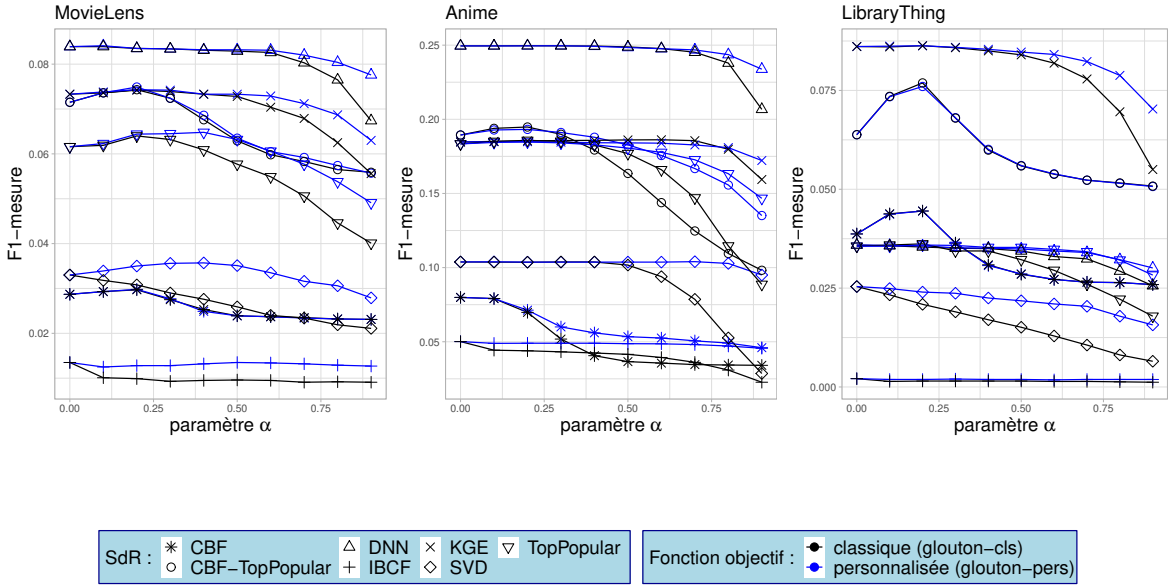


FIGURE 4.7 – Évaluation de l'*accuracy* pour les fonctions objectifs classique et personnalisée dans le post-traitement glouton.

4.4.3 Impact du nombre d'items candidats du post-traitement glouton sur la diversité et l'*accuracy*

Dans cette sous-section, nous présentons les résultats concernant l'impact du nombre m d'items candidats pris en compte lors le post-traitement glouton, sur l'*accuracy* et la diversité des recommandations.

Comme nous l'avons vu précédemment, le nombre m d'items candidats considérés lors de la phase d'optimisation gloutonne pourrait également être un moyen de gérer le compromis entre diversité et *accuracy* des systèmes de recommandation. Lorsque ce nombre d'items candidats augmente, le post-traitement offre, d'une part, plus de possibilités aux SdR pour trouver des items plus diversifiés mais, d'autre part, il peut également entraîner une baisse importante de pertinence. La figure 4.8 illustre un tel cas où l'augmentation du nombre d'items candidats permet aux approches CBF et CBF-TopPopular de recommander des listes d'items plus diversifiées mais moins pertinentes. Plus généralement, pour la plupart des SdR testés (notamment ceux basés sur le contenu, à savoir CBF et CBF-TopPopular), les valeurs d'$\overline{\text{ILD}}$ ont tendance à augmenter avec la taille des listes de candidats (m de 50 à 300), tant pour les fonctions objectifs classiques que personnalisées. Il est intéressant de noter que l'impact du paramètre m sur la diversité des recommandations semble être négligeable pour les modèles KGE et DNN, *i.e.* les deux SdR récents qui présentent les meilleures performances en termes d'*accuracy* (voir tableau 4.5). La figure 4.8 suggère que, pour ces deux modèles, l'ensemble des 50 items les mieux classés est déjà suffisamment diversifié et qu'il est donc inutile que les post-traitements de diversification considèrent les items moins bien classés. En termes de F-mesure, l'impact de m semble négligeable pour tous les SdR, à l'exception de ceux basés sur le contenu (CBF et CBF-TopPopular), pour lesquels la F1-mesure diminue suffisamment pour que cette baisse soit visuellement évidente sur le graphique.

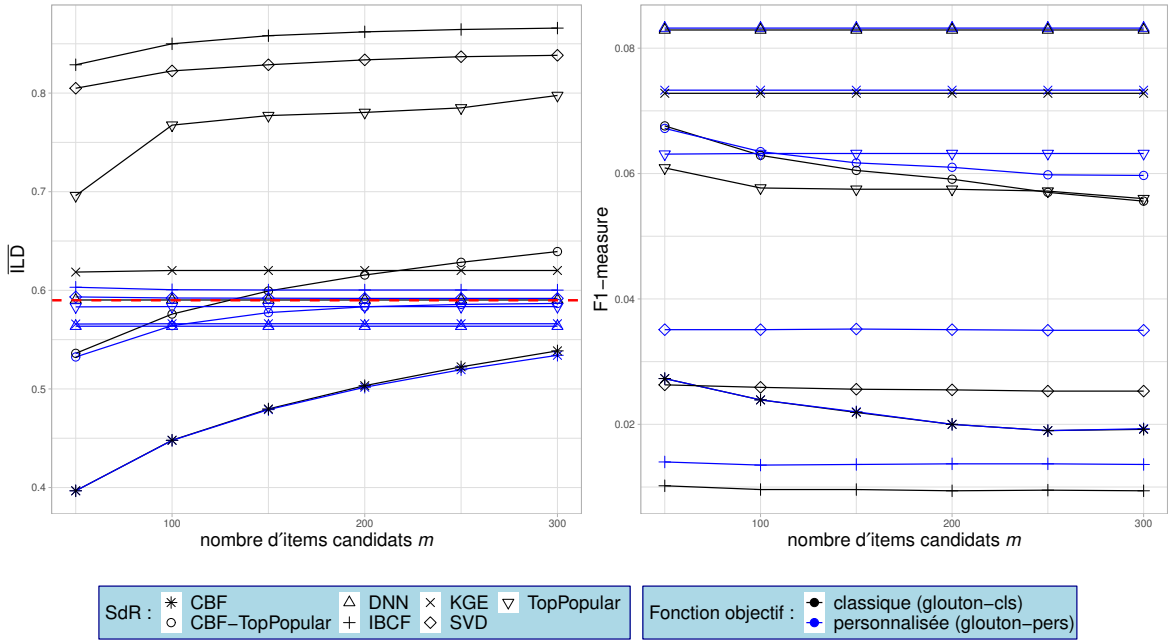


FIGURE 4.8 – *Accuracy* (F1-Mesure) et diversité absolue ($\overline{\text{ILD}}$) des SdR sur MovieLens pour différentes valeurs du nombre d'items candidats m considérés lors des post-traitements gloutons. Le facteur de diversification α est fixé à 0.5 pour les deux fonctions objectifs.

Globalement, l'impact du nombre m d'items candidats n'est pas aussi important que nous le pensions. Le choix de $m = 100$ (le carré du nombre de recommandations) que

nous avons fait pour notre étude semble tout à fait raisonnable car la figure 4.8 montre que les résultats de diversité (ILD) et de F1-mesure ne changent pas beaucoup à partir de 100 items candidats.

4.4.4 Limitation, discussion et implication

Dans cette étude, les besoins individuels des utilisateurs en termes de diversité sont estimés à partir des items contenus dans leur profil (c'est-à-dire l'ILD_p), ce qui peut présenter certaines limites.

Tout d'abord, les utilisateurs peuvent avoir des comportements d'évaluation différents : certains n'évaluent que quelques items alors que d'autres en notent beaucoup plus. Même si cette hétérogénéité du nombre d'annotations est partiellement gérée par le fait que la métrique ILD considère une valeur moyenne, il n'en reste pas moins que la fiabilité de l'estimation de cette valeur dépend du nombre d'items notés. On retrouve ici une problématique proche de celle traitée dans le chapitre sur EBCR. Une manière potentielle d'augmenter le nombre de données utilisées pour estimer la diversité attendue par l'utilisateur serait de considérer les *ratings* implicites. En effet, comme notre approche ignore les valeurs de *ratings* des utilisateurs pour estimer les ILD_p, elle se généralise naturellement à la prise en compte des *ratings* implicites [Aggarwal, 2016e]. La diversité attendue par un utilisateur serait alors l'ILD de la liste des items avec lesquels cet utilisateur a interagi (par exemple, les produits achetés sur un site marchand, les films regardés sur une plate-forme de *streaming*, etc.). Cela constitue généralement un ensemble d'items beaucoup plus large que les seuls items notés, et pourrait donc conduire à une estimation plus robuste du besoin en diversité propre à chaque utilisateur, notamment pour ceux ayant noté peu d'items.

Deuxièmement, suivant en cela les travaux de [Di Noia et al., 2014; Meymandpour and Davis, 2020], nous avons considéré tous les items notés par l'utilisateur pour estimer sa valeur d'ILD_p. Une autre alternative serait de se focaliser uniquement sur les items aimés par l'utilisateur (profil positif) pour la mesure de cette valeur. En effet, il est plus raisonnable pour les SdR de ne prendre en compte que les items aimés par un utilisateur dans le cas où l'objectif est de lui fournir des items qu'il pourrait aimer (autrement dit le critère de l'*accuracy*). Dans le contexte d'optimisation de la diversité, le fait de ne prendre en compte que ses items aimés peut conduire à une valeur d'ILD_p légèrement plus faible, car les items bien notés par l'utilisateur cible partagent souvent des propriétés sémantiques communes. Les faibles valeurs d'$\overline{\text{ILD}}$ de l'approche CBF le démontrent, car elle renvoie des items qui sont sémantiquement similaires à ceux qui sont très appréciés par l'utilisateur. D'autre part, considérer tous les items notés par les utilisateurs permet de prendre en compte un plus grand nombre d'items. On suppose ici que tant qu'un utilisateur a noté un item (même avec une note faible), celui-ci a attiré son attention, on peut alors supposer que l'item fait correspond plus ou moins a priori à ses goûts. Cependant,

cette proposition est loin d'être évidente et dépend probablement du comportement de chaque utilisateur en matière d'évaluation.

Le troisième point est que les besoins de diversité des utilisateurs peuvent également dépendre d'autres facteurs tels que leur humeur, leur personnalité, etc. et peuvent évoluer dans le temps. Par exemple, 62% des participants impliqués dans l'étude de [Yao and Harper, 2018] ont considéré l'humeur comme une caractéristique importante lors du choix du prochain film à regarder. Il pourrait être intéressant de prendre en compte ces facteurs lors de l'estimation des besoins en termes de diversité d'un utilisateur. Une autre limite importante de notre étude est qu'elle est fortement basée sur la qualité des connaissances des entités encodées dans le graphe de connaissances DBpedia, qui peut parfois contenir des faits très parcellaires sur certains items [Färber et al., 2018].

La présente étude est exclusivement basée sur des expérimentations hors-ligne et suppose que la métrique d'ILD basée sur la similarité sémantique peut capturer la perception des utilisateurs actifs quant à la diversité (également à la variation de la diversité) des items recommandés. Cette hypothèse, peut d'un premier abord sembler critiquable, mais elle semble, d'après les travaux de [Castagnos et al., 2013], être valable. Dans leur étude, les utilisateurs se sont vus présenter des programmes de télévision générés à partir de certains SdR classiques, y compris ceux basés sur le contenu et ceux basés sur le CF. La diversité des recommandations a été mesurée avec la métrique d'ILD basée sur la similarité sémantique en utilisant les attributs des items. Les auteurs de [Ekstrand et al., 2014] ont également rapporté les perceptions des utilisateurs actifs sur leur listes de films recommandés par rapport à différentes dimensions d'évaluation, par exemple l'*accuracy*, la diversité, la satisfaction, etc. Leur mesure de diversité est également basée sur la métrique d'ILD, la similarité des items est estimée à partir des vecteurs représentant la sémantique des items. Les utilisateurs ont été invités à communiquer leurs perceptions sur les listes de recommandations de trois SdR (IBCF, UBCF et SVD). Leurs résultats ont montré que le modèle IBCF fournissait des items plus diversifiés que SVD, ce qui est cohérent avec nos résultats (voir tableau 4.6). Ce constat suggère que les résultats de nos expérimentations hors-ligne devraient refléter efficacement les performances de diversité des SdR testés.

En termes d'implications de cette étude, les résultats indiquent que le niveau de diversité *absolue* des recommandations fournies par le modèle de filtrage collaboratif basé sur les items (IBCF) est toujours beaucoup plus élevé que celui observé au sein des profils utilisateurs. Ce type de SdR est pourtant encore largement utilisé dans les moteurs de recommandation de l'industrie (Amazon par exemple) et nos résultats fournissent une indication claire : il est contre-productif d'augmenter encore plus leur diversité *absolue* en les enchaînant avec un post-traitement de diversification classique de type glouton. En revanche, pour les approches basées sur le contenu, notre étude suggère qu'en général les méthodes d'optimisation de la diversité sont d'autant plus pertinentes que ces SdR produisent des recommandations ayant des faibles niveaux de diversité *absolue*. Les modèles de recommandation récents, tels que les modèles KGE et DNN testés, fournissent

des recommandations très pertinentes qui sont, en moyenne, suffisamment diversifiées. Cependant, ils ne sont pas aussi performants en termes de diversité *relative*, comme c'est également le cas pour tous les autres SdR testés. Aucun des SdR n'a été capable de recommander aux utilisateurs des listes d'items ayant des niveaux de diversité *relative* optimaux, *i.e.*, qui correspondent à leurs profils.

Comme la tendance actuelle de la communauté des systèmes de recommandation est axée sur les réseaux de neurones profonds et les *embeddings* des graphes [Ferrari Dacrema et al., 2019; Zhang et al., 2019; Xu et al., 2020], les résultats de notre étude pourraient pousser les chercheurs ou les développeurs de SdR à concevoir de nouveaux modèles qui tiennent compte des besoins des utilisateurs en termes de diversité, car ils mettent en évidence l'importante marge de progression qu'il reste pour proposer des recommandations ayant un niveau de diversité cohérent avec les attentes de chaque utilisateur. Nous espérons également que ce travail pourra encourager la mise à disposition de jeux de données de référence contenant des *feedbacks* implicites (comme la plate-forme *Deskdrop*[8]) afin d'obtenir des informations sur les items avec lesquels les utilisateurs ont interagi et pas seulement ceux qu'ils ont notés. Il s'agit d'une information cruciale car la plupart des utilisateurs regardent beaucoup plus de films ou lisent beaucoup plus de livres qu'ils n'en évaluent. Le recoupement de ces deux types d'information pourrait probablement permettre de mieux cerner les besoins des utilisateurs, notamment en ce qui concerne le critère de diversification.

4.5 Conclusion de chapitre

Les systèmes de recommandation sont depuis longtemps évalués en fonction de leur *accuracy*. L'importance de fournir des recommandations diversifiées est désormais également reconnue. Cependant, l'impact d'un post-traitement de diversification sur les principaux types de systèmes de recommandation n'avait jusqu'alors pas été étudié de manière approfondie. Cette étude présente une analyse et une comparaison systématique et complète des quatre principales familles de systèmes de recommandation (à savoir non-personnalisés, basés sur le contenu, basés sur le filtrage collaboratif et hybrides) en ce qui concerne l'*accuracy* et la diversité. Cette étude a été réalisée en utilisant trois jeux de données de différents domaines pour évaluer les systèmes de recommandation. Nous sommes convaincus que le fait de considérer uniquement le niveau absolu de diversité d'une liste de recommandations n'est pas suffisant et que la diversité de chaque liste de recommandations devrait également être évaluée par rapport à l'utilisateur auquel cette liste est destinée (et à ses besoins). Nous proposons d'utiliser la diversité des items notés par un utilisateur comme un indicateur de la diversification qu'il attend d'une liste de recommandations.

[8] https://www.kaggle.com/gspmoreira/articles-sharing-reading-from-cit-deskdrop

Les principales conclusions de notre étude sont les suivantes. Premièrement, différentes approches de SdR conduisent à des recommandations qui ont des niveaux de diversité très différents. Par conséquent, s'il peut sembler raisonnable d'augmenter la diversité pour certaines d'entre elles, d'autres nécessiteraient plutôt un post-traitement permettant d'atténuer leur tendance à la sur-diversification. Si l'on considère la moyenne des diversités des profils utilisateurs comme une référence, les approches basées sur le contenu semblent fournir des items sous-diversifiés, tandis que les approches classiques de filtrage collaboratif conduisent plutôt à des items sur-diversifiés. Des approches plus récentes, basées sur des réseaux de neurones profonds ou sur les *embeddings* de graphes de connaissances, fournissent des recommandations dont la diversité moyenne est beaucoup plus proche de celle des profils utilisateurs. Ces résultats étaient en quelque sorte attendus : recommander des items en utilisant uniquement leurs similarités ne laisse pas beaucoup de place à la diversification, tandis que recommander des items en se basant uniquement sur les suggestions des autres utilisateurs peut rapidement conduire à une liste *à la Prévert*.

Le deuxième résultat principal de ce chapitre est qu'aucun des SdR testés, même parmi les plus récents, ne semble fournir des recommandations dont le niveau de diversité est en adéquation avec les profils utilisateurs. Coupler ces SdR à un post-traitement de diversification classique rend les choses encore pires. Un tel post-traitement ne vise qu'à maximiser la diversité de la liste recommandée tout en favorisant autant que possible les items les mieux notés. Le résultat est une diminution de l'*accuracy* du SdR couplée à une augmentation de la diversité de la liste de recommandations, même si elle était déjà trop diversifiée. Ce type de post-traitement ne semble bénéfique que pour les approches de recommandation basées sur le contenu, car ce sont les seules qui conduisent à des recommandations réellement sous-diversifiées. Pour tous les autres modèles de recommandation testés, ce post-traitement *classique* diminue la pertinence des recommandations, éloigne la diversité moyenne des recommandations de la diversité moyenne des profils et diminue la corrélation entre les diversités des profils utilisateurs et celles des recommandations.

