République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de
la **R**echerche **S**cientifique
Université Ibn Khaldoun – Tiaret –

Faculté des Sciences et de la Technologie et Sciences de la Matière

Département des Sciences et de la Technologie

MEMOIRE EN VUE DE L'OBTENTION DU DIPLÔME DE MAGISTER

SPECIALITE: **Informatique**

OPTION : **Système d'Information et de Connaissances**

Présenté par

Mr. KOUADRIA Abderrahmane

SUJET DU MEMOIRE :

La prédiction d'ordre pour le filtrage collaboratif

SOUTENU LE ……2012 Devant Le Jury Composé de :

Mr	A. BALLA	Professeur	ESI	Président
Mr	O. NOUALI	Directeur de recherche	CERIST	Directeur de mémoire
Mme	N. NOUALI-Taboudjemat	Directeur de recherche	CERIST	Co-Directeur
Mr	R. CHALAL	Maître de conférences	ESI	Examinateur
Mr	Y. DAHMANI	Maître de conférences	U. Tiaret	Examinateur

Année Universitaire : 2011/2012

Résumé :

L'objectif des systèmes de recommandation est d'estimer la préférence d'un utilisateur et d'offrir une liste d'articles qui pourraient être privilégiés par un utilisateur donné. Le Filtrage Collaboratif (FC) représente l'une des techniques de recommandation les plus populaires, dont la plupart des méthodes fondent leur approche sur la prédiction de notes pour générer les recommandations. La présente étude, se base sur l'approche de prédiction d'ordre, qui consiste à ordonner correctement les articles selon les goûts des utilisateurs. Nous proposons à cet effet, une adaptation d'une méthode d'ordonnancement de recherche d'information ListMLE pour le filtrage collaboratif, en combinant cette dernière avec la méthode de factorisation matricielle FM. Pour l'évaluation et afin d'amélioré la qualité de la recommandation, nous l'avons expérimenté sur un jeu de données réelles de MovieLens.

Mots Clef :

Systèmes de Recommandation, Filtrage Collaboratif, Apprentissage d'une fonction d'ordonnancement, Factorisation Matricielle.

Abstract:

The purpose of recommendation systems is to estimate user preferences and provide a list of items that might be favoured by a given user. Collaborative filtering (CF) is one of the most popular recommendation techniques, of which methods mainly found their approach on the prediction of notes to generate recommendations. In this research, we rely on the prediction approach of ranking; we propose an adaptation of a ranking method for information retrieval ListMLE for collaborative filtering by combining it with the FM matrix factorisation method. For the evaluation and the improvement of the quality of the recommendation, we have experienced on a real MovieLens dataset.

Keywords:

Recommender systems, collaborative filtering, learning to rank, matrix factorization

ملخص:

الهدف من أنظمة الاقتراحات هو تقدير فضائل المستخدم وتقديم قائمة من المقالات قد تكون مفضلة من طـرف مستخدم معين. الترشيح التعاوني يعتبر إحـدى تقنيات الاقتراحات الأكثر استخداما، والذي معظم طرقه تعتمد على تنبؤ العلامات من أجل خلق و إنتاج الاقتراحات. الدراسة أو البحث الذي بين أيدينا نعتمد فيه على نهج التنبؤ الترتيبي، و الذي يهدف إلى الترتيب الصحيح للمقالات تبعا لأذواق المستخدمين. نقترح لهذا الغرض، تكييف طريقة ترتيب مستعملة في البحث المعلوماتي ListMLE من أجل استخدامها في الترشيح التعاوني، وذلك بالجمع بين هذه الأخيرة و بين طريقة التحليل إلى عوامل مصفوفة FM. لتقييم وتحسين نوعية هذا الترشيح عاينا مجموعة بيانات حقيقية من MovieLens.

كلمات مفتاحية:

أنظمة الإقتراحات، الترشيح التعاوني، تعلم الترتيب، التحليل إلى عوامل مصفوفة

Remerciements

Merci à Allah, nôtre grand seigneur, de nous avoir donné le courage, la volonté et la patience pour mener à terme ce travail.

Merci à mes parents, mes sœurs et frères de m'avoir encouragé et soutenu dans les moments difficiles.

Ensuite, je tiens à présenter mes vifs remerciements et ma profonde gratitude à mon directeur de mémoire : Le docteur Omar Nouali, directeur de recherche au sein du CE.R.I.S.T, pour sa qualité d'encadrement, son esprit scientifique et ses conseils fructueux, ainsi qu'à madame Nadia Nouali - Taboudjemat directeur de recherche (CE.R.I.S.T), co-encadreur, d'avoir contribué par ses aides et conseils à l'accomplissement de ce mémoire.

Mes remerciements vont aux membres du jury pour avoir accepté d'évaluer ce modeste travail.

Enfin, j'exprime mes remerciements à toutes les personnes qui ont, de près ou de loin, contribué à l'aboutissement de ce travail.

Table des matières

Chapitre IV : Les méthodes d'apprentissage d'ordonnancement pour la recherche et le filtrage d'information

Chapitre V: Proposition et évaluation

Liste des figures :

Liste des tableaux

Introduction Générale

Introduction Générale :

Face à la très grande masse d'information disponible sur le Web, il reste très difficile pour un utilisateur de trouver l'information qui l'intéresse. Au lieu de laisser l'utilisateur dépenser son temps à chercher l'information pertinente, les recherches actuelles visent à concevoir des outils qui permettent de lui faciliter cette tâche, en lui faisant parvenir continuellement l'information souhaitée.