Le troisième résultat principal de cette étude est qu'un post-traitement de diversification alternatif mérite d'être étudié. Nous avons proposé une variante simple du post-traitement classique de la diversification. Elle vise à construire une liste de recommandations dont la diversité est aussi proche que possible de celle du profil de l'utilisateur. En d'autres termes, au lieu de maximiser un niveau absolu de diversité, il vise à maximiser l'adéquation entre la diversité du profil utilisateur et celle des recommandations faites à l'utilisateur tout en maintenant un niveau d'*accuracy* acceptable. Ce simple changement de perspective permet d'améliorer radicalement les choses. Pour tous les SdR et jeux de données testés, l'utilisation de ce post-traitement personnalisé permet de rapprocher la diversité moyenne des recommandations de celle des profils utilisateurs et d'augmenter la corrélation entre les diversités des profils et les diversités des items recommandés tout

en diminuant beaucoup moins l'*accuracy* que lorsque le post-traitement de diversification classique est utilisé.

L'augmentation de la diversité des recommandations via un post-traitement est généralement perçue comme bénéfique. Cette étude démontre qu'un tel post-traitement peut en fait être préjudiciable pour la plupart des approches de recommandation pour lesquelles le niveau de diversité est déjà proche, voire supérieur, à la diversité observée dans les profils des utilisateurs. Cette étude propose une solution simple pour prendre en compte la spécificité des utilisateurs lors de l'ajustement du niveau de diversité de leurs recommandations. Cette approche simple s'avère pertinente et nous espérons qu'elle incitera les chercheurs à travailler sur ce sujet afin que les systèmes de recommandation puissent mieux répondre aux attentes des utilisateurs tant en termes d'*accuracy* que de diversité.

Chapitre 5

Recommandation plus transparente ? Mieux comprendre la recommandation à l'aide des graphes de connaissances

Sommaire

5.1 Introduction et contexte 128

5.1.1 Explications basées sur les modèles 128

5.1.2 Explications indépendantes des modèles (*post-hoc*) 130

5.1.3 Explications post-hoc basées sur les données liées 131

5.2 Motivations 134

5.3 Modèle d'explication basé sur les propriétés (PEM) 136

5.3.1 Définitions et notations 137

5.3.2 Fonction de score des propriétés 137

5.3.3 Calcul efficace des scores à l'aide de l'extraction de sous-ontologie 138

5.3.4 Éviter la redondance totale entre les propriétés sélectionnées 141

5.3.5 La méthode PEM 142

5.4 Protocole d'évaluation 143

5.4.1 Jeu de données *MovieTweetings* et alignements dans DBpedia . . . 143

5.4.2 Modèle de recommandation 144

5.4.3 Métriques d'évaluation 145

5.4.3.1 Métriques hors-ligne 145

5.4.3.2 Métriques en-ligne 146

5.4.4 Étude réalisée avec des utilisateurs 146

5.5 Résultats 147

5.5.1 Efficacité de l'optimisation de la sous-ontologie 147
5.5.2 Comparaison des performances sur l'étude d'utilisateurs 148
5.5.3 Un exemple d'étude de cas 149
5.6 Conclusion de chapitre 150

5.1 Introduction et contexte

- « Les clients ayant consulté cet article ont également regardé l'article X ».
- « Les personnes suggérées pour vous : M. X (6 relations en commun) ».
- « Vous aimez écouter du Jazz ? Nous vous recommandons : l'album X ».

Les phrases ci-dessus sont des exemples d'explication que nous pouvons obtenir des systèmes de recommandation réels (*e.g.*, Amazon, LinkedIn, Spotify). Ces exemples permettent d'illustrer clairement l'intérêt des explications accompagnant les recommandations : permettre à l'utilisateur de comprendre pourquoi un item (X) a été recommandé, l'aider à prendre une décision concernant un achat éventuel, une mise en relation interpersonnelle ou un moment d'écoute musicale et ainsi augmenter sa confiance dans le système [Berkovsky et al., 2017].

Les systèmes de recommandation (notamment les plus récents) ont la capacité de prédire, de manière très précise, les appréciations des utilisateurs pour les items candidats. Pourtant, malgré leur efficacité, le fait de n'afficher que les listes de recommandations (sans aucune justification) peut empêcher les utilisateurs de prendre les bonnes décisions sur la base des recommandations qui leur sont faites [Gedikli et al., 2014]. Tandis que de plus en plus d'aspects de notre vie quotidienne sont influencés par des décisions automatisées prises ou suggérées par des systèmes de recommandation, il devient naturel de se demander si ces systèmes sont dignes de confiance, compte tenu notamment de l'opacité et de la complexité de leur fonctionnement interne [Sacharidis, 2020]. Une attention particulière a été portée ces dernières années sur les explications produites par les systèmes de recommandation et ces dernières sont rapidement devenues une fonctionnalité centrale de ces systèmes, incitant les utilisateurs à considérer plus avant les items recommandés [Tintarev and Masthoff, 2012; Lops et al., 2019; Zhang and Chen, 2020].

Dans la littérature, les approches d'explication des recommandations peuvent être divisées en deux catégories : les approches basées sur les modèles et celles indépendantes des modèles, dites *post-hoc* [Zhang and Chen, 2020].

5.1.1 Explications basées sur les modèles

Les approches appartenant à cette catégorie visent à étudier le fonctionnement du processus de recommandation afin de rendre le modèle sous-jacent plus interprétable et

transparent. Néanmoins, la capacité à fournir des recommandations explicables dépend fortement des modèles de recommandation eux-mêmes, ce qui confère à certains modèles de SdR des avantages sur ce point par rapport à d'autres. A l'époque du développement des premiers SdR, il était assez courant d'expliquer les recommandations à partir de règles d'association [Lin et al., 2000] ou par un filtrage collaboratif basé sur le voisinage (NBCF) [Herlocker et al., 2000]. Le modèle de règles d'association vise à découvrir les *patterns* pertinents (*i.e.* des relations intéressantes et non triviales entre différentes variables) dans les données, *e.g.*, un client achetant du pain et du lait simultanément, est susceptible d'acheter des œufs. Quant au filtrage collaboratif basé sur le voisinage, la probabilité qu'un item soit aimé par un utilisateur est considérée élevée si les utilisateurs similaires ont également apprécié cet item. Nous constatons ainsi que ces modèles de recommandation possèdent des avantages en termes de capacité d'explicabilité puisqu'ils sont naturellement compréhensibles et interprétables par les humains.

Cependant, les approches plus récentes et modernes, comme par exemple les modèles de factorisation matricielle [Koren and Bell, 2015] et ceux basés sur les réseaux de neurones profonds [He et al., 2017; Ferrari Dacrema et al., 2019], bien que performants en termes d'*accuracy*, font intervenir des espaces latents opaques, ce qui rend difficile leur interprétation. Ces modèles de recommandation manquent ainsi de transparence. Pour pallier ce problème, des informations supplémentaires, telles que les avis ou les commentaires des utilisateurs, sont généralement utilisées. Par exemple, les auteurs de [Zhang et al., 2014] ont proposé la méthode *Explicit Factor Model (EFM)*, qui aligne chaque dimension latente de la matrice factorisée (générée par une méthode de réduction de dimensionnalité, *e.g.* SVD) avec une caractéristique explicite des items, extraite à partir des commentaires des utilisateurs. Ainsi, EFM peut fournir des explications personnalisées pour les produits recommandés, par exemple : « *Le produit vous est recommandé parce que vous êtes intéressé par {une caractéristique particulière}, et parce que le produit est performant sur cette caractéristique* ».

Mis à part les commentaires des utilisateurs, les connaissances, *i.e.* les *faits* liés aux utilisateurs et aux items, représentent également une source de données pertinente pour fournir des explications. Les auteurs de [Ai et al., 2018] proposent un modèle de recommandation explicable basé sur les *embeddings* de graphes de connaissances. A partir des produits proposés par Amazon, les auteurs peuplent le graphe de connaissances en rajoutant des relations entre les utilisateurs, les items et les éléments du graphe de connaissance initial, permettant ainsi d'encoder l'achat d'un item par un utilisateur et les caractéristiques de cet item. Les modèles d'apprentissage des *embeddings* (*e.g.*, TransE) sont entraînés sur ce graphe pour obtenir les représentations vectorielles de chaque utilisateur, item, entité et relation, et des recommandations sont fournies à un utilisateur en trouvant l'item le plus proche via la relation "acheter" dans l'espace d'*embeddings*. En outre, des explications peuvent être fournies en trouvant le chemin le plus court entre l'utilisateur et l'item recommandé à travers le graphe de connaissances.

5.1.2 Explications indépendantes des modèles (*post-hoc*)

Contrairement aux modèles d'explication basés sur les *modèles*, les approches d'explication *post-hoc* ne sont pas associées aux modèles à partir desquels les recommandations sont générées et ne produisent les explications qu'après que les items aient été recommandés. Dans ce cas, les systèmes de recommandation sont considérés comme des boîtes noires (*black boxes*). Il existe principalement deux manières de générer les explications *post-hoc* : non-personnalisée et personnalisée.

En ce qui concerne les *explications post-hoc non-personnalisées*, elles ne se focalisent que sur les items recommandés, *i.e.* ceux à expliquer, tout en ignorant les intérêts propres des utilisateurs. Autrement dit, les explications associées à un item sont toujours les mêmes quel que soit l'utilisateur auquel elles sont destinées. Cette absence de personnalisation permet de pré-calculer l'explication pour chaque item du catalogue et d'éviter ainsi les problèmes de temps de calcul des explications personnalisées. Les auteurs de [Tintarev and Masthoff, 2012] ont discuté et évalué différents styles d'explication non-personnalisée telle que par exemple l'explication basée sur la popularité des items – « On vous recommande ce film parce que c'est un film très populaire » ; l'explication basée sur les informations statistiques des items – « On vous recommande ce film parce que 80% de nos utilisateurs aiment ce film. » ; l'explication basée sur les caractéristiques décrivant les items – « On vous recommande ce film parce que c'est une comédie française réalisée par Dany Boon ». Les auteurs de [Musto et al., 2020] ont proposé une autre approche d'explication post-hoc non-personnalisée qui résume les avis positifs des utilisateurs sur les items. Les auteurs ont tiré parti du traitement automatique du langage naturel et de l'analyse des sentiments pour extraire des caractéristiques pertinentes pour les explications. Néanmoins, on ne dispose pas toujours d'un nombre suffisant de commentaires utilisateurs pour pouvoir mettre en œuvre cette stratégie. En effet, si l'on dispose de nombreux commentaires pour des films, ce n'est pas le cas pour d'autres types d'items (*e.g.*, articles scientifiques, podcasts radio). De plus, la quantité de commentaires disponibles peut également varier fortement entre les items d'un même catalogue (*e.g.*, un blockbuster américain par rapport à un film d'art et d'essai suédois).

A l'inverse, les *explications post-hoc personnalisées* prennent en compte les intérêts propres des utilisateurs, qui sont souvent représentés par les items appréciés explicitement par les utilisateurs, *i.e.* le profil utilisateur. Dans ce cas, on se focalise sur la découverte des relations partagées, des points en commun, entre les items aimés par les utilisateurs et ceux recommandés. Par exemple, « Ce film vous est recommandé parce que vous avez aimé des comédies ».

Ainsi, nous pouvons constater que les connaissances liées aux items constituent de réels atouts pour les explications post-hoc, qu'elles soient personnalisées ou non. Dans ce qui suit, nous nous focalisons sur les approches d'explication post-hoc basées sur le Web sémantique et les données liées.

5.1.3 Explications post-hoc basées sur les données liées

L'initiative des données liées, *i.e. Linked Open Data* (LOD), a facilité le développement à grande échelle de graphes de connaissances (KG) populaires, comme DBpedia, qui encodent des faits et des connaissances hétérogènes dans une structure standardisée et interprétable par les machines. D'une part, comme introduit dans les chapitres 2 et 4, les données liées et les graphes de connaissances possèdent des atouts réels pour les systèmes de recommandation, *e.g.* l'amélioration de l'*accuracy* des recommandations [Di Noia et al., 2012b; Musto et al., 2017; Palumbo et al., 2020]. D'autre part, ces connaissances riches relatives aux items peuvent contribuer à l'amélioration de l'explication des recommandations [Musto et al., 2016b; Lully et al., 2018; Musto et al., 2019], comme nous allons le discuter dans cette section.

Les auteurs de [Lully et al., 2018] ont proposé un modèle pour améliorer les explications des recommandations dans le domaine du tourisme, qui est basé sur les entités extraites de DBpedia. Pour une visite recommandée, les auteurs ont comparé quatre styles d'explication différents, notés respectivement *EN* (seules les entités sont présentées aux utilisateurs), *NL* (des phrases en langage naturel contenant des entités sont présentées), *PC* (seuls les concepts directement associés aux entités sont présentés, *e.g. Architectural Styles*) et *CC* (les entités sont présentées accompagnées de leurs classes de rattachement), *e.g. Art Nouveau (Architectural Styles)*. Se basant sur les résultats issus d'une étude en-ligne auprès de 30 participants, les auteurs ont constaté que les styles *NL* et *CC* étaient plus performants que les styles *EN* et *PC*. Dans [Lully et al., 2018], les explications sont *non-personnalisées* car elles se concentrent uniquement sur les descriptions (entités, classes etc.) des visites recommandées, tout en ignorant les préférences utilisateurs. Dans [Musto et al., 2016b], les auteurs ont proposé un modèle d'explication personnalisée nommé ExpLOD, qui exploite le graphe de connaissances DBpedia pour générer les explications post-hoc. L'approche que nous présentons et discutons dans les sections suivantes est largement inspirée du modèle ExpLOD. Dans la suite de cette section nous présentons ExpLOD et ses variantes, dans la section 5.2 nous détaillons ses limites qui ont motivé nos travaux et enfin nous présentons notre approche dans la section 5.3.

Étant donné l'utilisateur u, on note P_u l'ensemble de propriétés d'objet[1] qui décrivent les items du profil utilisateur I_u (*i.e.*, items aimés par l'utilisateur u) et P_r l'ensemble des propriétés décrivant ses items recommandés I_r. ExpLOD repose sur l'ensemble $p \in \mathrm{P}$ avec $\mathrm{P} = \mathrm{P}_u \bigcap \mathrm{P}_r$. La fonction de *score* pour une propriété donnée $p \in \mathrm{P}$ concernant I_u et I_r est définie par l'équation (5.1), où n_{p,I_u} (resp. n_{p,I_r}) représente le nombre d'items dans I_u (resp. I_r) qui sont décrits par la propriété p, α et β sont des facteurs de pondération, et

[1] En ingénierie des connaissances, la terminologie « propriété d'objet » (*object property*) fait référence à une relation entre groupe d'individus, *e.g.*, "dct:subject" qui associe une catégorie à un film. Pour éviter toute ambiguïté et par abus de langage, dans le reste du chapitre, le terme « propriété » désigne l'instance associée plutôt que la propriété elle-même (relation), *e.g.*, pour caractériser "dbr:Cool_Runnings" la propriété associée est "dbc:American_sports_comedy_films" et non pas "dct:subject".

le terme IDF(p) représente l'Inverse de la Fréquence de la propriété p dans le Document (catalogue). Ce dernier terme vise à pénaliser les propriétés décrivant de nombreux items du catalogue, *e.g.*, *American films*[2].

$$score(p, \mathrm{I}_u, \mathrm{I}_r) = \left(\alpha \frac{n_{p,\mathrm{I}_u}}{|\mathrm{I}_u|} + \beta \frac{n_{p,\mathrm{I}_r}}{|\mathrm{I}_r|}\right) * \mathrm{IDF}(p) \tag{5.1}$$

Une fois ces scores calculés pour l'ensemble des propriétés de P, les k propriétés ayant les scores les plus élevés (top-k) sont utilisées pour générer des explications en langage naturel basées sur un *template* prédéfini. Par exemple, l'explication du film recommandé *Memento* à un utilisateur qui a aimé *Sherlock Holmes* et *Le Prestige*, peut être (avec k = 2) : « Nous vous recommandons *Memento* parce que vous avez aimé *2000s mystery films* comme *Sherlock Holmes* et *Christopher Nolan directed films* comme *Le Prestige* ».

La fonction de *score* favorise les propriétés qui sont fréquentes dans I_u et I_r tout en étant rares dans le catalogue d'items. La part favorisant les propriétés apparaissant plusieurs fois dans la liste recommandée (*i.e.*, $\frac{n_{p,\mathrm{I}_r}}{|\mathrm{I}_r|}$) est une spécificité de la méthode ExpLOD. En effet, ce modèle a été conçu pour fournir des explications compactes pour un groupe d'items recommandés. L'attribution d'un bonus aux propriétés décrivant plusieurs items dans I_r permet de favoriser les explications qui s'appuient sur les mêmes propriétés pour différents items recommandés. Les explications sont ensuite factorisées par des phrases telles que : « Nous vous recommandons les films f_1 et f_2 car ce sont des *Romantic comedy films* comme la plupart de ceux que vous avez aimés ». La contrepartie de ce choix est que l'explication fournie pour un item $i_r \in \mathrm{I}_r$ n'est pas nécessairement aussi bonne qu'elle aurait pu l'être si i_r avait été le seul item devant être expliqué (*i.e.*, si $\mathrm{I}_r = \{i_r\}$). Il peut même arriver qu'aucune des top-k propriétés du groupe d'items I_r ne soit pertinente pour expliquer certains items de I_r, autrement dit, les top-k propriétés ne couvrent qu'une partie des items de I_r. De ce fait, plus $|\mathrm{I}_r|$ est grand, et diversifié, plus le risque que certains items ne soient pas (ou mal) expliqués est élevé.

Les auteurs ont étendu l'approche ExpLOD dans [Musto et al., 2019] en considérant un ensemble de propriétés *plus générales* (dites *broader properties*) P_b, c'est-à-dire des propriétés subsumant[3] directement celles qui décrivent (annotent) les items du profil utilisateur (P_u) ou les items recommandés (P_r). Plus formellement,

$$\mathrm{P}_b = \{p | parent(p, p_r) \wedge parent(p, p_u), p_u \in \mathrm{P}_u, p_r \in \mathrm{P}_r\}.$$

Par exemple, l'ensemble de propriétés *broader* qui subsument directement la propriété *2000 mystery films* (*i.e.*, les propriétés *parents* de cette dernière) contient des entités comme *2000 films*, *mystery films by decade* et *21st-century mystery films*. Les scores de ces pro-

[2] Notons ici que nous utilisons dans le texte le label anglais associé a la propriété *dbc:American_films* pour la représenter, le label français n'étant pas disponible dans DBpedia. Pour éviter toute ambiguïté nous employons par la suite les labels anglais en italique pour représenter les propriétés

[3] On dit qu'une propriété p_b *subsume* une propriété p s'il existe une relation de généralisation entre les deux propriétés, *e.g.*, "skos:broader" ou "rdf:type".

priétés *broader* $p_b \in \mathrm{P}_b$ sont calculés en additionnant les scores des propriétés p qu'elles subsument (cf. équation (5.2)). Les propriétés *broader* p_b les mieux classées sont ensuite utilisées pour générer l'explication.

$$score(p_b, \mathrm{I}_u, \mathrm{I}_r) = \sum_{p \in enfants(p_b) \cap (\mathrm{P}_u \cup \mathrm{P}_r)} score(p, \mathrm{I}_u, \mathrm{I}_r) * \mathrm{IDF}(p_b) \tag{5.2}$$

Notez que le même item peut contribuer plusieurs fois au score de p_b s'il est décrit par plusieurs enfants de la propriété p_b. Par exemple, considérons un profil utilisateur comprenant deux *films set in USA*; l'un est décrit comme *film set in New York* alors que l'autre est décrit à la fois comme *film set in New York* et *film set in Washington*; le deuxième film aura alors plus d'impact sur le score de la propriété *broader* "*films set in USA*" que le premier du fait de sa double contribution à la somme de l'équation (5.2). Notez également que le score de la propriété *broader* p_b ne prend pas seulement en compte sa valeur d'IDF(p_b) mais aussi les valeurs d'IDF(p) d'un sous-ensemble de ses enfants p, ce qui peut sembler discutable, en effet l'équation (5.2) peut être réécrite comme suit :

$$score(p_b, \mathrm{I}_u, \mathrm{I}_r) = \sum_{p \in enfants(p_b) \cap (\mathrm{P}_u \cup \mathrm{P}_r)} \left(\alpha \frac{n_{p,\mathrm{I}_u}}{|\mathrm{I}_u|} + \beta \frac{n_{p,\mathrm{I}_r}}{|\mathrm{I}_r|} \right) * \mathrm{IDF}(p) * \mathrm{IDF}(p_b) \tag{5.3}$$

Les auteurs ont évalué ExpLOD (versions *basic* et *broader*) en réalisant une étude en ligne auprès de 680 utilisateurs sur trois jeux de données relatifs à différents domaines de recommandation (film, musique et livre). Le modèle proposé a obtenu de bons résultats en termes de *transparence*, *persuasion*, *engagement* et *confiance*, qui sont des mesures classiques d'évaluation des explications de recommandation [Tintarev and Masthoff, 2012; Zhang and Chen, 2020]. De plus, il semble que ExpLOD permette de générer des explications post-hoc efficaces, quel que soit le modèle de SdR utilisé. Enfin, leurs évaluations ont également confirmé la supériorité de l'extension (version *broader*) de ExpLOD et la pertinence d'utiliser des propriétés plus générales que celles directement associées aux items des profils et des listes de recommandations.

Cette dernière observation est la principale motivation de notre travail qui vise à améliorer la variante *broader* du modèle ExpLOD en prenant en compte efficacement toute la hiérarchie des propriétés. Bien que l'approche que nous proposons soit générique, nous nous concentrerons ci-après sur la hiérarchie des catégories de DBpedia afin de fournir au lecteur un cadre familier. Cette hiérarchie est extraite à partir de l'ontologie associée au graphe de connaissances de DBpedia, en se restreignant aux relations de subsomption entre les catégories, et plus particulièrement la relation "skos:broader".

Les principales contributions de l'approche présentée dans ce qui suit sont les suivantes :

- nous prenons en compte toute la hiérarchie des propriétés avec un faible temps de calcul grâce à des optimisations algorithmiques reposant sur l'extraction de sous-ontologies [Ranwez et al., 2011].

- nous limitons les propriétés non pertinentes dans les explications en favorisant celles qui annotent (décrivent) directement les items du catalogue.
- nous proposons un critère simple pour éliminer les explications totalement redondantes.
- nous proposons une fonction de score non-paramétrique pour classer les propriétés.

Le reste du chapitre est organisé comme suit. Dans la section 5.2, nous présentons nos motivations de recherche avant de détailler l'approche proposée dans la section 5.3. Le protocole d'évaluation est détaillé en section 5.4. Les résultats d'une étude en-ligne réalisée auprès de 155 utilisateurs sont détaillés en section 5.5. Enfin dans la section 5.6, nous présentons les conclusions de ce chapitre.

5.2 Motivations

Les expériences réalisées par les auteurs de [Musto et al., 2019] pour comparer les variantes d'ExpLOD (*basic* et *broader*) ont montré l'intérêt de prendre en compte les propriétés plus générales (*i.e.*, celles subsumant d'autres propriétés). Cependant, il n'y a pas de raison évidente de ne restreindre cet élargissement qu'aux seules subsomptions directes (comme fait dans ExpLOD *broader*). En même temps, pour que cet élargissement à des propriétés plus générales soit bénéfique il faut : i) éviter d'utiliser des propriétés trop abstraites et non-pertinentes pour fournir une explication ; ii) limiter la redondance dans l'explication ; et iii) maintenir un temps de calcul faible.

Un item explicitement décrit par la propriété p est aussi implicitement décrit par toutes les propriétés qui subsument p. La plupart des graphes de connaissances possèdent *a minima* une hiérarchie de propriétés définie par une relation de subsomption transitive telle que "rdf:type" (*i.e.*, *isA*), "dbo:isPartOf" ou "skos:broader" qui constitue la *colonne vertébrale* du graphe. Il s'agit, en effet, d'une caractéristique essentielle des graphes de connaissances (ontologies) qui permet les raisonnements sémantiques, et sans laquelle la plupart des ontologies seraient vides de sens. La figure 5.1 illustre l'importance de la hiérarchie des propriétés à travers des exemples simples inspirés des catégories de films de DBpedia. Dans tous ces exemples, le but est d'expliquer un film recommandé i_r basé sur un ensemble de films aimés par l'utilisateur I_u. Si i_r est décrit par la propriété *French romantic comedy* (représentée par l'étoile verte) et que certains items de I_u (représentés par le triangle rouge) sont également décrits par cette propriété, celle-ci peut être utilisée pour expliquer la recommandation, comme illustré dans la figure 5.1A. Il s'agit de l'idée au cœur de la version *basic* d'ExpLOD. Cependant, cette version de base échouerait lorsque i_r est annoté par *French romantic comedy* alors que I_u contient quelques films n'étant décrits que par la propriété *Italian romantic comedy* (cf.

figure 5.1B). La variante *broader* d'ExpLOD résout ce problème tout en maintenant un temps de calcul raisonnable, puisqu'elle ne considère que quelques propriétés qui subsument directement chaque propriété d'annotation. Cela permet, dans l'exemple de la figure 5.1B, d'expliquer le film recommandé en utilisant la propriété *European romantic comedy* qui subsume directement les propriétés *Italian romantic comedy* et *French romantic comedy*.

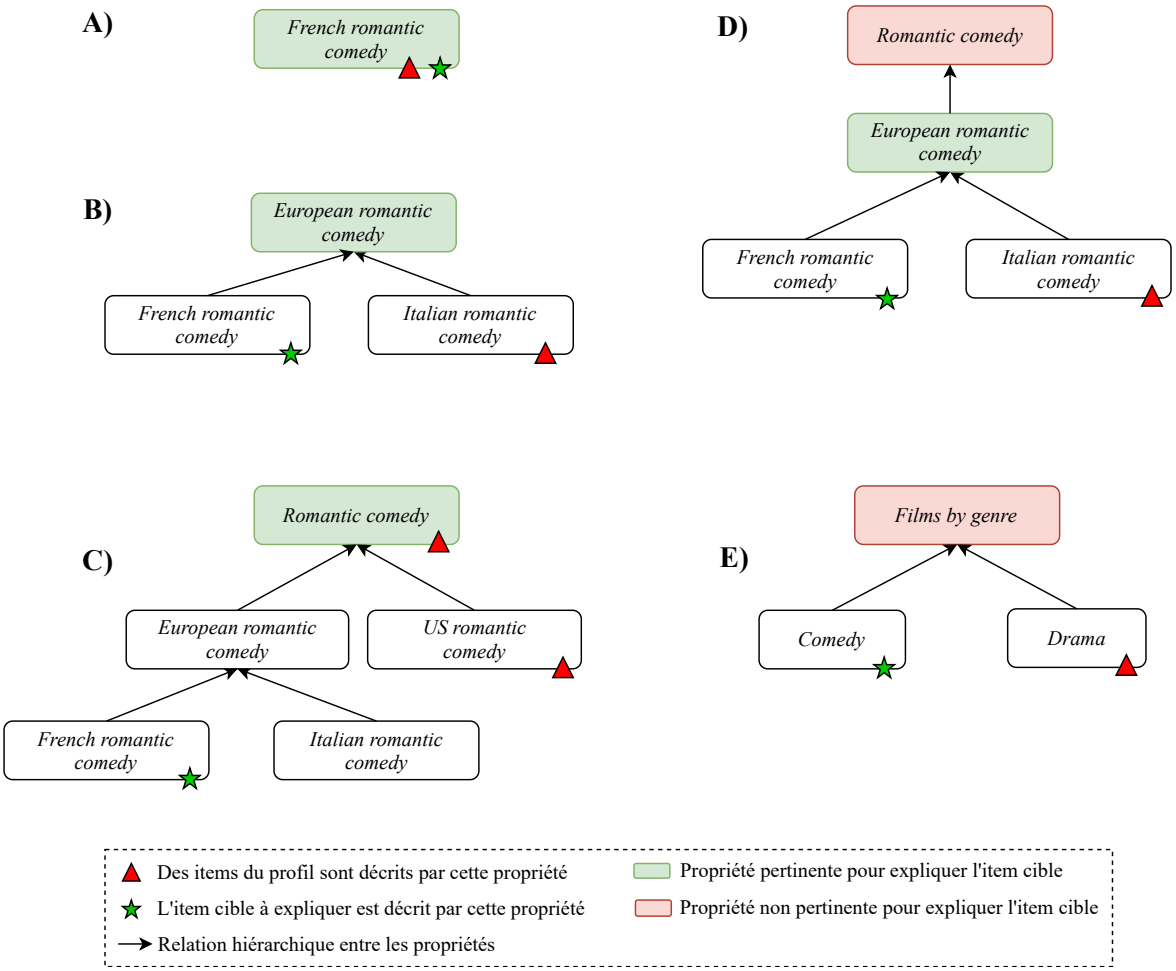


FIGURE 5.1 – Exemples illustrant l'importance de considérer la hiérarchie des propriétés dans le processus d'explication afin d'identifier les relations sémantiques entre les propriétés qui annotent des items du profil de l'utilisateur et celles qui annotent les items qui lui sont recommandés.

Notez également que les deux variantes d'ExpLOD proposées dans [Musto et al., 2016b, 2019] sont considérées par leurs auteurs comme deux approches distinctes puisqu'elles reposent sur des fonctions de score différentes. Par conséquent, la variante *basic* ne prend en compte que les propriétés en commun (*i.e.* $p \in \mathrm{P}$), alors que la variante *broader* d'ExpLOD ne prend en compte que les propriétés p_b, plus générales, qui subsument directement les propriétés explicites (*i.e.*, $p \in \mathrm{P}_u \cup \mathrm{P}_r$). Ainsi, la propriété *French romantic comedy* de la figure 5.1A sera prise en compte par la version *basic* mais pas par la variante *broader*, tandis que la propriété *European romantic comedy* de la figure 5.1B sera prise en compte par la variante *broader* d'ExpLOD mais pas par la version de base.

Bien que la variante *broader* constitue clairement une amélioration significative, les trois derniers exemples (cf. figures 5.1C, 5.1D et 5.1E) illustrent ses limites et les nouveaux problèmes qu'elle soulève. Dans la figure 5.1C, i_r est annoté comme *French romantic comedy* tandis qu'un des items de I_u est annoté comme *US romantic comedy* et l'autre comme *Romantic comedy*. Le problème ici est que la propriété subsumant directement *French romantic comedy* est *European romantic comedy*; de ce fait même la version *broader* d'ExpLOD ne parvient pas à déduire que les trois films sont des *Romantic comedy* et sera donc incapable d'utiliser cette propriété clé pour expliquer la recommandation. Notez également que toutes les propriétés qui subsument *Romantic comedy*, comme par exemple *Comedy*, bien que partagées (implicitement) par les annotations de ces trois films, ne seraient pas pertinentes pour l'explication car elles induiraient une redondance avec l'explication fournie en utilisant la propriété plus spécifique *Romantic comedy*. La figure 5.1D fournit un exemple de ce problème de redondance. Pour mesurer l'importance de cette remarque, il faut savoir que dans DBpedia, la propriété *Romantic comedy* possède environ 1 710 ancêtres (*i.e.*, les propriétés qui la subsument), dont *Romantic* ou *Creative works*, qui annoteraient tous (implicitement) les deux items de la figure 5.1D. Il semble clairement plus raisonnable de baser l'explication sur la propriété plus spécifique (*European romantic comedy*) que sur ses ancêtres, *e.g.*, *Creative works*, qui n'apporte pas d'information supplémentaire pour expliquer le choix d'un film, tous les films étant de type *Creative works*. Enfin, la prise en compte des propriétés *broader* (*i.e.*, P_b) dans la variante *broader* d'ExpLOD soulève un autre problème : certaines de ces propriétés *broader* peuvent être pertinentes pour l'organisation de la hiérarchie des propriétés mais possèdent peu de signification pour l'explication, comme l'illustre dans notre dernier exemple, la figure 5.1E. Ici, i_r est annoté comme *Comedy*, tandis qu'un item dans I_u est annoté comme *Drama*. Puisque ces deux propriétés sont directement subsumées par la propriété *Films by genre*, l'explication quelque peu surréaliste suivante pourrait être proposée par la variante *broader* d'ExpLOD : « Ce film vous est recommandé parce que c'est un *Films by genre* et que vous aimez habituellement les *Films by genres* ».

Dans ce chapitre, nous proposons un modèle générique d'explication basé sur les propriétés que nous nommons PEM (*Property-based Explanation Model*), qui exploite efficacement toute la hiérarchie des propriétés de l'ontologie (ou du graphe de connaissances). La section suivante présente l'approche proposée et détaille la manière dont nous exploitons cette hiérarchie complète en un temps de calcul raisonnable tout en fournissant des explications qui évitent la redondance et les propriétés ayant peu de signification.

5.3 Modèle d'explication basé sur les propriétés (PEM)

Nous détaillons dans cette section l'approche proposée pour exploiter efficacement la hiérarchie complète des propriétés lors de la construction post-hoc des explications de recommandation.

5.3.1 Définitions et notations

Tout d'abord, nous désignons par $\mathscr{C}$ l'ensemble des items du catalogue et par I_u l'ensemble des items aimés par l'utilisateur u (*i.e.*, son profil utilisateur). L'item recommandé qui est à expliquer (item cible) est désigné par i_r. Nous désignons ensuite par $\mathscr{H}(\mathscr{N},\mathscr{E})$ la hiérarchie des catégories de DBpedia, avec $\mathscr{N}$ l'ensemble des entités DBpedia préfixées par "dbr:Category" et $\mathscr{E}$ la relation "skos:broader" entre ces entités. $\mathscr{P}(i)$ représente l'ensemble des propriétés qui annotent directement l'item i, *i.e.*, $\mathscr{P}(i) = \{p | p \in \mathscr{N}$ et $\langle i, \text{"dct:subject"}, p\rangle\}$. Le prédicat "dct:subject"[4] est couramment utilisé dans DBpedia pour relier une ressource à ses catégories, *e.g.*, relier un film (i) aux propriétés désignant son genre (propriété p annotant i). Par extension, nous définissons la fonction $\mathscr{P}(\cdot)$ sur un ensemble d'items I comme $\mathscr{P}(\mathrm{I}) = \bigcup_{i\in\mathrm{I}} \mathscr{P}(i)$. Par exemple $\mathscr{P}(\mathscr{C})$ est l'ensemble des propriétés explicitement utilisées pour annoter au moins un item du catalogue $\mathscr{C}$.

Deuxièmement, nous désignons par $\overline{\mathscr{P}(i,\mathscr{H})}$ l'ensemble de toutes les propriétés qui *décrivent* directement ou indirectement l'item i dans la hiérarchie $\mathscr{H}$, *i. e.*, $\overline{\mathscr{P}(i,\mathscr{H})} = \{p | \exists p_x$ tel que p subsume p_x dans $\mathscr{H}$ et $p_x \in \mathscr{P}(i)\}$. De nouveau, nous étendons cette notation à un ensemble d'items en utilisant l'union : $\overline{\mathscr{P}(\mathrm{I},\mathscr{H})} = \bigcup_{i\in\mathrm{I}} \overline{\mathscr{P}(i,\mathscr{H})}$. Par souci de simplicité, nous omettrons le paramètre $\mathscr{H}$ et utiliserons $\overline{\mathscr{P}(i)}$ et $\overline{\mathscr{P}(\mathrm{I})}$ s'il n'y a aucune ambiguïté concernant la hiérarchie utilisée. De façon similaire, nous désignons par $\mathscr{I}(p,\mathrm{I})$ (resp. $\overline{\mathscr{I}(p,\mathrm{I})}$) le sous-ensemble des items de I directement (resp. directement ou indirectement) annotés par la propriété p. Par exemple, $\mathscr{I}(p,\mathrm{I}_u)$ désigne l'ensemble des items du profil utilisateur I_u qui sont annotés directement par la propriété p.