Les systèmes de recommandation ou systèmes de filtrage ont pour but de filtrer un flux entrant d'informations d'une manière personnalisée [**Montaner et al. 03**]. Les systèmes de recherche d'information exigent à l'utilisateur de formuler systématiquement sa requête en utilisant des mots-clés. Le résultat retourné contient généralement un nombre important de documents non souhaités. Il appartient à l'utilisateur de procéder à une sélection manuelle des documents estimés pertinents, ce qui est une tâche pénible et fastidieuse. Par contre, un système de filtrage d'information « achemine des documents qui se présentent vers des groupes de personnes, en se basant sur leurs profils à long terme », et élaborés à partir de données d'apprentissage [**Croft 93**].

Le filtrage d'information est considéré aussi comme une expression utilisée pour décrire une variété de processus se rapportant à la fourniture de l'information adéquate aux personnes qui en ont besoin [**Belkin 92**]. Il existe plusieurs grandes familles de ce type de systèmes :

– Le filtrage basé sur le contenu (filtrage cognitif) : Un document est indexé par thèmes (généralement sous forme de termes) puis comparé aux thèmes intéressants l'utilisateur à qui on recommande ceux qui sont les plus proches. L'ensemble des termes donnés explicitement par l'utilisateur constitue son profil qui exprime son centre d'intérêt.

– Le filtrage collaboratif : La sélection des documents proposés à un utilisateur se base sur les annotations et les commentaires attribués par d'autres utilisateurs aux documents.

– Le filtrage hybride : combine les deux approches citées précédemment.

Dans ce mémoire, nous nous intéressons à l'approche collaborative, pour laquelle, nous considérons des ensembles d'utilisateurs et d'articles, où chaque utilisateur exprime ses préférences pour un nombre d'articles, en général sous forme de notes entières. L'objectif est de déterminer pour chaque utilisateur un sous-ensemble d'articles non notés correspondant à ses goûts. L'aspect collaboratif vient du fait que pour chaque utilisateur, les recommandations sont générées à partir de toutes les notes de la base, et non seulement de celles qu'il a fourni.

L'idée intuitive est que lorsque deux utilisateurs se partagent des goûts similaires sur un certain nombre d'articles, alors il est probable qu'ils aient les mêmes avis sur d'autres articles

non notés. En faisant collaborer ainsi les différents utilisateurs, le filtrage collaboratif exploite cette intuition pour générer les recommandations.

La prédiction de notes est l'approche la plus étudiée en filtrage collaboratif. La littérature fait état de plusieurs travaux d'inspirations diverses, telles que la classification supervisée ou non supervisée, la régression, la réduction dimensionnelle ou encore les modèles probabilistes. En ramenant le problème de la recommandation à une tâche de prédiction de notes, toutes ces méthodes présentent un même objectif: prédire les notes le plus précisément possibles. Pourtant, du point de vue de la recommandation, la manière dont les articles sont ordonnés est plus importante que les notes elles-mêmes. En effet, un utilisateur attend avant tout d'un système de recommandation qu'il lui suggère les articles les plus intéressants, la prédiction des notes manquantes n'étant qu'une stratégie parmi d'autres pour y parvenir [**Jean el al 07**].

Afin d'améliorer la qualité de la recommandation, nous proposons une solution qui se base sur la prédiction d'ordre qui permet d'ordonner correctement les articles, plutôt que d'essayer de prédire des notes le mieux possible. Nous prédisons des scores respectant au mieux les préférences des utilisateurs.

Ce mémoire est structuré en cinq chapitres :

-**Chapitre I :** Dans ce chapitre nous introduisons le contexte et la problématique de notre recherche, ainsi que les objectifs mentionnés afin de contourner cette problématique.

-**Chapitre II :** Nous présentons au sein de ce chapitre un état de l'art sur les systèmes de filtrage d'information tels que leurs principes, fonctionnements et caractéristiques. De plus, nous présentons la modélisation du profil utilisateur ainsi que les différentes formes de représentation et d'acquisition. Nous énonçons les différents types de filtrage.

- **Chapitre III :** Le troisième chapitre concerne les systèmes de filtrage collaboratif. Il présente les origines, le principe de fonctionnement, l'architecture générale de ces systèmes, ainsi que leurs avantages et Inconvénients. Il présente aussi les différents algorithmes de prédiction utilisés pour ce type de filtrage.

-**Chapitre IV :** Nous présentons les méthodes d'apprentissage d'ordonnancement pour la recherche et le filtrage d'information.

- **Chapitre V :** Dans ce dernier chapitre, nous mettons en relief notre proposition qui répond à la problématique ainsi que l'évaluation de cette proposition sur le jeu de données réelles de MovieLens.

Enfin, nous terminons par une conclusion qui récapitule notre contribution et ouvre la voie à des travaux futurs.

Chapitre I

Contexte de recherche et objectifs

1. Présentation du contexte :

Les systèmes de recommandations ont été introduits comme une technique informatique intelligente pour traiter le problème de la surcharge de l'information. Ils peuvent être utilisés pour fournir efficacement des services personnalisés dans plusieurs domaines tels que la recherche documentaire, le commerce électronique, les loisirs …etc.