5.3.2 Fonction de score des propriétés

Avec la méthode PEM, nous cherchons à prendre en compte toute la hiérarchie des propriétés tout en évitant les propriétés non pertinentes qui ne sont là que pour organiser la hiérarchie des propriétés (*e.g.*, *Films by genre*). Pour cela, nous considérons toutes les propriétés p qui sont simultanément : i) une propriété subsumant des propriétés de i_r ii) une propriété subsumant des propriétés des items de I_u et iii) une propriété qui annote directement au moins un item du catalogue, *i.e.*, $p \in \overline{\mathscr{P}(i_r)} \bigcap \overline{\mathscr{P}(\mathrm{I}_u)} \bigcap \mathscr{P}(\mathscr{C})$. De ce fait, nous prenons en compte toute la hiérarchie des propriétés et traitons ainsi les potentiels problèmes illustrés par les figures 5.1B et 5.1C. L'avantage de cette intersection est qu'il n'est plus nécessaire de calculer le score de toutes les propriétés de la hiérarchie mais seulement de celles annotant directement au moins un item du catalogue considéré, ce qui nous permet de proposer une solution à faible latence tout en considérant toute la hiérarchie. Le *score* des propriétés dans cette intersection est défini de sorte à favoriser les propriétés qui sont à la fois fréquemment utilisées pour annoter directement les items du catalogue et sur-représentées dans le profil utilisateur, comme détaillé dans l'équation

[4] https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms/subject

(5.4).

$$score(p, \mathrm{I}_u, i_r) = \log_{10}\left(|\mathscr{I}(p,\mathscr{C})|\right) * \left(\frac{\left|\overline{\mathscr{I}(p,\mathrm{I}_u)}\right| / |\mathrm{I}_u|}{\left|\overline{\mathscr{I}(p,\mathscr{C})}\right| / |\mathscr{C}|}\right) \tag{5.4}$$

Plus précisément, pour chaque propriété candidate p, nous introduisons un facteur, $\log_{10}\left(|\mathscr{I}(p,\mathscr{C})|\right)$, qui pénalise fortement le score de p lorsque p est rarement utilisé pour décrire directement des items du catalogue. Les scores des propriétés annotant directement moins de 10 items seront réduits tandis que les scores des propriétés annotant directement plus de 10 items seront augmentés (sans faire beaucoup de différence entre les propriétés annotant directement 300 ou 320 items, grâce à la transformation logarithmique). L'avantage de ce facteur est qu'il permet d'éviter les propriétés qui ne sont pertinentes que pour organiser la hiérarchie mais pas pour décrire les items, *i.e.*, le problème illustré dans la figure 5.1E. Enfin, pour toutes les propriétés à considérer, la fonction de score que nous avons proposée est obtenue en combinant ce facteur $\log_{10}\left(|\mathscr{I}(p,\mathscr{C})|\right)$ avec un ratio de *fold-change*. La métrique de *fold-change* est largement utilisée pour détecter les propriétés sur-représentées dans un sous-ensemble d'éléments [Love et al., 2014]. Ici, nous l'utilisons pour détecter les propriétés de l'item recommandé sur-représentées dans le profil utilisateur par rapport à l'ensemble du catalogue d'items. Notez que dans ce ratio, une propriété est considérée comme décrivant un item si elle-même ou l'un de ses descendants est utilisé pour décrire l'item. Les informations de la hiérarchie complète sont donc prises en compte pour déduire les descriptions implicites des items.

5.3.3 Calcul efficace des scores à l'aide de l'extraction de sous-ontologie

Dans les graphes de connaissances de grande échelle tels que DBpedia, une propriété peut avoir des milliers d'ancêtres (*e.g.*, *Romantic comedy* possède environ 1 700 ancêtres). Ainsi, même si un item n'est, en général, annoté directement que par une poignée de propriétés, il est de ce fait implicitement décrit par des milliers de propriétés *ancêtres* de celles-ci. La figure 5.2A présente un exemple *jouet* où les annotations directes des items sont représentées par une forme géométrique colorée (une forme par item du catalogue) et les descriptions implicites sont représentées par la forme géométrique blanche correspondante. Même dans ce petit exemple, inspiré de la hiérarchie des propriétés de DBpedia, il est évident qu'il y a beaucoup plus de propriétés implicites que de propriétés directes. La prise en compte de ces descriptions implicites est donc essentielle pour tirer le meilleur parti des données liées, mais elle pose un défi algorithmique car le calcul des scores doit pouvoir passer à l'échelle (*scalability*) pour être réalisé sans introduire trop d'attente pour l'utilisateur.

Le calcul du score que nous proposons ici assure la *scalability* grâce à deux considérations clés. Premièrement, seules les propriétés annotant directement au moins un item du catalogue ont un score supérieur à 0 et méritent donc d'être évaluées. En pratique, cela

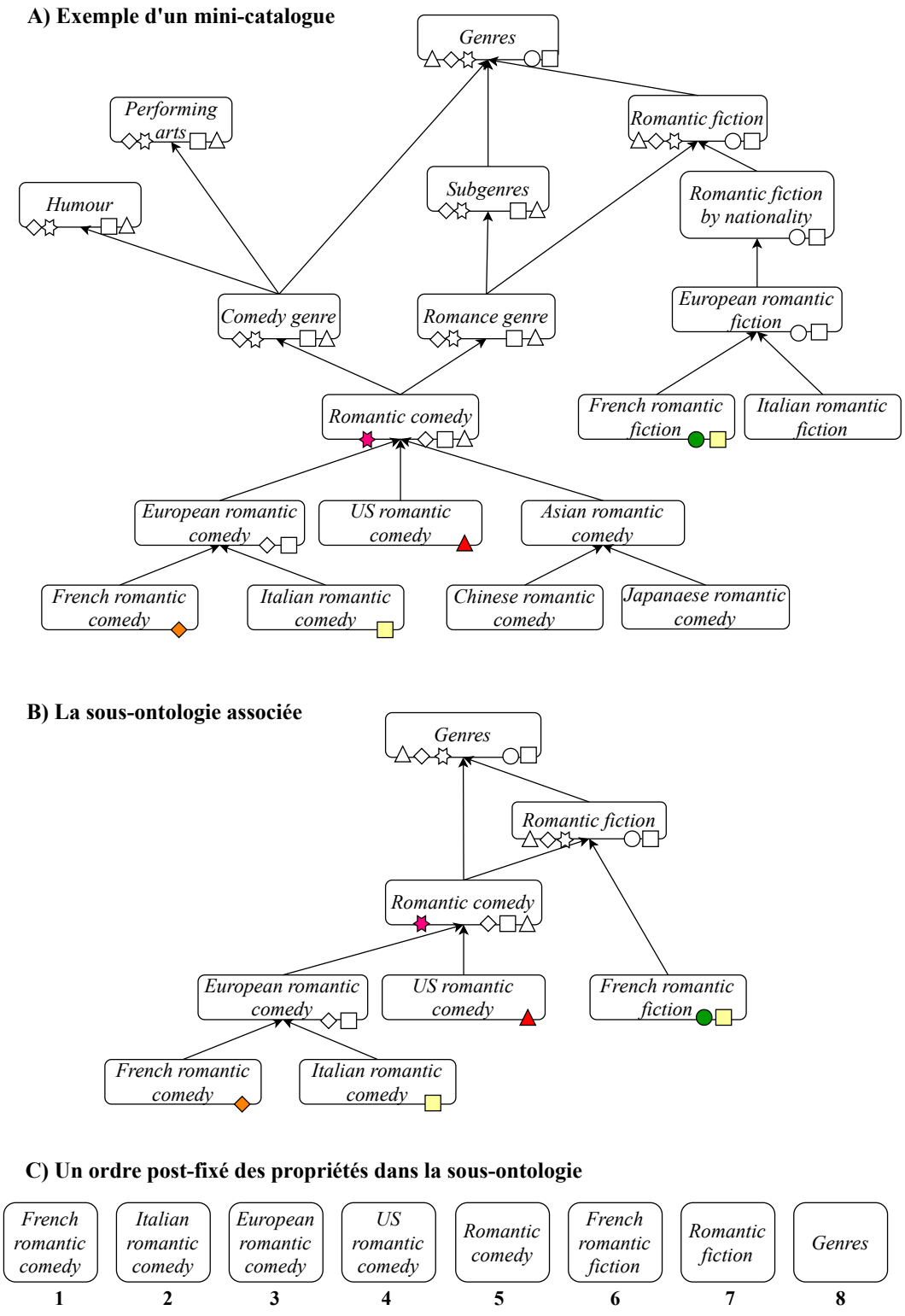


FIGURE 5.2 – Exemple *jouet* illustrant le processus d'extraction de sous-ontologie : A) fournit une mini hiérarchie de propriétés (inspirée de DBpedia) ; B) fournit la sous-ontologie associée et C) un post-ordonnancement des propriétés de cette dernière, qui est au centre de nos optimisations algorithmiques.

réduit considérablement le nombre de propriétés pour lesquelles le score doit être évalué. Calculer le score de chaque propriété indépendamment est inefficace car cela nécessiterait de considérer plusieurs fois les mêmes propriétés. Par exemple, pour estimer la cardi-

nalité de $\overline{\mathscr{I}(\textit{Romantic comedy}, \mathscr{C})}$ il faut, entre autres, identifier tous les *French romantic comedy* du catalogue, ce qui doit également être fait pour calculer $\overline{\mathscr{I}(\textit{French romantic comedy}, \mathscr{C})}$ et $\overline{\mathscr{I}(\textit{European romantic comedy}, \mathscr{C})}$. En supposant que le catalogue des items soit relativement stable, on pourrait argumenter que ces valeurs peuvent être pré-calculées une seule fois et ne nécessitent qu'une mise à jour lorsque le catalogue change. Cependant, un tel argument n'est pas valable pour les termes liés au profils utilisateurs, *e.g.*, $\overline{\mathscr{I}(p, \mathrm{I}_u)}$ ou $\left(\overline{\mathscr{P}(i_r)} \cap \overline{\mathscr{P}(\mathrm{I}_u)}\right)$, qui nécessitent une mise à jour permanente.

Comme $\mathscr{H}$ encode les relations de subsomption, il constitue un graphe orienté acyclique – ou *Directed Acyclic Graph* (DAG). L'absence de cycles permet d'ordonner les nœuds de manière cohérente, de sorte qu'un nœud apparaît toujours après tous ses descendants (ordre post-fixé) ou après tous ses ascendants (ordre pré-fixé). La figure 5.2B représente un extrait de hiérarchie de propriétés, ainsi qu'un ordonnancement possible des propriétés de cette dernière selon un ordre post-fixé (cf. figure 5.2C). L'utilisation d'un parcours par ordre post-fixé des propriétés dans le DAG permet de factoriser efficacement le calcul de $\overline{\mathscr{I}(p, \mathrm{I}_u)}$ et de $\overline{\mathscr{I}(p, \mathscr{C})}$. En effet, pour un ensemble d'items quelconque I, tous les $\overline{\mathscr{I}(p, \mathrm{I})}$ peuvent être obtenus en traitant toutes les propriétés dans l'ordre post-fixé et en utilisant la formule récursive suivante :

$$\overline{\mathscr{I}(p, \mathrm{I})} = \mathscr{I}(p, \mathrm{I}) \bigcup \left(\bigcup_{p_c \in \textit{enfants}(p)} \overline{\mathscr{I}(p_c, \mathrm{I})} \right) \tag{5.5}$$

Notez que pour notre fonction de score (cf. équation (5.4)), seules les tailles des ensembles sont nécessaires ; les ensembles réels ne sont utiles que temporairement pour éviter de compter plusieurs fois le même item. Par conséquent, une fois que tous les pères d'une propriété p ont été traités, l'ensemble réel $\mathscr{I}(p, \mathrm{I})$ peut être supprimé pour libérer la mémoire tant que sa cardinalité est conservée. Cela peut être fait efficacement en assignant un compteur à chaque propriété qui est initialement fixé au nombre de ses parents directs et diminué chaque fois que l'un d'entre eux est traité (le lecteur intéressé peut se référer à la section 4.2.3 de [Ranwez et al., 2011] pour plus de détails sur cette optimisation).

Jusqu'à présent, nous avons mentionné deux solutions pour optimiser le calcul des scores : la première consiste à calculer les scores uniquement pour les propriétés annotant directement au moins un item du catalogue (*i.e.*, celles dans $\mathscr{P}(\mathscr{C})$) et la seconde consiste à utiliser la formule récursive (équation (5.5)). Le problème est qu'elles semblent s'exclure mutuellement puisqu'il n'y a aucune garantie qu'un enfant des propriétés annotant directement des items soit également une annotation directe des items. Cependant, ces deux idées d'optimisation peuvent être combinées grâce à l'algorithme d'extraction de sous-ontologie décrit dans [Ranwez et al., 2011]. La figure 5.2 illustre la construction de $\mathscr{H}_{\mathscr{C}}$ sur un exemple simple. Extraire de $\mathscr{H}$ le sous-DAG induit par l'ensemble de propriétés $\mathscr{P}(\mathscr{C})$ permet d'obtenir un graphe $\mathscr{H}_{\mathscr{C}}$ beaucoup plus petit. Cette sous-ontologie est, par définition, le plus petit sous-DAG contenant toutes les propriétés de

$\mathscr{P}(\mathscr{C})$ ainsi que tous les plus petits ancêtres communs (*least common ancestors*) de tout sous-ensemble des propriétés conservées, ce qui est utile pour factoriser efficacement les calculs de $\overline{\mathscr{I}(p,\mathrm{I}_u)}$ et de $\overline{\mathscr{I}(p,\mathscr{C})}$. De plus, notez que les propriétés de $\mathscr{H}_\mathscr{C}$ qui ne sont pas dans $\mathscr{P}(\mathscr{C})$ ne sont utiles, dans notre cas, que si elles ont un ancêtre dans $\mathscr{P}(\mathscr{C})$. Par conséquent, il est possible de réduire encore plus $\mathscr{H}_\mathscr{C}$ en élaguant toutes ses propriétés qui ne sont pas dans $\mathscr{P}(\mathscr{C})$ et qui n'ont pas d'ancêtre dans $\mathscr{P}(\mathscr{C})$. Considérons la figure 5.2B comme un exemple. Alors que la propriété *Romantic fiction* est utile pour saisir la relation existante entre *Romantic comedy* et *French romantic fiction* et est donc conservée par l'extraction de sous-ontologie, elle n'est pas utile dans notre cas puisque ni elle-même ni son seul ancêtre (*i.e.*, *Genres*) n'annotent directement un item du catalogue fictif. Ces deux propriétés peuvent donc être supprimées de $\mathscr{H}_\mathscr{C}$ en toute sécurité sans que notre algorithme n'en soit affecté. Au contraire, *European romantic comedy*, bien que ne figurant pas dans l'ensemble $\mathscr{P}(\mathscr{C})$ sur cet exemple, doit être conservée afin d'évaluer efficacement le score de *Romantic comedy* (*i.e.*, une propriété qui annote directement certains items et peut donc être pertinente pour l'explication). L'identification des propriétés qui peuvent, ou non, être supprimées en toute sécurité (de $\mathscr{H}_\mathscr{C}$) peut se faire en traitant toutes les propriétés selon l'ordre pré-fixé à l'aide de la formule récursive suivante :

$$PasSupprimable(p,\mathscr{C}) = \left(p \in \mathscr{P}(\mathscr{C})\right) \text{ or } \left(\exists p_p \in parents(p) | PasSupprimable(p_p,\mathscr{C})\right) \tag{5.6}$$

5.3.4 Éviter la redondance totale entre les propriétés sélectionnées

Précisons tout d'abord ce que nous désignons par *redondance* et *redondance totale* parmi les propriétés candidates pour une explication. Nous disons qu'une propriété p est *redondante* par rapport à p_1, si et seulement si, p subsume p_1 dans $\mathscr{H}$. Par exemple, *Romantic comedy* est redondante par rapport à *French romantic comedy*. Cependant, il pourrait être intéressant d'avoir en même temps ces deux propriétés pour expliquer une recommandation qui est de type *French romantic comedy* si le profil utilisateur contient peu de *French romantic comedy* et beaucoup de *Romantic comedy*. En effet, dans un tel cas, il n'est pas trivial de comparer les deux explications : serait-il plus significatif pour l'utilisateur d'expliquer la recommandation en utilisant « nous vous recommandons ce film parce que c'est une *French romantic comedy* comme 3 des films que vous avez aimés » ou en utilisant « nous vous recommandons ce film parce que c'est une *Romantic comedy* comme 8 des films que vous avez aimés » ? D'un autre côté, si toutes les *Romantic comedy* du profil utilisateur sont françaises, alors la propriété *Romantic comedy* est *totalement redondante* par rapport à la propriété *French romantic comedy* et son utilisation dans l'explication n'a pas de sens puisqu'elle n'apporte aucune nouvelle information. Plus formellement, nous disons qu'une propriété p est *totalement redondante* par rapport à un profil utili-

sateur I_u et une propriété p_1 si, et seulement si, p subsume p_1 dans la hiérarchie $\mathscr{H}$ et $\left|\overline{\mathscr{I}(p,I_u)}\right| = \left|\overline{\mathscr{I}(p_1,I_u)}\right|$. Notons en revanche que si p_1 n'annote directement aucun item du catalogue (*i.e.*, $p_1 \notin \mathscr{P}(\mathscr{C})$) alors p_1 ne sera jamais utilisé dans une explication et p peut donc être utilisé sans craindre la redondance avec p_1. Pour identifier la redondance totale, nous devons connaître le nombre maximum d'items du profil utilisateur annotés par une propriété subsumée par p et qui annote directement au moins un item du catalogue :

$$M_u(p,I_u) = \max_{p_1 \in \{descendants(p) \cap \mathscr{P}(\mathscr{C})\}} \left|\overline{\mathscr{I}(p_1,I_u)}\right|.$$

Toute propriété p telle que $M_u(p,I_u) = \left|\overline{\mathscr{I}(p,I_u)}\right|$ est exclue pour les explications car cette égalité implique que p est totalement redondante par rapport à une propriété pertinente et plus spécifique. Les valeurs de $M_u(p,I_u)$ peuvent être calculées efficacement en utilisant un parcours de $\mathscr{H}_{\mathscr{C}}$ par ordre post-fixé à l'aide de la formule récursive suivante :

$$M_u(p,I_u) = \begin{cases} max\left(m_u(p,I_u), \left|\overline{\mathscr{I}(p,I_u)}\right|\right) & si\ p \in \mathscr{P}(\mathscr{C}) \\ m_u(p,I_u) & sinon \end{cases} \tag{5.7}$$

avec

$$m_u(p,I_u) = \max_{p_c \in enfants(p)} M_u(p_c,I_u) \tag{5.8}$$

5.3.5 La méthode PEM

La méthode PEM que nous proposons fournit une solution efficace pour identifier et classer les propriétés qui peuvent être utilisées pour expliquer un item recommandé par rapport à un profil utilisateur tout en considérant la hiérarchie entière des propriétés. Les premières étapes de la méthode PEM pré-calculent les différentes valeurs qui ne dépendent que du catalogue d'items $\mathscr{C}$ et du DAG, *i.e.*, la hiérarchie de propriétés $\mathscr{H}$ utilisée pour décrire les items du catalogue, à savoir :

1. identifier l'ensemble des propriétés pertinentes $\mathscr{P}(\mathscr{C})$.
2. construire la sous-ontologie, sous-DAG $\mathscr{H}_{\mathscr{C}}$.
3. pour toutes les propriétés $p \in \mathscr{P}(\mathscr{C})$, calculer le nombre d'items du catalogue directement, $|\mathscr{I}(p,\mathscr{C})|$, ou indirectement, $|\overline{\mathscr{I}(p,\mathscr{C})}|$, décrits par p.