Ces systèmes sont des types spécifiques du filtrage d'information qui présente des éléments d'information qui sont susceptibles d'intéresser un utilisateur. Les éléments d'information à recommander sont divers : films, musique, livres, images, site Web, émissions télévisées … etc.

Figure I.1 *Les systèmes de recommandations*

Différentes formes de recommandations existent, suivant les données à recommander, les informations disponibles et bien évidemment l'objectif visé. Parmi les formes de recommandations les plus répandues on trouve :

1.1 Recommandation éditoriale :

La recommandation éditoriale est généralement utilisée lorsqu'aucun autre système de recommandation n'est présent ou encore lorsque le système n'a aucune connaissance sur le visiteur du site. L'objectif principal de la forme de recommandation est d'attirer rapidement l'œil de l'utilisateur et le persuader afin de s'intéresser à un ou plusieurs articles et ainsi provoquer l'acte d'achat.

Les recommandations ne sont pas personnalisées aux utilisateurs mais qualifiées de génériques [**Thomas 11**].

Exemple : de nombreux sites Internet mettent en avant des nouveautés, des articles fréquemment achetés, des promotions, …etc.

1.2 Recommandation sociale :

Dans cette méthode, les recommandations sont réalisées par des utilisateurs différents de l'utilisateur actif.

Exemple : Il peut s'agir d'internautes navigants sur YouTube[1] ou Flixter[2], ou de consommateurs d'Amazon. Ce type de recommandations peut être basé sur le principe de bouche à oreille. Sur Flixster, les utilisateurs ont la possibilité de faire des recommandations à l'intérieur de leur réseau social en transmettant leurs commentaires et leurs appréciations à leurs amis.

Une autre solution utilisée par certains sites est de permettre aux utilisateurs de créer des listes de 'coups de cœur' qui accompagnent ensuite les profils des produits, comme sur Amazon, ou accompagnent les profils des utilisateurs, comme sur Youtube [**Damien 11**].

1.3 Recommandation contextuelle :

Le principe de la recommandation contextuelle est de proposer des articles proches de celui consulté. Une première démarche consiste en le rapprochement d'articles pour lequel les caractéristiques descriptives sont similaires.

Exemple : On peut sélectionner les articles d'un même univers, auteur, réalisateur, compositeur, couleur, etc. Ensuite, une approche plus complexe, telle que celle d'Amazon, qui recommande une liste d'articles appréciés par d'autres utilisateurs ayant également apprécié l'article consulté ; Flickr[3] ou IMDb[4] utilisent les étiquettes (ou tags) pour rapprocher les articles et enfin, Genius, système de recommandation d'Apple qui permet d'obtenir des suggestions d'applications directement sur son iPhone en supprimant à sa propre initiative les mauvaises recommandations, ou de recommander l'achat de musiques ou de films sur l'Itunes Store [**Thomas 11**].

1.4 Recommandation personnalisée :

L'objectif de la recommandation personnalisée est de recommander à un utilisateur particulier, les articles ou les services les plus susceptibles de l'intéresser. Ces recommandations sont établies en fonction du comportement d'achat ou de navigation de l'utilisateur courant. Les enjeux de la recommandation personnalisée sont nombreux et bénéficient autant à l'utilisateur qu'au fournisseur de services.

Exemple : Le système de recommandation peut être considéré comme 'remplaçant du vendeur', qui généralement capte et capitalise les préférences des utilisateurs afin de les guider dans leurs choix. Ensuite, l'utilisateur lui-même est avantagé par un gain de temps et

1 http: //www.youtube.com
2 http: //www.flixster.com
3 http: //www.flickr.com
4 http: //www.imdb.com

une découverte d'articles souvent cachés auxquels il n'aurait pas pensé. Enfin, du côté fournisseur de la recommandation, la satisfaction et la fidélisation des clients viennent souvent engendrer des bénéfices et guider la stratégie marketing de l'entreprise [**Damien 11**].

Actuellement, la recommandation personnalisée représente le domaine le plus attractif pour l'utilisateur et le plus intéressant économiquement pour le fournisseur de recommandations [**Tintarev et al 07**].

Plus concrètement, la recommandation personnalisée a pour objectif de filtrer des articles afin de ne conserver que les plus pertinents pour un utilisateur donné. L'idée principale est de prédire l'opinion qu'un utilisateur portera sur les articles qu'il ne connaît pas encore, afin de ne lui proposer que ceux qu'il est susceptible d'apprécier, ou tout du moins, qui auront une grande chance de l'intéresser.

Une définition plus formelle de la recommandation est donnée par Adomavicius et Tuzhilin [**Adomavicius et al 05**].

Définition formelle:

Soit U l'ensemble des utilisateurs, A l'ensemble des articles qui peuvent être recommandés, R l'ensemble ordonné et $f : U \times A \rightarrow R$ la fonction qui prédit l'intérêt que portera l'utilisateur $u \in U$ à l'article $a \in A$. tel que pour chaque utilisateur $u \in U$, le système de recommandation sélectionne l'article $a' \in A$ qui maximise l'intérêt de l'utilisateur u :

$$\forall u \in U, a_u^{'} = \arg\max_{a \in A} f(u,a)$$

L'objectif ou l'intérêt d'un utilisateur pour un article (la fonction *f(u, i)*) est généralement représenté par une note indiquant l'appréciation que l'utilisateur porterait sur un article. Afin de deviner cet intérêt, des connaissances sur l'utilisateur en question sont nécessaires. Les goûts connus des utilisateurs sont généralement caractérisés par leurs appréciations portées sur les contenus déjà consultés. Ces informations sont regroupées dans une matrice appelée matrice utilisateur-article. Le tableau I.1 présente un exemple de matrice binaire (aime/n'aime pas).