Ensuite, chaque fois qu'une explication doit être fournie pour un item i_r et un profil utilisateur I_u, les étapes suivantes sont exécutées :

1. compter le nombre d'items de I_u qui sont directement, $|\mathscr{I}(p,I_u)|$, ou indirectement, $|\overline{\mathscr{I}(p,I_u)}|$, décrits par une propriété quelconque $p \in \mathscr{P}(\mathscr{C})$, cf. sous-section 5.3.3
2. filtrer les propriétés entièrement redondantes de $\mathscr{P}(\mathscr{C})$ par rapport à I_u, cf. sous-section 5.3.4.

3. enfin, calculer le score de chaque propriété restante de $\mathscr{P}(\mathscr{C})$ et les classer selon ces scores, cf. sous-section. 5.3.2

La méthode PEM proposée a fait l'objet d'une soumission à la revue *Knowledge-Based Systems* (KBS) [Du et al., 2021b].

5.4 Protocole d'évaluation

Cette section détaille l'évaluation du modèle d'explication basé sur les propriétés (PEM) que nous avons présenté dans la section précédente. Nous avons choisi de comparer PEM avec le modèle ExpLOD qui exploite les propriétés *broader* [Musto et al., 2019]. Les auteurs ont montré dans [Musto et al., 2016b] que la version *basic* d'ExpLOD permet de surpasser les autres approches d'explication de référence et que la version *broader* d'ExpLOD est encore plus efficace puisqu'elle permet de considérer des propriétés indirectes (*broader*). Dans cette section, nous utiliserons le terme ExpLOD tout court pour désigner sa variante *broader*.

La façon la plus pertinente d'évaluer et de comparer les approches d'explication des recommandations est de réaliser une étude en-ligne auprès d'utilisateurs [Nunes and Jannach, 2017]. Dans les sous-sections suivantes, nous discutons d'abord les éléments de l'évaluation de notre approche (*e.g.*, le jeu de données, le modèle de recommandation, etc.). Nous discutons ensuite du protocole expérimental (*i.e.*, les métriques d'évaluation, l'étude en-ligne avec les utilisateurs etc.). Nous présenterons et discuterons les résultats obtenus avec ce protocole dans la section 5.5 suivante.

5.4.1 Jeu de données *MovieTweetings* et alignements dans DBpedia

Nous adoptons le jeu de données *MovieTweetings*[5], qui est composé de *ratings* de films contenus dans des tweets structurés sur *Twitter* [Dooms et al., 2013]. Il s'agit d'un jeu de données mis à jour quotidiennement et largement utilisé dans la communauté des systèmes de recommandation. Les données de *ratings* que nous avons utilisées pour notre expérimentation comportent 877 374 évaluations (échelle de 1 à 10) de 68 376 utilisateurs sur 35 944 films.

Pour que les items d'un système de recommandation puissent bénéficier de la richesse des connaissances encodées dans DBpedia, il faut d'abord les *aligner* aux entités de DBpedia. Pour ce faire, nous avons suivi la méthode présentée dans la section 4.3.2 du chapitre précédent qui s'appuie sur une comparaison des distances de *Levenshtein* entre les labels des entités dans DBpedia et les titres des films. Grâce à cette méthode, nous avons pu aligner 18 983 films avec des entités de DBpedia.

Pour collecter les propriétés qui annotent directement les films du catalogue, nous avons choisi de récupérer les *genres*, *i.e.*, les propriétés indiquant les catégories de films.

[5] https://github.com/sidooms/MovieTweetings

Ces informations ont été récupérées pour tous les films alignés via le prédicat "*dct:subject*" dans le graphe de connaissances DBpedia. Notez que dans nos expérimentations, nous n'avons pas pris en compte d'autres prédicats tels que "*dbo:director*", "*dbo:musicComposer*", etc. Ce choix nous semble être justifié pour trois raisons principales. Premièrement, le genre des films est une caractéristique cruciale qui représente le mieux les caractéristiques des films et constitue un critère clé ayant un impact sur la satisfaction et les expériences des utilisateurs lorsqu'ils utilisent un système de recommandation de films [Nanou et al., 2010]. La deuxième raison est que les propriétés issues d'autres prédicats de DBpedia tels que "*dbo:director*" et "*dbo:writer*" sont souvent des nœuds littéraux (représentés par des chaînes de caractères) qui ne sont pas utiles. Enfin, la sémantique de ces propriétés est souvent répétée via le prédicat "*dct:subject*" que nous considérons, *e.g.*, ⟨*The Intouchables*[6], "*dbo:musicComposer*", *Ludovico_Einaudi*⟩ et ⟨*The Intouchables*, "*dct:subject*", *Films_scored_by_Ludovico_Einaudi*⟩.

En extrayant de DBpedia les genres de films pour tous les films du catalogue, nous avons rassemblé 22 720 annotations directes de films. Nous avons téléchargé le fichier *dump* sur le site officiel[7]. Ce fichier contient toutes les informations pour reconstruire le KG des catégories de DBpedia constitué de 1 639 815 nœuds. Nous l'avons utilisé pour construire une version locale de la hiérarchie des propriétés. Une vérification rapide révèle que ce graphe a plusieurs racines et contient quelques cycles inattendus (*e.g.*, *Pakistan Tehreek-e-Insaf, "skos:broader", Pakistan Tehreek-e-Insaf politicians, "skos:broader", Imran Khan, "skos:broader", Pakistan Tehreek-e-Insaf*). Nous avons utilisé une approche basique pour rompre ces cycles en supprimant certaines des 4 084 504 arêtes de ce graphe. Le DAG résultant possède 4 080 682 d'arêtes. Pour simplifier les traitements de ce graphe, nous avons lié les 61 017 nœuds racines distincts de ce graphe à un seul nœud plus général, étiqueté par "*owl:Thing*".

Notez que l'objectif est de proposer une approche rapide et générique basée sur les graphes de connaissances et les données liées, qui permet de fournir des explications personnalisées aux utilisateurs. Dans notre cas, le benchmark du film n'est rien de plus qu'un benchmark très pratique pour tester notre approche. Afin de garantir la reproductibilité de notre approche, les alignements des films que nous avons générés ainsi que la version locale de la hiérarchie des catégories de DBpedia sont disponibles dans un répertoire public[8].

5.4.2 Modèle de recommandation

Comme les deux modèles (ExpLOD et le nôtre) sont des modèles d'explication post-hoc, indépendants du modèle de recommandation utilisé. Ce dernier n'étant pas le point sur lequel nous nous focalisons, nous avons choisi d'utiliser le modèle de recommandation

[6] https://dbpedia.org/page/The_Intouchables
[7] http://downloads.dbpedia.org/repo/dbpedia/generic/categories/
[8] https://doi.org/10.5281/zenodo.5122902

basé sur les similarités sémantiques des items (l'approche CBF présentée dans la section 4.3.1.2 du chapitre précédent).

5.4.3 Métriques d'évaluation

La qualité des approches d'explication peut être partiellement évaluée par certaines mesures hors-ligne (qui ne nécessitent pas l'évaluation des utilisateurs), mais les mesures en-ligne (basées sur l'évaluation faite par des utilisateurs) sont généralement plus informatives.

5.4.3.1 Métriques hors-ligne

En utilisant un jeu de données des *ratings*, on pourrait essayer d'évaluer automatiquement la qualité des explications qui seraient fournies à un utilisateur en fonction de son profil. Comme il n'y a pas de *feedback* centré sur l'utilisateur, l'évaluation est limitée, mais on peut néanmoins tirer quelques enseignements de ces mesures hors-ligne. Par exemple, une méthode qui, souvent, ne fournit aucune explication ou seulement des explications peu informatives n'est clairement pas idéale.

Pour évaluer la capacité d'une méthode à produire des explications, nous nous référons à la métrique *précision de l'explicabilité* (*explainability precision*) [Abdollahi and Nasraoui, 2017]. Elle représente la proportion d'items explicables parmi la liste des top-N recommandations et est notée *EPrec*@N.

Un autre aspect que nous voudrions comparer ici est le contenu informationnel des propriétés utilisées lors de l'explication. En général, le contenu informationnel – ou *Information Content* (IC) d'un concept (ou d'une propriété dans notre cas) donne une estimation de son degré de généralité/spécificité, et est utilisé pour une meilleure compréhension de la sémantique du concept [Sánchez et al., 2011]. Plus une propriété est générale, *e.g.*, *Films*, *Creative works*, moins elle est informative. L'IC d'une propriété est généralement défini en fonction de sa position au sein de la hiérarchie ontologique (définition intrinsèque) ou en fonction du nombre d'items qu'elle décrit indirectement (définition extrinsèque). Comme notre manière de choisir les propriétés dépend du nombre d'items qu'elles décrivent (tout comme l'IC extrinsèque), nous préférons nous appuyer sur une définition intrinsèque de l'IC pour éviter les biais possibles qui pourraient favoriser notre méthode. En suivant l'équation (9) de [Sánchez et al., 2011], nous utilisons donc la métrique d'IC suivante :

$$\mathrm{IC}(p) = -\log\left(\frac{|\mathrm{feuilles}(p)+1|}{|\mathrm{feuilles}(\mathscr{H})|}\right) \tag{5.9}$$

où feuilles(p) est l'ensemble des propriétés subsumées par p, qui sont des feuilles (*i.e.* elles ne subsument aucune autre propriété), et $|\mathrm{feuilles}(\mathscr{H})|$ est le nombre total de feuilles dans le DAG de propriétés.

5.4.3.2 Métriques en-ligne

Les explications des recommandations devraient augmenter la confiance des utilisateurs envers le SdR et favoriser leurs interactions. En particulier, une bonne explication devrait amener les utilisateurs à : i) mieux comprendre la recommandation (*Transparence*) ; ii) être plus convaincus par la recommandation (*Persuasion*) ; iii) apprendre de nouvelles informations sur les items recommandés (*Engagement*) ; iv) augmenter leur confiances dans le SdR (*Confiance*) et v) mieux évaluer la qualité de l'item recommandé (*Efficacité*), selon [Tintarev and Masthoff, 2012; Gedikli et al., 2014]. Il a été proposé d'évaluer ces métriques par un protocole standardisé en deux phases, mené avec des utilisateurs. En présentant l'affiche du film recommandé et l'explication proposée, l'utilisateur est invité à utiliser une échelle à 5 points (de 1=tout à fait en désaccord à 5=tout à fait d'accord) pour répondre aux cinq questions énumérées dans le tableau 5.1. L'utilisateur est ensuite invité à réévaluer la pertinence de la recommandation (la question associée à la métrique d'*Efficacité* dans le tableau 5.1), mais après avoir regardé sa bande-annonce. La différence entre l'évaluation du film recommandé faite avant et après avoir vu la bande-annonce est considérée comme un indicateur de l'*Efficacité* de l'explication. Un changement de note pourrait indiquer que la bande-annonce fournit de nouvelles informations pertinentes qui manquaient dans l'explication, tandis qu'une note stable peut indiquer que l'explication était suffisamment complète. Ce protocole a été utilisé dans de nombreuses études de la littérature [Tintarev and Masthoff, 2012; Gedikli et al., 2014; Musto et al., 2016b; Lully et al., 2018; Musto et al., 2019].

TABLEAU 5.1 – Détails du Questionnaire

Métrique (Phase)	Question correspondante dans l'étude
Efficacité (I, II)	J'aime la recommandation.
Transparence (I)	Je comprends pourquoi ce film m'a été recommandé.
Persuasion (I)	L'explication fournie est convaincante.
Engagement (I)	Cette explication m'a permis de découvrir certaines informations sur le film recommandé.
Confiance (I)	L'explication fournie augmente ma confiance dans le système de recommandation.

5.4.4 Étude réalisée avec des utilisateurs

Pour réaliser notre étude auprès des utilisateurs, nous avons déployé une application web implémentant le modèle PEM proposé et le modèle ExpLOD *broader*. L'ensemble du code source mettant en œuvre ces approches et l'application web sont disponibles dans un répertoire GitHub[9].

En bref, chaque participant à notre étude a effectué les étapes suivantes (évaluation en deux phases) :

1. **Collecte des données démographiques de l'utilisateur**. Nous avons d'abord de-

[9] https://github.com/lgi2p/web_app

mandé au participant de remplir un formulaire pour collecter ses données démographiques telles que le nom/prénom, le genre et l'adresse mail.

2. **Génération de recommandations et d'explications**. Le participant est invité à sélectionner dans le catalogue au moins 5 films qu'il aime. Cela permet de construire le profil utilisateur (I_u). Ensuite, le système de recommandation basé sur le contenu est utilisé pour identifier le film le plus pertinent (top-1) en fonction du profil utilisateur. En même temps, l'un des deux modèles d'explication (ExpLOD et PEM) est choisi au hasard. Ensuite, pour le film recommandé, l'explication est générée sur la base du modèle d'explication attribué. Le nombre de propriétés utilisées pour l'explication a été fixé à $k = 3$ pour les deux approches.

3. **Phase-I de l'évaluation par questionnaire**. Pour le film recommandé accompagné de son explication, le participant est invité à évaluer les différents critères d'évaluation du tableau 5.1. Au cours de la phase I, seuls le titre et l'affiche du film sont montrés à l'utilisateur, qui est invité à indiquer dans quelle mesure il aime cette recommandation (*i.e.*, phase-I de l'*Efficacité*), et à l'évaluer également selon les quatre autres critères.

4. **Phase-II de l'évaluation par questionnaire**. Au cours de la deuxième phase, l'utilisateur est invité à réévaluer le film recommandé après avoir regardé une bande-annonce de ce dernier. La proximité des deux notes attribuées au film recommandé peut alors être utilisée comme indicateur de l'*Efficacité* de l'explication. [Tintarev and Masthoff, 2012; Gedikli et al., 2014].

5.5 Résultats

Nous discutons dans cette section les résultats de notre étude en-ligne réalisée sur un panel de 155 participant (45% de femmes et 55% d'hommes), chacun évaluant l'explication d'un film décrit par les propriétés de DBpedia. Nous discutons d'abord de quelques indicateurs concernant l'optimisation de la sous-ontologie ; nous fournissons ensuite les métriques d'évaluation mesurées au cours de l'étude des utilisateurs en-ligne et nous terminons en détaillant une étude de cas (un exemple) qui met en lumière les différences entre ExpLOD et PEM, en termes de sélection des propriétés.

5.5.1 Efficacité de l'optimisation de la sous-ontologie

Alors que le graphe de connaissances des catégories de DBpedia contient 1 639 815 propriétés, seules 22 720 (1.38 %) d'entre elles décrivent directement l'un des 18 983 films de notre catalogue. Étant donné l'étendue de cette ontologie et compte tenu de l'ensemble

des annotations directes de films que nous avons extraites, nous avons appliqué la méthode proposée dans [Ranwez et al., 2011] pour extraire une sous-ontologie relative au domaine cinématographique de taille plus raisonnable. La sous-ontologie obtenue contient 28 424 (1.73%) propriétés. L'exclusion de cette sous-ontologie des propriétés qui n'ont pas d'ancêtres (y compris elles-mêmes) annotant directement un film (voir équation (5.6)) permet de réduire légèrement la hiérarchie des propriétés à considérer, ce qui conduit à un DAG de 28 016 (1.70 %) propriétés. Pour le modèle ExpLOD, l'ontologie entière a été utilisée lors de la génération des explications. L'utilisation de la sous-ontologie nous a permis de développer une solution à faible latence tout en traitant la hiérarchie entière de propriétés de DBpedia.

Le code source mettant en œuvre le processus d'extraction des sous-ontologies est disponible dans un répertoire GitHub[10].

5.5.2 Comparaison des performances sur l'étude d'utilisateurs

Parmi ces 155 explications, 82 ont été générées en utilisant l'approche ExpLOD et 73 en utilisant la méthode PEM que nous avons proposée. Ces 155 profils utilisateurs et explications ont également été utilisés pour évaluer les métriques hors-ligne. Les deux approches ont un *EPrec*@1 = 1 optimal. Si une liste d'items plus grande doit être expliqués (*i.e.*, la taille de I_r augmente), *EPrec*@5 = 0.7 diminue pour ExpLOD alors qu'il reste égal à 1 pour PEM. Ceci reflète essentiellement la différence des objectifs des deux méthodes ; lorsque plusieurs items doivent être expliqués simultanément, ExpLOD se concentre sur la factorisation des explications plutôt que sur le fait d'essayer de produire la meilleure explication pour chaque item. Comme cette tâche d'explication d'un groupe d'items sort du cadre de nos travaux, nous concentrons le reste de nos comparaisons sur les explications produites par ExpLOD pour un seul item (*i.e.* $I_r = \{i_r\}$), de ce fait les deux approches ont un même objectif : produire la meilleure explication pour i_r.

L'IC moyen des 246 propriétés utilisées dans les 82 explications générées par ExpLOD est de 7.49 alors qu'il est de 10.42 pour les 219 propriétés utilisées dans les 73 explications générées par PEM. Cela signifie que les explications fournies par PEM sont en moyenne beaucoup plus informatives. Nous avons mesuré la valeur moyenne des 5 métriques en-ligne sur les 82 évaluations des utilisateurs associés avec la méthode d'ExpLOD et les 73 évaluations de PEM. En moyenne, PEM a été mieux évalué que ExpLOD pour toutes les métriques. Pour tester la significativité de ces différences concernant le ressenti des utilisateurs, nous avons adopté le test de *Wilcoxon-Mann-Whitney* (avec $p\text{-}value < 0.05$). Les améliorations sont significatives pour *Persuasion*, *Engagement*, *Confiance* ainsi que pour le contenu informationnel moyen, *IC*. Tous ces résultats sont fournis dans le tableau 5.2. Le fichier de résultats de notre étude est accessible dans un répertoire public sur Ze-

[10] https://github.com/lgi2p/Sub-Ontology_Extraction

nodo[11].