	Article1	**Article2**	**Article3**	**Article4**
Utilisateur1			?	
Utilisateur2		?		?
Utilisateur3	?			?
Utilisateur4			?	

Tableau I.1 *Matrice binaire (aime/n'aime pas)*

Ces informations peuvent également se mesurer sur un nombre plus élevé de classes : « à voter 1, 2, 3, 4 ou 5 étoiles » (tableau I.2). L'objectif du système de recommandation est de prédire les mesures d'intérêts utilisateur-article manquantes. En d'autres termes, remplir les cases vides ***f(u, a)*** de la matrice utilisateur-article en évaluant si l'article ***A*** intéresse l'utilisateur ***U***.

	Article1	**Article2**	**Article3**	**Article4**
Utilisateur1	**5**	**2**	**?**	**4**
Utilisateur2	**1**	**?**	**5**	**?**
Utilisateur3	**?**	**3**	**1**	**?**
Utilisateur4	**4**	**5**	**?**	**2**

Tableau I.2 *Matrice utilisateur-article de votes*

2. Problématique :

Le filtrage collaboratif est l'une des approches les plus populaires des systèmes de recommandation. Leur principe est de suggérer de nouveaux articles ou de prédire l'utilité des articles inconnus pour un utilisateur donné, en se basant sur les opinions et les évaluations d'un groupe d'utilisateurs afin de proposer un ou plusieurs articles.

Une fonction très importante de la plupart des systèmes de recommandation est la génération de la liste des N meilleurs articles pour chaque utilisateur, afin de faire des recommandations personnalisées, qui consiste essentiellement à résoudre un problème d'ordonnancement. Pour ordonner les articles, la plupart des algorithmes de filtrage collaboratif formulent cela comme un problème de prédiction de notes qui permet de prédire tout d'abord les notes potentielles d'un utilisateur sur les articles, puis ordonner les articles selon les évaluations prévues [**Liu et al 08, 09**]. Cependant, une plus grande précision dans la prédiction de notes ne conduit pas nécessairement à la meilleure efficacité d'ordonnancement comme l'illustre l'exemple simple suivant. Soient [2, 3] les notes de deux articles A et B, r_1 = [2.5, 3.6] et r_2 = [2.5, 2.4] deux vecteurs de prédictions obtenus par deux méthodes différentes. Bien que r_1 et r_2 soient équivalents en terme d'erreur carrée (les deux sont égales à 0.5 + 0.6), seule r_1 prédit l'ordre correctement, puisque le score qu'elle attribue à B est supérieur à celui de A. Par contre r_2 n'assure pas le bon ordonnancement [**Jean el al 07**].

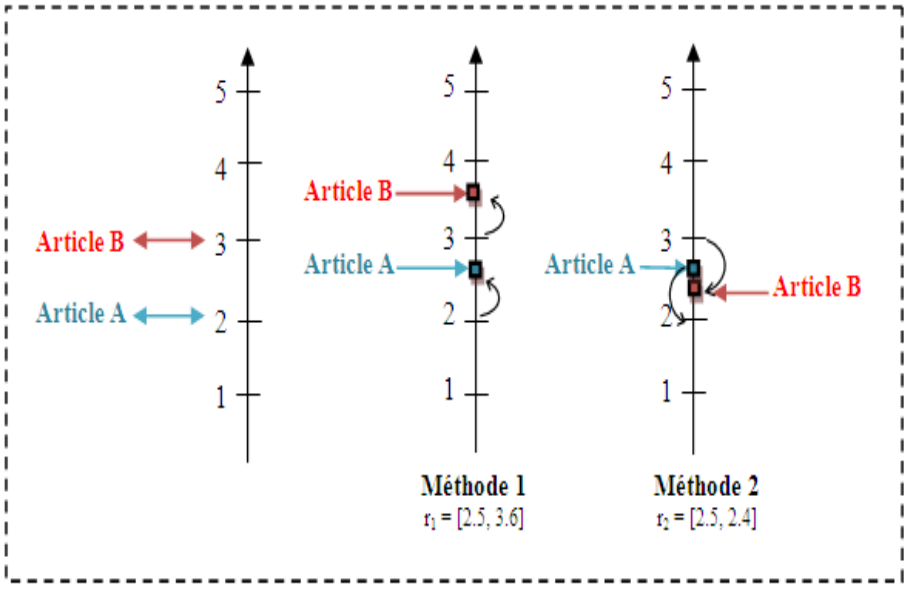


Figure I.2 *Exemple d'ordonnancement*

Ces dernières années, et comme de plus en plus les bases standards de filtrage collaboratif avec les jugements de pertinence sont disponibles, les méthodes à base d'apprentissage d'ordonnancement supervisé (Learning To Rank methods) ont donné lieu à différentes études et développements [**Clémençon et al 05**], [**Amini 07**], [**Cossock et al 08**], par exemple leurs utilisation pour apprendre automatiquement une fonction d'ordonnancement efficace à partir des données d'apprentissage [**Burges et al 08**], [**Cao et al 07**], [**Crammer et al 02**], [**Freund et al 03**].