TABLEAU 5.2 – Résultats de l'étude utilisateur. Les meilleures mesures sont en gras et les améliorations statistiquement significatives sont indiquées par (*).

	Transparence	*Persuasion*	*Engagement*	*Trust*	*Efficacité*	*IC*
ExpLOD	3.91	2.83	2.45	2.62	0.40	7.49
PEM	**4.21**	**3.45(*)**	**3.16(*)**	**3.30(*)**	**0.34**	**10.42(*)**

5.5.3 Un exemple d'étude de cas

Dans cette section, nous détaillons un exemple d'étude de cas pour mettre en lumière les différences entre ExpLOD et PEM (figure 5.3).

Le film recommandé (*Bitter Moon*[12]) doit être expliqué à partir d'un profil utilisateur composé de trois films : *Intouchables*[13], *Bienvenue chez les Ch'tis*[14] et *Amélie Poulain*[15]. Les top-3 propriétés selon la méthode PEM que nous proposons sont *French comedy films*, *Films shot in France* et *French-language films*. Aucune de ces propriétés ne figure dans les top-3 propriétés classées par ExpLOD. La première propriété (*i.e., French comedy films*) est prise en compte par ExpLOD mais elle n'est pas classée dans les top-3 propriétés tandis que les deux dernières ne sont même pas prises en compte puisqu'elles ne font pas partie de l'ensemble des propriétés *broader* considérées par ExpLOD. Les top-3 propriétés sélectionnées par ExpLOD dans cet exemple sont *Film scores by composer*, *Films by director* et *Films by French directors*. Les deux premières propriétés sont assez peu informatives (et ignorées par la méthode PEM puisqu'elles n'annotent directement aucun item du catalogue). Notez que la propriété *Films by director* est également *totalement redondante* par rapport à la troisième propriété *Films by French directors*, sélectionnée par ExpLOD.

[11] https://doi.org/10.5281/zenodo.5122902
[12] https://dbpedia.org/page/Bitter_Moon
[13] https://dbpedia.org/page/The_Intouchables
[14] https://dbpedia.org/page/Welcome_to_the_Sticks
[15] https://dbpedia.org/page/Am%C3%A9lie

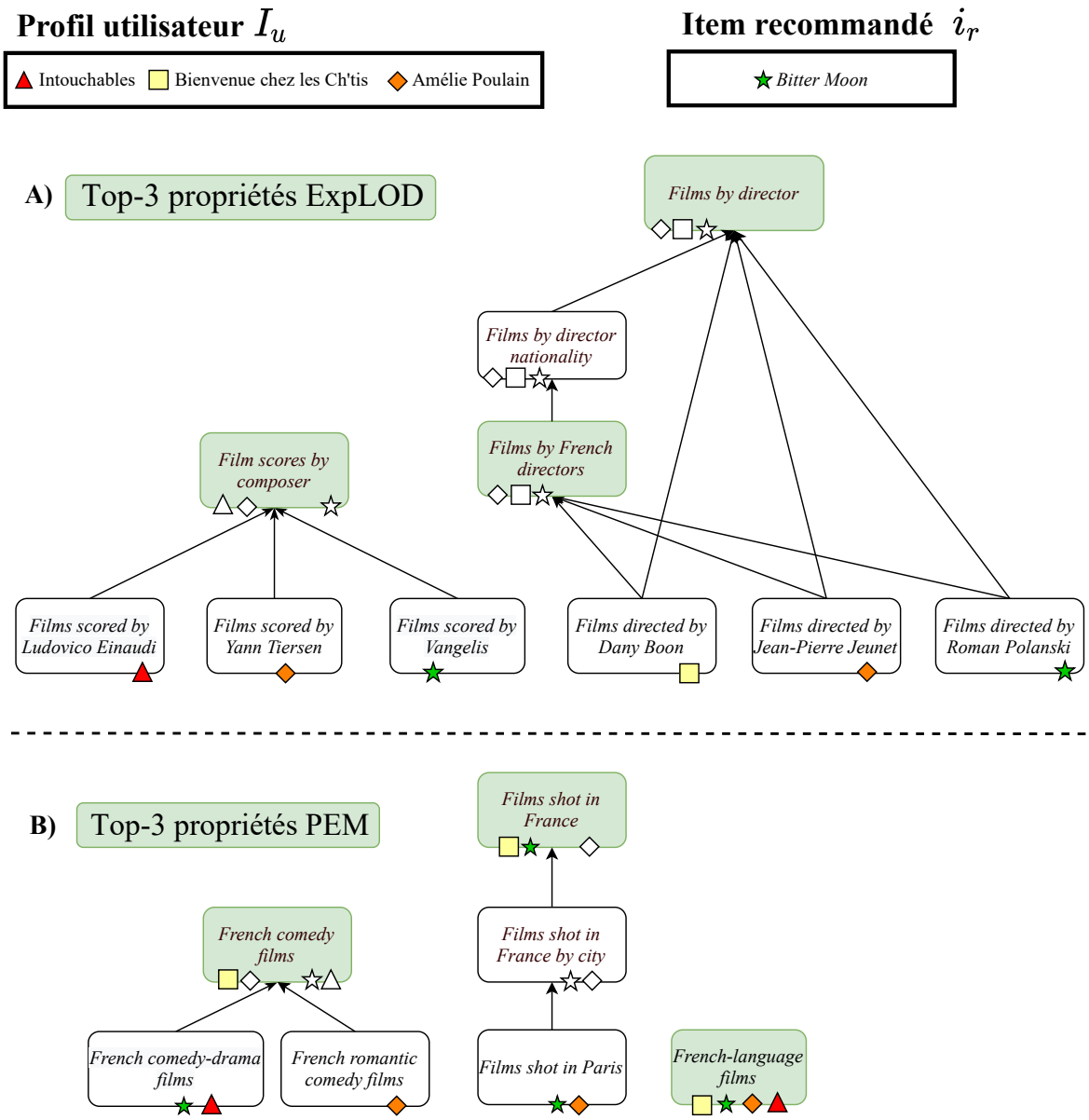


FIGURE 5.3 – Un exemple d'étude de cas comparant la variante *broader* de l'approche ExpLOD A) et PEM B). Les films du profil utilisateur sont représentés par des symboles rouges, jaunes et orange, tandis que l'étoile verte représente le film recommandé à expliquer. Les symboles géométriques blancs représentent les descriptions indirectes des films induites par la hiérarchie des propriétés. Les trois propriétés les mieux classées pour chaque approche sont mises en évidence par la couleur verte.

5.6 Conclusion de chapitre

L'explication des recommandations constitue un aspect important des systèmes de recommandation car elle fournit aux utilisateurs des informations supplémentaires, justifiant pourquoi un item est recommandé et les aide ainsi à prendre plus rapidement la meilleure décision concernant un achat, un visionage de film ou une écoute de titre musical.

Dans ce chapitre, nous avons présenté une approche d'explication post-hoc générique et personnalisée, qui exploite les données liées et les graphes de connaissances. Plus

précisément, les données liées s'appuient sur une hiérarchie qui organise les concepts au travers de relations de subsomption telles que *skos:broader* ou *rdf:type*. Cette hiérarchisation englobe une sémantique implicite, particulièrement utile pour l'explication de recommandations.

Grâce à des optimisations algorithmiques reposant sur l'extraction de sous-ontologies et le parcours de graphes par ordre post-fixé, nous démontrons que cette hiérarchie peut être exploitée de manière efficace. Nous fournissons également des solutions simples pour limiter la redondance des explications dans l'approche proposée et éviter l'utilisation de propriétés non-pertinentes qui sont utiles pour organiser la hiérarchie, mais peu informatives pour l'explication. L'étude menée auprès de 155 utilisateurs confirme l'avantage de l'approche que nous proposons par rapport aux approches existantes basées sur le LOD, comme ExpLOD, qui ne prend pas en compte l'ensemble de la hiérarchie de propriétés et ne tient pas explicitement compte de la pertinence des propriétés et des éventuels problèmes de redondance.

Il existe un autre type d'approches d'explications personnalisées qui ignorent le contenu des items et tirent uniquement parti de la matrice des *ratings* utilisateur-item. Ces méthodes génèrent des explications post-hoc basées sur les interactions utilisateur-item (données collaboratives), conduisant à des explications telles que : « Nous vous recommandons l'item x parce que vous avez aimé l'item y et que 80% des utilisateurs qui aiment y aiment aussi x ».

Nous pensons que ces deux types d'approches d'explication ne sont pas exclusifs. En fonction du contexte, l'utilisateur peut être plus convaincu par une explication basée sur les connaissances liées aux items, extraites à partir des données liées, ou par une approche collaborative. En effet, si l'explication basée sur le LOD n'est composée que de propriétés très génériques, il peut être préférable de s'appuyer sur une explication collaborative ; inversement, si l'explication collaborative repose sur un pourcentage très faible, l'explication basée sur les données liées est probablement préférable. On peut donc imaginer de privilégier une stratégie plutôt que l'autre en fonction de leurs résultats ou même de combiner les résultats des deux types d'approches dans une explication hybride mélangeant les deux types de justification.

Chapitre 6

Conclusions et perspectives

6.1 Conclusions

De plus en plus de données sont collectées lors de notre utilisation des outils numériques. Cela soulève évidemment des questions éthiques, et l'équilibre entre la protection de la vie privée des utilisateurs et l'amélioration des services proposés qui découlent de l'utilisation de ces données est au cœur de nombreux débats actuels. Pour les fournisseurs de services (publics ou privés), un des enjeux majeurs est de transformer la masse de données dont ils disposent en connaissances permettant d'améliorer l'expérience utilisateur en la personnalisant. Dans cette thèse, nous nous sommes intéressés aux systèmes de recommandation (SdR) sous l'angle non seulement de l'*accuracy* des recommandations qu'ils produisent mais également de la diversité de ces recommandations et de la capacité du système à expliquer ses choix à l'utilisateur. En nous appuyant sur des méthodes d'apprentissage automatique et/ou d'ingénierie des connaissances, nous avons mené des travaux de recherche visant à : i) améliorer l'*accuracy* de la prédiction des préférences des utilisateurs (*i.e. ratings*), ii) mieux comprendre la *diversification* au sein des différents systèmes de recommandation et iii) rendre les recommandations plus transparentes et convaincantes en fournissant des *explications* personnalisées et appropriées. Pour évaluer nos contributions, nous nous sommes basés sur des protocoles et des jeux de données largement utilisés au sein de la communauté scientifique travaillant sur les SdR. Nos évaluations s'appuient sur plusieurs jeux de données couvrant différents domaines : films (MovieLens-100K, MovieLens-1M, MovieTweetings), blagues (Jester), dessins animés japonais (Anime) et livres (LibraryThing). Cette section récapitule les approches proposées dans cette thèse et résume les principales conclusions qui découlent des résultats de nos évaluations sur différents jeux de données.

Notre première contribution concerne l'amélioration de l'*accuracy* des recommandations. Nous avons proposé l'approche EBCR[1] (cf. chapitre 3), permettant d'ajuster l'es-

[1] *Empirical Bayes Concordance Ratio*

timation des similarités, entre utilisateurs ou items, qui sont au cœur des approches de filtrage collaboratif (CF) basé sur le voisinage. L'approche EBCR permet d'ajuster la similarité entre deux utilisateurs en prenant en compte l'hétérogénéité possible de leur système de notation, le nombre d'items qu'ils ont co-notés et la distribution globale des similarités observées entre utilisateurs. Si deux utilisateurs n'ont co-noté que peu d'items, l'idée sous-jacente au modèle EBCR est que, faute d'information, leur similarité doit être proche de la moyenne des similarités observées. A l'inverse, s'ils ont co-noté un grand nombre d'items, leur similarité sera essentiellement basée sur la concordance de leurs notations sur ces items. L'approche EBCR est simple et générique, et elle peut facilement se combiner aux métriques classiques sans augmenter les temps de calculs. Nos tests confirment que cette approche permet d'améliorer l'*accuracy* des recommandations pour toutes les métriques de similarité testées (*e.g.*, cosinus, corrélation de Pearson, etc.). Nos résultats montrent également qu'un SdR de type CF intégrant l'approche EBCR peut, sur certains jeux de données ayant relativement peu de données manquantes, conduire à une meilleure prédiction des *ratings* que les modèles les plus récents basés sur des réseaux de neurones profonds (*i.e.*, le modèle NeuMF [He et al., 2017]). Par contre si l'évaluation se base non pas sur la qualité de la prédiction des *ratings* (métriques d'évaluation de type *Root Mean Squared Error* – RMSE, ou *Mean Absolute Error* – MAE), mais sur la qualité globale de la liste de recommandations (métriques top-N de type *Hit Rate* – HR, ou *Normalised Discounted Cumulative Gain* – NDCG), alors le modèle NeuMF montre de meilleurs performances prédictives que les SdR de type filtrage collaboratif classiques, qu'ils soient basés sur la factorisation matricielle (*Singular Value Decomposition* – SVD/SVD++) ou sur l'utilisation d'utilisateurs/items voisins et cela même si la similarité utilisée pour définir le voisinage intègre l'approche EBCR proposée. Lors de nos évaluations des différents SdR, nous avons également constaté que l'utilisation des connaissances sur les items permet d'améliorer significativement la qualité des listes d'items recommandés (l'*accuracy* top-N). Cela permet de dépasser les performances des approches de CF classiques (en réduisant les problèmes de démarrage à froid) et d'atteindre, sur certains jeux de données, des performances équivalentes aux approches basées sur les réseaux de neurones. Par exemple, une méthode hybride combinant simplement le fait de recommander un item en fonction de sa popularité et de sa similarité sémantique (basée sur la description de son contenu) avec les items aimés par l'utilisateur peut montrer des performances nettement supérieures aux approches de filtrage collaboratif conventionnelles. Lors de nos expérimentations, l'approche hybride basée sur les *embeddings* de graphes de connaissances se révèle être quasiment aussi performante que l'approche basée sur les réseaux de neurones NeuMF. Ainsi, il semble que l'intégration de connaissances sur les items permet de largement améliorer les performances en termes d'*accuracy* des SdR classiques. D'une manière ou d'une autre il semble que les approches basées sur les réseaux de neurones profonds arrivent à mieux gérer l'absence de telles informations et sont capables de produire des recommandations aussi pertinentes sur la seule base des

données d'interaction « utilisateur-item » quand ces dernières sont en quantité suffisantes. Cela légitime en partie l'engouement récent pour ces approches. Néanmoins, si l'intégration basique de connaissances (souvent imparfaites et incomplètes) sur les items permet d'atteindre des performances proches de celles des réseaux de neurones profonds, on peut légitimement se demander si cela ne vaudrait pas la peine d'explorer plus avant l'utilisation de ces données sémantiques.

Les systèmes de recommandation sont depuis longtemps évalués de manière systématique sur leur aptitude à proposer des items pertinents. Si ce critère est naturellement important, il ne suffit pas, à lui seul, pour prédire la satisfaction qu'auront les utilisateurs lors de l'utilisation d'un SdR. Il a été montré que d'autres critères, comme par exemple la *diversité* des recommandations, jouent un rôle important dans la satisfaction des utilisateurs. Cependant, peu d'études ont été publiées concernant les performances des différents systèmes de recommandation en termes de diversité. La manière la plus répandue de traiter ce problème est d'utiliser un post-traitement qui construit de manière gloutonne la liste finale des recommandations en piochant les items dans une liste plus large. Le même post-traitement est généralement utilisé pour tous les utilisateurs (qui peuvent avoir des attentes différentes en termes de diversification) et quel que soit le SdR utilisé (ignorant le fait que certains SdR renvoient généralement des listes bien plus diversifiées que d'autres). C'est dans ce contexte que nous avons conduit une étude approfondie (cf. chapitre 4) visant à comparer différentes familles de systèmes de recommandation sous l'angle de la diversification. Dans ce cadre, l'utilisation de données sémantiques relatives aux items représente un réel atout pour quantifier précisément la diversité d'une liste de recommandations. Ces données permettent en effet d'utiliser des mesures de similarité sémantique entre (les descriptions des) items pour quantifier cette diversité. Nos résultats indiquent que : i) les approches de CF classiques fournissent en général des listes d'items plus diversifiées que les approches hybrides et que les approches de CF basées sur les réseaux de neurones ; ii) l'approche basée sur le contenu (CBF) semble être celle qui souffre le plus de la problématique de la sur-spécialisation puisque ses recommandations sont généralement les moins diversifiées ; iii) il semble raisonnable de coupler l'approche CBF avec une optimisation gloutonne classique de diversification qui permet d'améliorer simultanément l'*accuracy* et la diversité des recommandations ; iv) il semble par contre problématique d'utiliser cette optimisation gloutonne de la diversification avec les autres approches de recommandation, leurs listes de recommandations étant déjà assez, voire trop diversifiées par rapport aux attentes des utilisateurs et v) l'approche alternative de diversification que nous proposons permet de prendre en compte ce niveau d'attente des utilisateurs et ainsi d'éviter ce phénomène de sur-diversification.

Un autre aspect influençant fortement la satisfaction des utilisateurs pour un SdR est lié au fait qu'ils comprennent pourquoi les items sélectionnés leurs sont recommandés.