L'apprentissage d'une fonction d'ordonnancement peut être vu comme l'apprentissage d'une fonction score : une fonction à valeurs réelles, qui prend en entrée un élément d'un ensemble à ordonner. L'ordre est ensuite prédit en triant les éléments selon les scores croissants ou décroissants [**Nguyen 09**].

3. Objectifs :

Dans ce mémoire, nous nous intéressons à l'application des méthodes d'apprentissage d'ordonnancement pour le filtrage collaboratif. L'objectif est de proposer une adaptation d'une méthode d'ordonnancement de recherche d'information pour la tâche de prédiction d'ordre au filtrage collaboratif, qui consiste à ordonner correctement les articles plutôt que de prédire correctement leurs notes. Cette approche combine la méthode de factorisation matricielle (MF) avec une méthode d'ordonnancement de l'approche listewise. Une liste ordonnée d'articles est obtenue en réduisant au minimum une fonction d'erreur qui représente l'incertitude entre la liste d'apprentissage en entrée et la liste résultante en sortie selon la méthode proposée.

Nous avons validé expérimentalement notre proposition sur la base standard de filtrage collaboratif MovieLens, afin de comparer sa performance aux méthodes de factorisation matricielle et CoFiRank-NDCG. Les comparaisons sont faites dans le cadre de prédiction d'ordre, pour la tâche de recommandation hors ligne.

Chapitre II

Systèmes de filtrage d'information

1. Introduction :

L'accès à une information pertinente, adaptée aux besoins de l'utilisateur, est un enjeu capital dans le contexte actuel, caractérisé par une prolifération massive de ressources hétérogènes. L'accès à ces informations peut se faire de manière délibérée et instantanée via une requête exprimée par l'utilisateur, mais quand ce besoin est permanent, il est inutile de demander à l'utilisateur de reprendre à chaque fois sa requête. Le processus qui permet de répondre à cette attente est le filtrage d'information [**Tebri 04**]. Le rôle de ce dernier est d'augmenter la quantité d'informations pertinentes, collectées à partir de différentes sources.

Un système de filtrage d'informations achemine l'information aux utilisateurs, en se basant sur leurs profils. Un tel profil représente les centres d'intérêts et les préférences de l'utilisateur.

L'objectif de ce chapitre est de présenter les systèmes de filtrage d'information d'une façon générale. Pour cela, nous définissons tout d'abord, les concepts de base du filtrage d'information. Nous présentons ensuite les grandes familles de filtrage d'information : basé sur le contenu, ainsi que collaboratif et hybride.

2. Filtrage d'information :

Le filtrage d'information est le processus permettant à partir d'un large volume d'informations dynamiques, d'extraire et de présenter les seuls documents intéressant un utilisateur ou un groupe d'utilisateurs ayant des centres d'intérêts relativement semblables appelés profils [**Belkin 92**].

Le but principal d'un système de filtrage d'information (SFI), est de filtrer un flux entrant d'informations de façon personnalisée pour chaque utilisateur, tout en s'adaptant en permanence à son besoin d'information. Autrement dit, dans l'objectif de personnaliser la recherche d'information dans un domaine d'application particulier, un système de filtrage collecte, sélectionne, classifie et suggère à l'utilisateur les informations qui répondent vraisemblablement à ses intérêts à long terme.

Le filtrage peut être vu comme la sélection d'informations pertinentes sur un flux entrant, tel qu'il est indiqué dans la figure II.1. Pour sélectionner les informations pertinentes du «Flux entrant», le système fait une «prédiction». Cette prédiction s'appuie sur le «profil» de l'utilisateur et se termine par une prise de décision : «recommander» ou «ne pas recommander». **[Gallardo 05]**.

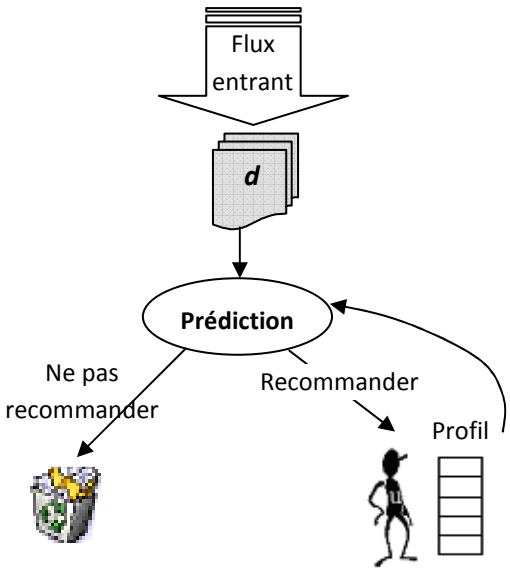


Figure II.1 – *Filtrage d'information* **[Gallardo 05].**

Les utilisateurs décrivent leurs centres d'intérêt eux-mêmes, interprétés par un «profil». Ce dernier explique le besoin d'information en permanence de l'utilisateur.

Une des premières approches à mettre en œuvre la notion de filtrage d'information est la DSI (Dissémination Sélective de l'Information) vers les années 60. Elle consistait à filtrer l'information produite par des scientifiques, dans le but de maintenir leurs paires continuellement informés des nouveautés relatives à leurs domaines de spécialisation.

A partir de 1982, Denning a introduit le principe de filtrage de message email en utilisant des techniques basées sur l'organisation des mail-boxes et nécessitant la coopération des différents usagers. Par la suite la notion de filtrage a été étendue aux articles de presse et articles diffusés sur Internet **[Zemirli 05]**.

3. Caractéristiques d'un système de filtrage :

Un système de filtrage d'information est caractérisé principalement par ce qui suit :

- Il est destiné à des informations peu ou pas structurées contrairement aux bases de données qui utilisent des documents très structurés;
- Il diffuse en général des informations textuelles, mais peut également gérer d'autres types d'information tel que : image, son ou vidéo;
- Le filtrage concerne un flux d'information en provenance d'une ou plusieurs sources extérieures *(ex news)* ou adressé directement à l'utilisateur *(ex. email)*;
- Le filtrage doit prendre en compte le profil de l'usager qui spécifie au système ses caractéristiques;

- Il exploite un grand volume de données entrants, transmis par de sources distantes ;
- Il se base principalement sur le profil de l'utilisateur ou d'un groupe d'utilisateurs ;
- Il ne diffuse que les informations en adéquation avec le profil l'utilisateur.

4. Fonctionnement d'un système de filtrage :

Un système de filtrage d'information (SFI) pourrait être défini comme étant la technique qui vise à filtrer les informations pertinentes d'un flux pour les faire acheminer vers des groupes de personnes, en se basant sur leurs profils (intérêts) à long terme, contrairement à la recherche d'information où l'utilisateur à l'aide d'une requête sélectionne l'information pertinente à partir du flux.

La figure ci-dessous schématise un modèle de filtrage d'information. Il débute avec des personnes (les utilisateurs du système de filtrage d'information) qui ont des objectifs ou des intérêts relativement stables, à long terme ou périodiques. Des groupes, aussi bien que des personnes peuvent être caractérisés par de tels buts. Ceci amène à des besoins réguliers d'information qui peuvent évoluer lentement au cours du temps au fur et à mesure que les conditions, objectifs et connaissances changent. De tels intérêts engagent les utilisateurs dans un processus relativement passif de recherche d'information. Ce processus est réalisé à travers la représentation des besoins en information par des profils ou des requêtes destinés au système de filtrage d'information [**Berrut 03**].

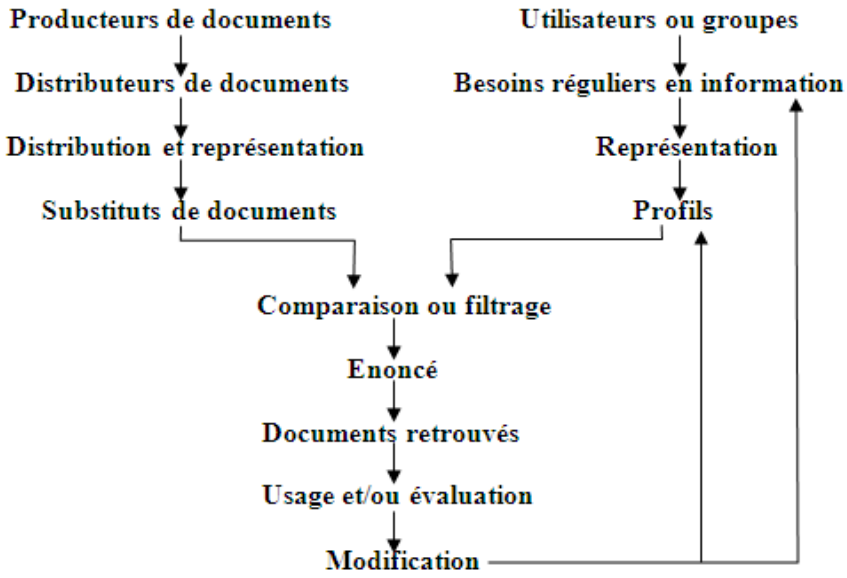


Figure II.2 *Modèle général pour le filtrage d'information* **[Belkin 92]**

En contrepartie les producteurs des documents diffusent leurs produits dès qu'il est généré, ainsi une représentation adéquate sera alors associée à chaque document diffusé. L'appariement est effectué par la suite entre les documents représentés et les profils. Seuls les documents dont les thèmes répondent au profil de l'utilisateur seront sélectionnés, puis évalués par celui-ci. Sur la base de cette évaluation, le profil de l'utilisateur sera mis à jour en conséquence.

5. Profil utilisateur :

5.1 Définition :

On appelle profil utilisateur toute structure qui permet de modéliser et de stocker les données caractérisant l'utilisateur, où le concept profil est lié directement à l'utilisateur [**Tamine et al 08**].

Ces données représentent particulièrement les centres d'intérêt de l'utilisateur, auxquels s'ajoutent ses informations personnelles. D'autres éléments peuvent être intégrés aux profils tels que sa familiarité avec le sujet de recherche, son domaine professionnel et le contexte de la recherche. Notons que le concept de profil est un concept très important dans tout le processus d'accès à l'information. La modélisation de ce concept a pour objectif fondamental de représenter puis faire évoluer les besoins en information de l'utilisateur à court et moyen terme.