Notre dernière contribution se focalise sur ce point. Nous avons développé et testé un modèle générique d'explication basé sur les propriétés des items que nous avons nommé PEM (*Property-based Explanation Model*, cf. chapitre 5). Notre approche est inspirée des travaux de [Musto et al., 2019] dans lesquels les explications des items recommandés sont construites à partir des propriétés qu'ils partagent avec les items aimés par l'utilisateur en question. Ces propriétés partagées sont identifiées en exploitant les connaissances dont on dispose sur les items, qui sont encodées au sein d'un graphe de connaissances. Le graphe de connaissances utilisé doit être suffisamment riche et complet pour permettre de décrire le contenu de l'ensemble des items gérés par le SdR. DBpedia fournit un tel graphe, contenant plusieurs millions d'éléments organisés, notamment de manière hiérarchique via la relation de subsomption (généralisation). Lorsqu'un item est explicitement annoté par une propriété, il est également implicitement annoté par toutes les propriétés qui la subsument. L'approche proposée par [Musto et al., 2019] n'utilise que partiellement la connaissance encodée dans cette hiérarchie en ne considérant qu'une faible fraction des annotations implicites (subsumers de premier ordre). Pour aller plus loin, l'approche PEM que nous proposons exploite l'intégralité de cette immense hiérarchie de propriétés. Cela nous permet de prendre en compte le fait qu'un item est décrit non seulement par les propriétés qui lui sont explicitement associées mais également par toutes celles qui les subsument. La prise en compte de l'ensemble de ces annotations implicites permet d'identifier davantage de propriétés communes entre les items recommandés et ceux aimés. Certaines de ces propriétés additionnelles sont indispensables pour pouvoir expliquer de manière convaincante les items recommandés. Cependant, la prise en compte de l'ensemble des subsumers augmente considérablement la masse de données à traiter. Cela nécessite l'utilisation d'optimisations algorithmiques permettant de traiter efficacement un graphe de connaissances qui, dans le cas de DBpedia par exemple, peut contenir des millions de nœuds et de relations. Pour ces optimisations nous nous sommes appuyés sur l'algorithme d'extraction de sous-ontologie, qui permet de restreindre une ontologie pour ne garder qu'un sous ensemble de concepts/-propriétés d'intérêt [Ranwez et al., 2011]. Nous avons ainsi pu restreindre efficacement le graphe de DBpedia pour ne garder qu'un sous-ensemble des concepts (propriétés) relatifs au domaine de la recommandation (domaine cinématographique dans notre test). L'utilisation d'algorithmes adaptés permet d'identifier rapidement des propriétés décrivant (de manière explicite ou implicite) à la fois l'item recommandé à expliquer et les items aimés par l'utilisateur focal. Parmi ces propriétés communes, certaines peuvent être non informatives (un exemple frappant est la propriété "*Films by genres*") et d'autres fortement redondantes (*e.g.*, "*Films set in France*" et "*Films set in Paris*"). L'approche PEM que nous proposons inclut donc des méthodes permettant de sélectionner parmi ces propriétés communes celles qui sont les plus informatives et d'éviter d'utiliser des propriétés redondantes dans une même explication. Nous avons conduit une étude en-ligne (reprenant un protocole d'évaluation largement utilisé pour ce type de tests, notamment par

[Musto et al., 2019]). Les résultats de cette étude incluant 155 participants ont montré que notre méthode PEM propose des explications significativement meilleures (*e.g.*, en termes de confiance, persuasion, etc.) que celles proposées par [Musto et al., 2019]. Nos résultats semblent ainsi confirmer que : les connaissances liées aux items représentent un réel atout pour expliquer de manière convaincante les recommandations proposées par les SdR. Une exploitation efficace de ces connaissances peut permettre de fournir aux utilisateurs des explications utiles pour mieux comprendre les raisons qui ont conduit le SdR à recommander les items proposés. Cette compréhension permet d'augmenter la confiance des utilisateurs dans le SdR et les aide à faire plus efficacement leurs choix définitifs parmi la liste d'items recommandés.

6.2 Perspectives de travaux futurs

Dans cette section, nous discutons quelques pistes de recherche relatives aux travaux que nous avons présentés dans ce manuscrit de thèse.

Concernant l'étude de la diversité des recommandations présentée dans le chapitre 4, nous avons considéré 7 systèmes de recommandation appartenant à différentes familles telles que les recommandations basées sur i) la popularité des items, ii) le contenu des items, iii) le filtrage collaboratif classique, iv) le filtrage collaboratif basé sur des réseaux de neurones et v) l'*embedding* du graphe de connaissances des items. Les résultats de cette étude montrent que les approches les plus récentes, basées sur des réseaux de neurones [He et al., 2017] et les *embeddings* du graphe de connaissances [Palumbo et al., 2018b], permettent de constituer des listes de recommandations contenant des items très pertinents en termes d'*accuracy* tout en ayant un niveau intermédiaire de diversité. Il nous semblerait donc intéressant d'évaluer de manière plus exhaustive les différentes méthodes de ce type tout en questionnant le compromis « *diversité-accuracy* », en utilisant par exemple des fonctions objectifs tirant profit d'une analyse multicritère minutieuse. Une étude complémentaire à la nôtre pourrait reprendre notre protocole d'évaluation et y intégrer d'autres approches basées sur des réseaux de neurones et les *embeddings* de graphe comme par exemple *Wide & Deep* [Cheng et al., 2016], *CDL* [Wang et al., 2015], *entity2rec* [Palumbo et al., 2020], etc. En plus des évaluations hors-lignes, une étude en-ligne pourrait être envisageable en demandant à des participants d'indiquer leur satisfaction concernant les items recommandés tant en termes d'*accuracy* que de diversité.

Concernant les explications *post-hoc* des recommandations, la méthode que nous avons proposée (chapitre 5) exploite les propriétés des items, disponibles sous forme de données liées, pour expliquer à l'utilisateur les recommandations faites par le système de recommandation. Les travaux de [Musto et al., 2019] ont montré que la qualité des explications basées sur les données liées semble stable et indépendante du système de recommandation utilisé, *i.e.*, ils n'observent pas de variations significatives de la qualité des explications selon le système de recommandation utilisé pour générer la liste d'items

à expliquer. Il arrive cependant que certaines recommandations soient difficilement explicables, même avec ce type d'approche. Ceci s'explique notamment par la grande variabilité du nombre de données liées disponibles par item. Certaines descriptions sont très riches alors que d'autres sont minimalistes voire complètement absentes. Comme nous l'avons discuté dans la section 5.6, une autre approche pour générer des explications *post-hoc* consiste à s'appuyer sur des statistiques d'interactions *utilisateur-item*, *e.g.* « nous vous recommandons l'item x parce que vous avez aimé l'item y et que 80% des utilisateurs qui aiment y aiment aussi x ». Lorsque les pourcentages indiqués dans l'explication sont élevés, comme c'est le cas dans notre exemple, l'explication parait convaincante. Mais il peut arriver que les explications ainsi générés s'appuient, faute de mieux, sur des pourcentages très faibles conduisant alors à des explications peu satisfaisantes. Cette approche souffre également du problème classique de *démarrage à froid* puisque les nouveaux items, ayant moins de notes, seront forcément plus difficiles à expliquer. Nous pensons donc qu'il serait intéressant de combiner ces deux types d'approches pour générer des explications *post-hoc* en tirant le meilleur parti de chacune. L'idée serait de favoriser les explications reposant soit sur des propriétés très informatives retrouvées fréquemment dans les items du profil utilisateur ou à défaut sur des pourcentages d'interactions *utilisateur-item* élevés. Il faudrait pour cela définir une fonction de score permettant d'estimer la qualité des explications générées par ces deux approches pour pouvoir les interclasser et sélectionner les plus pertinentes. Au delà de la qualité des annotations et du nombre de notes disponibles pour un item, il est probable que le SdR sous-jacent joue également un rôle dans le fait que les items recommandés soient globalement plus facilement explicables par une approche de type *Linked-Data* ou par une approche de type interactions *utilisateur-item*. Par exemple nous pouvons, raisonnablement penser qu'il serait plus pertinent d'expliquer les recommandations faites par un SdR de type filtrage collaboratif, *i.e.*, sélectionnant ses recommandations sur la base de données d'interaction *utilisateur–item*, à partir d'explications construites elles aussi sur ces interactions plutôt que par celles construites à partir des données liées. Définir une manière de combiner au mieux ces deux types d'explications nous semble donc être une piste très prometteuse.

Nous espérons donc qu'au terme de cette lecture, de nouveaux travaux seront inspirés par nos réalisations et qu'ils ouvriront la voie à de nouvelles innovations dans le domaine de la recommandation.

Bibliographie

Abdollahi, B. and Nasraoui, O. (2017). Using explainability for constrained matrix factorization. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 79–83. 145

Abdollahpouri, H. (2019). Popularity bias in ranking and recommendation. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, page 529–530. 102

Adomavicius, G. and Kwon, Y. (2012). Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5) :896–911. 92, 93, 95, 98, 101

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6) :734–749. 84

Aggarwal, C. C. (2016a). Content-based recommender systems. In *Recommender Systems*, pages 139–166. Springer. 29

Aggarwal, C. C. (2016b). Context-sensitive recommender systems. In *Recommender Systems*, pages 255–281. Springer. 10

Aggarwal, C. C. (2016c). Ensemble-based and hybrid recommender systems. In *Recommender Systems*, pages 199–224. Springer. 37

Aggarwal, C. C. (2016d). Evaluating recommender systems. In *Recommender systems*, pages 225–254. Springer. 40, 42, 43

Aggarwal, C. C. (2016e). An introduction to recommender systems. In *Recommender systems*, pages 1–28. Springer. 10, 122

Aggarwal, C. C. (2016f). Model-based collaborative filtering. In *Recommender systems*, pages 71–138. Springer. 20

Aggarwal, C. C. (2016g). Time-and location-sensitive recommender systems. In *Recommender Systems*, pages 283–308. Springer. 10

Ai, Q., Azizi, V., Chen, X., and Zhang, Y. (2018). Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9) :137. 129

Al-Hassan, M., Lu, H., and Lu, J. (2011). Personalized e-government services : Tourism recommender system framework. In *Web Information Systems and Technologies*, WEBIST '10, pages 173–187. Springer, Berlin, Heidelberg. 54

Ali, M., Jabeen, H., Hoyt, C. T., and Lehmann, J. (2019). The keen universe. In Ghidini, C., Hartig, O., Maleshkova, M., Svátek, V., Cruz, I., Hogan, A., Song, J., Lefrançois, M., and Gandon, F., editors, *The Semantic Web - ISWC 2019*, volume 11779, pages 3–18, Cham. Springer International Publishing. 110

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia : A nucleus for a web of open data. In *The semantic web*, volume 4825, pages 722–735. Springer. 57

Aytekin, T. and Karakaya, M. Ö. (2014). Clustering-based diversity improvement in top-N recommendation. *Journal of Intelligent Information Systems*, 42(1) :1–18. 92, 93, 112

Bellogin, A., Castells, P., and Cantador, I. (2011). Precision-oriented evaluation of recommender systems : an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 333–336. 45

Bennett, J., Lanning, S., et al. (2007). The netflix prize. In *Proceedings of KDD Cup and Workshop*, volume 2007, page 35. New York. 10, 39

Benouaret, I. (2017). *Un système de recommandation contextuel et composite pour la visite personnalisée de sites culturels*. Theses, Université de Technologie de Compiègne. 38, 54, 55

Berkovsky, S., Taib, R., and Conway, D. (2017). How to recommend? user trust factors in movie recommender systems. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, IUI '17, pages 287–300. 128

Berners-Lee, T. (2006). Linked data. *International Journal on Semantic Web and Information Systems*, 4(2). 56

Billsus, D., Pazzani, M. J., and Chen, J. (2000). A learning agent for wireless news access. In *Proceedings of the 5th international conference on Intelligent user interfaces*, IUI '00, pages 33–36. 38

Blanco-Fernandez, Y., Pazos-arias, J., Gil-Solla, A., Ramos-Cabrer, M., and Lopez-Nores, M. (2008). Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Transactions on Consumer Electronics*, 54(2) :727–735. 52

Boratto, L., Carta, S., and Satta, M. (2010). Groups identification and individual recommendations in group recommendation algorithms. In *Practical Use of Recommender Systems, Algorithms and Technologies*, PRSAT '10, pages 27–34. 89

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS '13, pages 2787–2795. 61, 105

Bradley, K. and Smyth, B. (2001). Improving recommendation diversity. In *Proceedings of the 12th Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*, AICS'01, pages 85–94. 93, 94, 95, 98, 100, 101

Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F., and Horrocks, I. (2002). Enabling knowledge representation on the web by extending rdf schema. *Computer Networks*, 39(5) :609 – 634. 48

Brown, L. D. (2008). In-season prediction of batting averages : A field test of empirical bayes and bayes methodologies. *The Annals of Applied Statistics*, pages 113–152. 3, 73, 75

Burke, R. (2002). Hybrid recommender systems : Survey and experiments. *User modeling and user-adapted interaction*, 12(4) :331–370. 37

Cacheda, F., Carneiro, V., Fernández, D., and Formoso, V. (2011). Comparison of collaborative filtering algorithms : Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1) :1–33. 84

Çano, E. and Morisio, M. (2017). Hybrid recommender systems : A systematic literature review. *Intelligent Data Analysis*, 21(6) :1487–1524. 37

Cantador, I., Bellogín, A., and Castells, P. (2008). News@hand : A semantic web approach to recommending news. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 5149 of *AH '08*, pages 279–283. Springer. 48

Carrer-Neto, W., Hernández-Alcaraz, M. L., Valencia-García, R., and García-Sánchez, F. (2012). Social knowledge-based recommender system. Application to the movies domain. *Expert Systems with Applications*, 39(12) :10990–11000. 48, 50, 51, 53, 54, 113

Castagnos, S., Brun, A., and Boyer, A. (2013). When diversity is needed... but not expected! In *Proceedings of the 3rd International Conference on Advances in Information Mining and Management*, IMMM'13, pages 44–50. 95, 123

Castells, P., Hurley, N. J., and Vargas, S. (2015). Novelty and diversity in recommender systems. In *Recommender systems handbook*, pages 881–918. Springer. 46

Chai, T. and Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae) ?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3) :1247–1250. 41

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4) :359–394. 75

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., and Shah, H. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 7–10, New York, NY, USA. Association for Computing Machinery. 156

Cheng, P., Wang, S., Ma, J., Sun, J., and Xiong, H. (2017). Learning to recommend accurate and diverse items. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 183–192. 101

Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, RecSys '16, pages 191–198. 10

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46. 39, 43, 44

Crnic, J. (2011). Introduction to modern information retrieval. *Library Management*, 32(4). 14, 31

De Gemmis, M., Lops, P., Musto, C., Narducci, F., and Semeraro, G. (2015). Semantics-aware content-based recommender systems. In *Recommender systems handbook*, pages 119–159. Springer. 29

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6) :391–407. 23

Di Noia, T., Mirizzi, R., Ostuni, V. C., and Romito, D. (2012a). Exploiting the web of data in model-based recommender systems. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 253–256. 48

Di Noia, T., Mirizzi, R., Ostuni, V. C., Romito, D., and Zanker, M. (2012b). Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, page 1–8. 48, 50, 57, 94, 103, 106, 108, 131

Di Noia, T. and Ostuni, V. C. (2015). Recommender systems and linked open data. In *Reasoning Web International Summer School*, pages 88–113. Springer. 48

Di Noia, T., Ostuni, V. C., Rosati, J., Tomeo, P., and Di Sciascio, E. (2014). An analysis of users' propensity toward diversity in recommendations. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 285–288. 93, 95, 96, 99, 101, 113, 122

Di Noia, T., Ostuni, V. C., Tomeo, P., and Sciascio, E. D. (2016). Sprank : Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1) :1–34. 59, 107, 113

Dooms, S., De Pessemier, T., and Martens, L. (2013). Movietweetings : a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys*, volume 2013, page 43. 143

Du, Y., Ranwez, S., Sutton-Charani, N., and Ranwez, V. (2019a). Ajustement bayesien des mesures de similarité entre utilisateurs pour améliorer les recommandations basées sur un filtrage collaboratif. In *LFA 2019*, Actes des 28es rencontres francophones sur la logique floue et ses applications - LFA 2019, Alès, France. IMT Mines Alès. 70

Du, Y., Ranwez, S., Sutton-Charani, N., and Ranwez, V. (2019b). Apports des ontologies aux systèmes de recommandation : état de l'art et perspectives. In Hernandez, N., editor, *30es Journées Francophones d'Ingénierie des Connaissances, IC 2019*, 30es Journées Francophones d'Ingénierie des Connaissances, IC 2019, pages 64–77, Toulouse, France. AFIA. 4, 49

Du, Y., Ranwez, S., Sutton-Charani, N., and Ranwez, V. (2021a). Is diversity optimization always suitable? toward a better understanding of diversity within recommendation approaches. *Information Processing & Management*, 58(6) :102721. 94

Du, Y., Ranwez, S., Sutton-Charani, N., and Ranwez, V. (2021b). Post-hoc recommendation explanations through an efficient exploitation of the dbpedia category hierarchy. *Knowledge-Based Systems*, soumise au journal, révision en cours. 143

Du, Y., Sutton-Charani, N., Ranwez, S., and Ranwez, V. (2021c). Ebcr : Empirical bayes concordance ratio method to improve similarity measurement in memory-based collaborative filtering. *PLoS ONE*, 16(8). 70

Ebesu, T., Shen, B., and Fang, Y. (2018). Collaborative memory network for recommendation systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 515–524. 102

Ekstrand, M. D., Harper, F. M., Willemsen, M. C., and Konstan, J. A. (2014). User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 161–168, New York, USA. 95, 103, 123

El-Dosuky, M. A., Rashad, M. Z., Hamza, T. T., and El-Bassiouny, A. H. (2012). Food recommendation using ontology and heuristics. In *International conference on advanced machine learning technologies and applications*, AMLTA '12, pages 423–429. 48

Euzenat, J., Ferrara, A., Hollink, L., Isaac, A., Joslyn, C., Malaisé, V., Meilicke, C., Nikolov, A., Pane, J., Sabou, M., et al. (2010). Results of the ontology alignment evaluation initiative 2009. Technical report, University of Trento. 63

Färber, M., Bartscherer, F., Menne, C., and Rettinger, A. (2018). Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web*, 9(1) :77–129. 123

Ferrari Dacrema, M., Cremonesi, P., and Jannach, D. (2019). Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 101–109. 5, 26, 43, 86, 94, 102, 104, 108, 124, 129

García-Sánchez, F., Colomo-Palacios, R., and Valencia-García, R. (2020). A social-semantic recommender system for advertisements. *Information Processing & Management*, 57(2) :102153. 113

Gedikli, F., Jannach, D., and Ge, M. (2014). How should i explain? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies*, 72(4) :367–382. 128, 146, 147

Gilad-Bachrach, R., Navot, A., and Tishby, N. (2004). Margin based feature selection-theory and algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, page 43. 61

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2) :199 – 220. 48

Gunawardana, A. and Shani, G. (2015). Evaluating recommender systems. In *Recommender systems handbook*, pages 265–308. Springer. 39

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). Deepfm : a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1725–1731. 26

Harispe, S., Ranwez, S., Janaqi, S., and Montmain, J. (2013). Mesures sémantiques basées sur la notion de projection RDF pour les systèmes de recommandation. In *24e journées d'ingénierie des connaissances - IC 2013*. 48, 54, 55

Harispe, S., Ranwez, S., Janaqi, S., and Montmain, J. (2015). Semantic similarity from natural language and ontology analysis. *Synthesis Lectures on Human Language Technologies*, 8(1) :1–254. 113

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets : History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4) :1–19. 78

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182. vi, 26, 27, 28, 29, 77, 86, 94, 104, 108, 129, 153, 156

Heitmann, B. and Hayes, C. (2010). Using linked data to build open, collaborative recommender systems. In *AAAI spring symposium : Linked Data Meets Artificial Intelligence*, volume 10, page 07. 57

Herlocker, J., Konstan, J. A., and Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5(4) :287–310. 14, 17, 68, 69, 70, 77, 81

Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 241–250. 129

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1) :5–53. 39, 43

Hou, L., Pan, X., and Liu, K. (2018). Balancing the popularity bias of object similarities for personalised recommendation. *European Physical Journal B*, 91(47). 112

Hug, N. (2020). Surprise : A python library for recommender systems. *Journal of Open Source Software*, 5(52) :2174. 45, 110

Jannach, D., Lerche, L., Kamehkhosh, I., and Jugovac, M. (2015). What recommenders recommend : an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, 25(5) :427–491. 95, 108

Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 687–696. 61

Jugovac, M., Jannach, D., and Lerche, L. (2017). Efficient optimization of multiple recommendation quality factors according to individual user tendencies. *Expert Systems with Applications*, 81 :321–331. 95, 96, 101

Karatzoglou, A., Baltrunas, L., and Shi, Y. (2013). Learning to rank for recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems*, RecSys '13, pages 493–494. 9

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. (1997). Grouplens : applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3) :77–87. 13

Koren, Y. (2008). 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. 20, 23, 25

Koren, Y. (2010). Factor in the neighbors : Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1) :1–24. 77

Koren, Y. and Bell, R. (2015). Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer. 20, 23, 129

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8) :30–37. 23, 94, 103, 108

Kunaver, M. and Požrl, T. (2017). Diversity in recommender systems–a survey. *Knowledge-Based Systems*, 123 :154–162. 92

Lampropoulos, A. S., Lampropoulou, P. S., and Tsihrintzis, G. A. (2012). A cascade-hybrid music recommender system for mobile services based on musical genre classification and personality diagnosis. *Multimedia Tools and Applications*, 59(1) :241–258. 38

Langer, E., Blank, A., and Chanowitz, B. (1978). The mindlessness of ostensibly thoughtful action : The role of "placebic" information in interpersonal interaction. *Journal of Personality and Social Psychology*, 36 :635–642. 47

Lathia, N., Hailes, S., and Capra, L. (2007). Private distributed collaborative filtering using estimated concordance measures. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 1–8. 71

Lee, D. and Hosanagar, K. (2014). Impact of recommender systems on sales volume and diversity. In *Proceedings of the 35th International Conference on Information Systems*, ICIS '14. 96

Li, S., Zhou, Y., Zhang, D., Zhang, Y., and Lan, X. (2017). Learning to diversify recommendations based on matrix factorization. In *Proceedings of the 15th International Conference on Dependable, Autonomic and Secure Computing, 15th International Conference on Pervasive Intelligence and Computing, 3rd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress, IEEE '17*, pages 68–74. 101

Liang, D., Krishnan, R. G., Hoffman, M. D., and Jebara, T. (2018). Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 689–698. 26

Lin, W., Alvarez, S. A., and Ruiz, C. (2000). Collaborative recommendation via adaptive association rule mining. *Data Mining and Knowledge Discovery*, 6(1) :83–105. 129

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, volume 29. 61

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3) :225–331. 9

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234 :11–26. 26

Lops, P., De Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems : State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer. 12, 29, 50, 98

Lops, P., Jannach, D., Musto, C., Bogers, T., and Koolen, M. (2019). Trends in content-based recommendation. *User Modeling and User-Adapted Interaction*, 29(2) :239–249. 128

Love, M. I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12) :1–21. 138

Lully, V., Laublet, P., Stankovic, M., and Radulovic, F. (2018). Enhancing explanations in recommender systems with knowledge graphs. *Procedia Computer Science*, 137 :211–222. 131, 146

Mahara, T. et al. (2016). A new similarity measure based on mean measure of divergence for collaborative filtering in sparse environment. *Procedia Computer Science*, 89 :450–456. 67

McNee, S. M., Riedl, J., and Konstan, J. A. (2006). Being accurate is not enough : How accuracy metrics have hurt recommender systems. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '06, pages 1097–1101. 92

Meymandpour, R. and Davis, J. G. (2020). Measuring the diversity of recommendations : a preference-aware approach for evaluating and adjusting diversity. *Knowledge and Information Systems*, 62(2) :787–811. 92, 93, 96, 97, 113, 122

Middleton, S. E., De Roure, D. C., and Shadbolt, N. R. (2001). Capturing knowledge of user preferences : ontologies in recommender systems. In *Proceedings of the 1st International Conference on Knowledge Capture*, K-CAP '01, pages 100–107. 48, 50, 53

Middleton, S. E., Shadbolt, N. R., and De Roure, D. C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1) :54–88. 48, 51

Miranda, T., Claypool, M., Gokhale, A., Mir, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*. 37

Moreno, A., Valls, A., Isern, D., Marin, L., and Borràs, J. (2013). Sigtur/e-destination : ontology-based personalized recommendation of tourism and leisure activities. *Engineering applications of artificial intelligence*, 26(1) :633–651. 48, 50, 52, 53

Murakami, T., Mori, K., and Orihara, R. (2007). Metrics for evaluating the serendipity of recommendation lists. In *Annual conference of the Japanese society for artificial intelligence*, pages 40–46. Springer. 46

Musto, C., de Gemmis, M., Lops, P., and Semeraro, G. (2020). Generating post hoc review-based natural language justifications for recommender systems. *User Modeling and User-Adapted Interaction*, 31 :629–673. 130

Musto, C., Lops, P., Basile, P., de Gemmis, M., and Semeraro, G. (2016a). Semantics-aware graph-based recommender systems exploiting linked open data. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, UMAP '16, pages 229–237. 48

Musto, C., Narducci, F., Lops, P., De Gemmis, M., and Semeraro, G. (2016b). Explod : a framework for explaining recommendations based on the linked open data cloud. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 151–154. 5, 131, 135, 143, 146

Musto, C., Narducci, F., Lops, P., de Gemmis, M., and Semeraro, G. (2019). Linked open data-based explanations for transparent recommender systems. *International Journal of Human-Computer Studies*, 121 :93–107. 5, 131, 132, 134, 135, 143, 146, 155, 156

Musto, C., Semeraro, G., de Gemmis, M., and Lops, P. (2017). Tuning personalized pagerank for semantics-aware recommendations based on linked open data. In Blomqvist,

E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., and Hartig, O., editors, *The Semantic Web*, pages 169–183, Cham. Springer International Publishing. 131

Nanou, T., Lekakos, G., and Fouskas, K. (2010). The effects of recommendations' presentation on persuasion and satisfaction in a movie recommender system. *Multimedia Systems*, 16(4–5) :219–230. 144

Natarajan, S., Vairavasundaram, S., Natarajan, S., and Gandomi, A. H. (2020). Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data. *Expert Systems with Applications*, 149 :113248. 115

Nickel, M., Tresp, V., and Kriegel, H.-P. (2012). Factorizing yago : scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 271–280. 61

Nogueira, F. (2014). Bayesian optimization : Open source constrained global optimization tool for python. 108

Nunes, I. and Jannach, D. (2017). A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction*, 27(3) :393–444. 143

Orso, V., Ruotsalo, T., Leino, J., Gamberini, L., and Jacucci, G. (2017). Overlaying social information : The effects on users' search and information-selection behavior. *Information Processing & Management*, 53(6) :1269–1286. 89

Ostuni, V. C., Di Noia, T., Di Sciascio, E., and Mirizzi, R. (2013). Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 85–92. 48, 50

Palumbo, E., Monti, D., Rizzo, G., Troncy, R., and Baralis, E. (2020). entity2rec : Property-specific knowledge graph embeddings for item recommendation. *Expert Systems with Applications*, 151 :113235. 48, 50, 131, 156

Palumbo, E., Rizzo, G., Troncy, R., Baralis, E., Osella, M., and Ferro, E. (2018a). An empirical comparison of knowledge graph embeddings for item recommendation. In *DL4KGS 2018, 1st Workshop on Deep Learning for Knowledge Graphs and Semantic Technologies 2018, co-located with the 15th Extended Semantic Web Conference (ESWC 2018)*, pages 14–20. 5, 48, 94, 109

Palumbo, E., Rizzo, G., Troncy, R., Baralis, E., Osella, M., and Ferro, E. (2018b). Translational models for item recommendation. In *European Semantic Web Conference (ESWC '18)*, pages 478–490. 48, 62, 63, 64, 107, 108, 156

Passant, A. (2010a). dbrec—music recommendations using dbpedia. In *International Semantic Web Conference (ISWC '10)*, pages 209–224. 57

Passant, A. (2010b). Measuring semantic distance on linking data and using it for resources recommendations. In *AAAI spring symposium : linked data meets artificial intelligence*, volume 77, page 123. 113

Pesquita, C., Faria, D., Falcão, A. O., Lord, P., and Couto, F. M. (2009). Semantic similarity in biomedical ontologies. *PLOS Computational Biology*, 5(7) :1–12. 55

Polatidis, N. and Georgiadis, C. K. (2016). A multi-level collaborative filtering method that improves recommendations. *Expert Systems with Applications*, 48 :100–110. 69, 77, 81

Powers, D. M. (2011). Evaluation : from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1) :37–63. 41

Ranwez, V., Ranwez, S., and Janaqi, S. (2011). Subontology extraction using hyponym and hypernym closure on is-a directed acyclic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 24(12) :2288–2300. 133, 140, 148, 155

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens : an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 175–186. 13, 14, 15

Ricci, F., Rokach, L., and Shapira, B. (2015). Recommender systems : introduction and challenges. In *Recommender systems handbook*, pages 1–34. Springer. 9

Rodríguez-García, M. Á., Colombo-Mendoza, L. O., Valencia-García, R., Lopez-Lorca, A. A., and Beydoun, G. (2015). Ontology-based music recommender system. In *Distributed Computing and Artificial Intelligence, 12th International Conference*, pages 39–46. Springer. 48, 50, 55

Ruder, S. (2016). An overview of gradient descent optimization algorithms. 22

Russel, S. and Norvig, P. (2013). *Artificial Intelligence : A Modern Approach*. Pearson Education Limited. 81

Sacharidis, D. (2020). Building user trust in recommendations via fairness and explanations. In *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20, page 313–314. Association for Computing Machinery. 128

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11) :613–620. 31

Sánchez, D., Batet, M., and Isern, D. (2011). Ontology-based information content computation. *Knowledge-Based Systems*, 24(2) :297–303. 145

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science. 23

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295. 14, 94, 103, 108

Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. (2015). Autorec : Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 111–112. 26

Sha, C., Wu, X., and Niu, J. (2016). A framework for recommending relevant and diverse items. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI'16, page 3868–3874. 92, 101

Sieg, A., Mobasher, B., and Burke, R. (2010). Improving the effectiveness of collaborative recommendation with ontology-based user profiles. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 39–46. 48, 50, 51, 53

Sieg, A., Mobasher, B., and Burke, R. D. (2007). Learning ontology-based user profiles : A semantic approach to personalized web search. *IEEE Intelligent Informatics Bulletin*, 8 :7–18. 53

Smyth, B. and Cotter, P. (2000). A personalised tv listings service for the digital tv age. *Knowledge-Based Systems*, 13(2-3) :53–59. 38

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS '12, page 2951–2959. 108

Socher, R., Chen, D., Manning, C. D., and Ng, A. Y. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Proceddings of the 26th International Conference on Neural Information Processing Systems*, volume 1 of *NIPS '13*. 61

Spencer, N. H. (2013). *Essentials of multivariate data analysis*. CRC Press. 16

Steck, H. (2013). Evaluation of recommendations : Rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 213–220. 39, 43, 44, 45, 107, 111

Steck, H. (2018). Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 154–162. 97, 99, 101

Studer, R., Benjamins, V., and Fensel, D. (1998). Knowledge engineering : Principles and methods. *Data & Knowledge Engineering*, 25(1) :161 – 197. 48

Tintarev, N. and Masthoff, J. (2012). Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5) :399–439. 128, 130, 133, 146, 147

Tintarev, N. and Masthoff, J. (2015). Explaining recommendations : Design and evaluation. In *Recommender systems handbook*, pages 353–382. Springer. 47

Trattner, C. and Jannach, D. (2020). Learning to recommend similar items from human judgments. *User Modeling and User-Adapted Interaction*, 30(1) :1–49. 112

Vargas, S. and Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys '11, pages 109–116. 92, 95, 101

Wang, H., Wang, N., and Yeung, D.-Y. (2015). *Collaborative Deep Learning for Recommender Systems*, page 1235–1244. Association for Computing Machinery. 156

Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge graph embedding : A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12) :2724–2743. 60, 62

Wang, Q., Yu, J., and Deng, W. (2019). An adjustable re-ranking approach for improving the individual and aggregate diversities of product recommendations. *Electronic Commerce Research*, 19(1) :59–79. 95, 98, 101

Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28. 61

Weston, J., Bordes, A., Yakhnenko, O., and Usunier, N. (2013). Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, page 1366–1371. 61

Wexelblat, A. and Maes, P. (1999). Footprints : history-rich tools for information foraging. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 270–277. 89

Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1) :79–82. 41

Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J., and Jing, H. (2017). Recurrent recommender networks. In *Proceedings of the tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 495–503. 26

Wu, W., Chen, L., and Zhao, Y. (2018). Personalizing recommendation diversity based on user personality. *User Modeling and User-Adapted Interaction*, 28(3) :237–276. 93, 96, 99, 101

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., and Achan, K. (2020). Product knowledge graph embedding for e-commerce. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, pages 672–680. 124

Xue, H.-J., Dai, X., Zhang, J., Huang, S., and Chen, J. (2017). Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, volume 17 of *IJCAI '17*, pages 3203–3209. 86

Yao, Y. and Harper, F. M. (2018). Judging similarity : A user-centric study of related item recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 288–296. 112, 123

Yigit-Sert, S., Altingovde, I. S., Macdonald, C., Ounis, I., and Özgür Ulusoy (2020). Supervised approaches for explicit search result diversification. *Information Processing & Management*, 57(6) :102356. 93

Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019). Deep learning based recommender system : A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1) :1–38. vi, 26, 27, 124

Zhang, Y. and Chen, X. (2020). Explainable recommendation : A survey and new perspectives. *Foundations and Trends in Information Retrieval*, 14(1) :1–101. 128, 133

Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., and Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 83–92. 129

Zhou, T., Kuscsik, Z., Liu, J. G., Medo, M., Wakeling, J. R., and Zhang, Y. C. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences of the United States of America*, 107(10) :4511–4515. 98

Ziegler, C. N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, page 22. 46, 92, 93, 101

Annexe A

Détails de la configuration de l'optimisation bayésienne pour le réglage des hyper-paramètres des modèles de recommandation comparés

TABLEAU A.1 – Espaces de recherche des hyper-paramètres des modèles de recommandation comparés pendant le processus d'optimisation bayésienne

Modèle de recommandation	Espaces de recherche des hyper-parameters
CBF-TopPopular	$\omega \in [0,1]$
IBCF	$k \in [1,60]$
SVD	$n_{factors} \in [50,200]$, $epoch \in [1,100]$, $\eta \in [0,0.1]$, $\lambda \in [0,0.1]$
KGE	$d \in [10,200]$, $\gamma \in [1,10]$, $\eta \in [0.001,0.1]$, $s_{batch} \in [16,256]$, $epochs \in [100,1000]$
DNN	$s_{batch} \in [16,256]$, $n_{factors} \in [8,128]$, $n_{layers} \in [1,4]$, $\eta \in [0.001,0.1]$, $epochs \in [1,30]$, $n_{neg} \in [0,10]$

TABLEAU A.2 – Les configurations des hyper-paramètres optimaux trouvées pour chaque modèle de recommandation sur chaque jeu de données

Jeu de données	Modèle de recommandation	hyper-parameters optimaux trouvées
Anime	CBF-TopPopular	$\omega = 0.4176$
	IBCF	$k = 3$
	SVD	$n_{factors} = 200$, $epoch = 3$, $\eta = 0.0003$, $\lambda = 0.0626$
	KGE	$d = 147$, $\gamma = 4$, $\eta = 0.001$, $s_{batch} = 116$, $epochs = 232$
	DNN	$s_{batch} = 174$, $n_{factors} = 128$, $n_{layers} = 1$, $\eta = 0.001$, $epochs = 10$, $n_{neg} = 10$
MovieLens-1M	CBF-TopPopular	$\omega = 0.2275$
	IBCF	$k = 2$
	SVD	$n_{factors} = 175$, $epoch = 11$, $\eta = 0.0354$, $\lambda = 0.0813$
	KGE	$d = 199$, $\gamma = 5$, $\eta = 0.0421$, $s_{batch} = 255$, $epochs = 715$
	DNN	$s_{batch} = 256$, $n_{factors} = 66$, $n_{layers} = 1$, $\eta = 0.001$, $epochs = 30$, $n_{neg} = 10$
LibraryThing	CBF-TopPopular	$\omega = 0.1211$
	IBCF	$k = 3$
	SVD	$n_{factors} = 199$, $epoch = 1$, $\eta = 0.0011$, $\lambda = 0.0815$
	KGE	$d = 194$, $\gamma = 6$, $\eta = 0.0435$, $s_{batch} = 21$, $epochs = 439$
	DNN	$s_{batch} = 144$, $n_{factors} = 44$, $n_{layers} = 1$, $\eta = 0.001$, $epochs = 30$, $n_{neg} = 10$