5.2 Modélisation d'un profil utilisateur :

L'objectif principal de la modélisation du profil utilisateur est de sélectionner les informations les plus appropriées qui traduisent les intérêts de celui-ci. Cette modélisation consiste à désigner une structure pour stocker toutes les informations qui caractérisent l'utilisateur et qui décrivent principalement ses centres d'intérêts en plus d'autres informations relatives à ses préférences, le contexte dans lequel il travaille, le but escompté et les objectifs de sa recherche, ses traits individuels, son background et son expérience, … etc.

5.3 Approches de représentation du profil utilisateur :

Il existe principalement trois approches de représentation : ensembliste, sémantique et multidimensionnelle.

5.3.1 Représentation ensembliste :

Appelée aussi représentation vectorielle. Le profil est alors représenté par un ou plusieurs vecteurs définis dans un espace de termes et dont les coordonnées correspondent à

leurs poids respectifs. L'utilisation de plusieurs vecteurs pour représenter le profil permet la prise en compte de la diversité des centres d'intérêts et de leur évolution à travers le temps.

5.3.2 Représentation sémantique :

C'est une représentation hiérarchique, qui vise à la construction d'une hiérarchie de concepts ou d'une ontologie personnelle. Chaque catégorie de la hiérarchie représente la connaissance d'un domaine d'intérêt de l'utilisateur avec un certain score ou degré.

La représentation du profil met en évidence, dans ce cas, les relations sémantiques entre les concepts qui le décrivent. La représentation est essentiellement basée sur l'utilisation d'ontologies ou réseaux sémantiques probabilistes [**Sieg 07**]

5.3.3 Représentation multidimensionnelle :

Le profil est structuré selon un ensemble de dimensions, représentées selon divers formalismes et peut contenir plusieurs types d'informations, telles que les données démographiques, centres d'intérêts, objectif, information historique et autres. Chaque type d'information est une dimension dans le modèle multidimensionnel [**Kien 06**].

5.4 Construction et acquisition de profil :

Le profil utilisateur est construit en utilisant des techniques de personnalisation de l'information. Nous distinguons deux types de personnalisation: la personnalisation explicite et implicite.

5.4.1 Personnalisation explicite (Filtrage explicite) :

Elle consiste à interroger l'utilisateur directement sur ses besoins. Le contenu du profil est construit selon les informations fournies volontairement par l'utilisateur au moment de son abonnement (c.-à-d. de son enregistrement sur le site). L'utilisateur décrit, de manière explicite, ses renseignements personnels et ses centres d'intérêts, via un formulaire par exemple. Ceci est appelé modélisation déclarative ou statique des préférences de l'utilisateur. Le filtrage explicite présent l'inconvénient d'exiger une participation active de l'utilisateur afin de décrire ses préférences. Par ailleurs, l'utilisateur hésite à introduire la vérité de ses informations personnelles.

5.4.2 Personnalisation implicite (Filtrage implicite):

Dans la personnalisation implicite, le contenu du profil est construit en fonction des habitudes de l'utilisateur de sa navigation tout en analysant ses clics. Les formulaires d'achat,

les abonnements électroniques, les demandes de documentation, sont comme beaucoup d'autres occasions, une source de recueillements de quelques informations sur l'utilisateur. La personnalisation implicite fournit dynamiquement (évolutif et extensible) la modélisation des préférences des utilisateurs, ainsi qu'elle présente l'avantage de la non nécessité de participation active de l'utilisateur.

5.5 Evolution du profil utilisateur :

L'évolution des profils désigne leur adaptation à la variation des centres d'intérêts des utilisateurs qu'ils décrivent, et par conséquent, de leurs besoins en information au cours du temps. L'évolution y est d'avantage abordée comme un problème de représentation de la diversité des domaines d'intérêts de l'utilisateur en expriment des techniques de classification ou heuristiques liées à la notion de cycle de vie artificielle d'un centre d'intérêt [**Chen 98**].

6. Types de filtrage d'information :

Il existe trois grandes approches de filtrage : basé sur le contenu, collaboratif et hybride. Le filtrage basé sur le contenu compare les nouveaux documents aux préférences de l'utilisateur (profil), et recommande ceux qui sont les plus proches. Le filtrage collaboratif compare les utilisateurs entre eux sur leurs jugements des documents passés et crée des communautés où chaque utilisateur reçoit les documents appréciés par ses voisins dans la communauté. Le filtrage hybride combine le filtrage basé sur le contenu et le filtrage collaboratif afin d'exploiter au mieux les avantages des deux approches.

6.1 Filtrage basé sur le contenu « cognitif » *:*

Le filtrage d'information basé sur le contenu des documents, est l'approche la plus anciennement utilisée dans le domaine du filtrage, elle trouve ses racines dans le monde de la recherche d'information. Cette approche est basée sur un processus à deux phases :

- La première étant la sélection de ressources correspondant au profil de l'utilisateur,
- La deuxième étant la mise à jour du profil de l'utilisateur après retour de pertinence des résultats.

Les documents entrants sont indexés par des termes et comparés au profil utilisateur représenté sous la forme d'un ensemble de thèmes décrivant ses centres d'intérêts. Le résultat de la comparaison est le filtrage des documents qui correspondent parfaitement à son profil. Dans ce genre de système, chaque utilisateur opère indépendamment des autres comme s'il était entrain de faire une recherche d'information classique **[Gallardo 05].**

La figure II.3 présente un processus de filtrage d'information basé sur le contenu, où la décision de sélection d'un document donné, est calculée en rapprochant les thèmes énoncés par l'utilisateur comme constituant son profil, et les thèmes extraits des documents par un processus d'indexation.

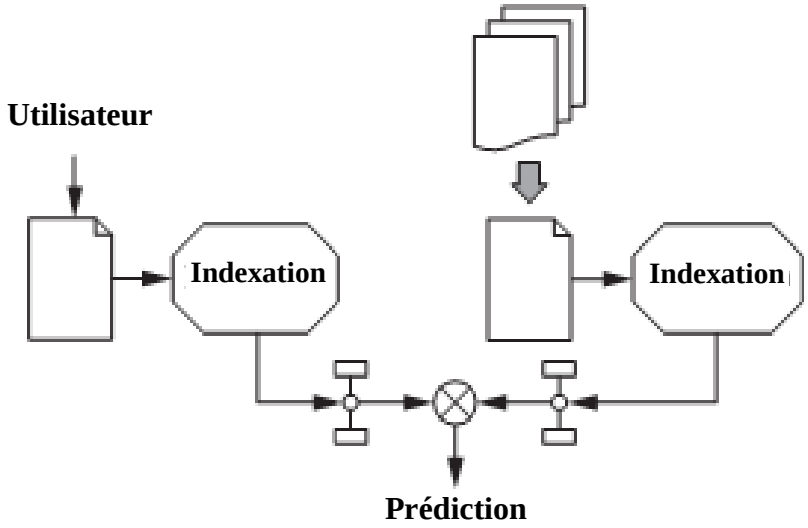


Figure II.3 *Filtrage basé sur le contenu* [**Berrut 03**]

Dans le filtrage cognitif, la sélection des documents est basée sur l'analyse de leur contenu, le document n'est alors proposé à l'utilisateur que s'il répond aux même thèmes intéressant celui-ci. Ce type de filtrage peut être perçu, comme un système de recherche d'informations qui se sert de la requête pour extraire les informations pertinentes du corpus interrogé, parallèlement, le filtrage par le contenu, emploie plutôt un profil utilisateur qui est une sorte de requête dynamique à long terme pour filtrer en permanence un flot évolutif de documents entrants. Les fonctionnalités principales du filtrage par le contenu visent la sélection des documents pertinents par rapport au profil et à mettre à jour ce dernier en fonction du retour de pertinence émis par l'utilisateur sur les documents qu'il a reçu [**Maltz 95**]. La mise à jour se ferra par l'insertion des nouveaux thèmes abordés par les documents jugés pertinents [**Berrut 03**].

Cette approche souffre de certaines limitations, parmi lesquelles :

- ✓ L'incapacité à traiter d'autres critères de pertinence que les critères strictement thématiques posent également problème. Le filtrage des documents basé sur le contenu ne permet pas d'intégrer d'autres facteurs de pertinence que le facteur thématique. Pourtant il existe de nombreux autres facteurs de pertinence comme par exemple l'adéquation entre le public visé par l'auteur et l'utilisateur, ou encore la

qualité scientifique des faits présentés, la fiabilité de la source d'information, le degré de précision des faits présentés, … etc;

- ✓ La difficulté à indexer les documents multimédia (images, vidéos, … etc) et donc la difficulté à recommander ce genre de documents ;
- ✓ L'effet « entonnoir » : les besoins de l'utilisateur sont de plus en plus spécifiques, ce qui l'empêche d'avoir une diversité de sujets. Pire encore, un nouvel axe de recherche dans un domaine bien précis peut ne pas être pris en compte, car il ne fait pas parti du profil explicite de l'utilisateur.

6.2 Filtrage collaboratif « Filtrage social »:

Le filtrage collaboratif est une technique qui s'inscrit dans la recherche sociale d'information. Il regroupe les méthodes pour le développement du système de recommandation en se fondant sur les préférences et les apports de la communauté d'utilisateurs du système. Le filtrage collaboratif permet de contourner certaines difficultés liées au système de filtrage par le contenu et au système de recherche d'information.

Le filtrage collaboratif consiste à filtrer les documents du flux entrant, en se basant sur l'opinion que chaque utilisateur de la communauté a porté dessus. Tout document qu'il l'aura alors jugé intéressant, sera diffusé à l'ensemble des utilisateurs ayant eu des opinions similaires par le passé. Le profil ne se présente plus sous la forme classique d'un ensemble de thème pondéré comme c'est le cas dans le filtrage par le contenu, mais plutôt sous forme d'un ensemble d'évaluations sur des documents faites dans le passé par l'utilisateur. Le profil est modifié au cours du temps à partir des nouvelles évaluations de celui-ci.

La sélection d'un document se base sur l'avis exprimé par les utilisateurs quant à sa pertinence, cet avis intègre souvent une dimension de subjectivité [**Gallardo 05**], dû en partie aux goûts et préférences de chacun et à leurs niveaux d'expertise par rapport à la thématique abordée dans le document évalué, par ailleurs les utilisateurs ayant des avis similaires vont inévitablement se regrouper. Le système emploie des méthodes statistiques pour faire des prévisions selon les intérêts des utilisateurs. Ces prévisions vont servir pour proposer un document à un utilisateur selon la corrélation avec le profil des utilisateurs de son voisinage. L'utilisateur après avoir reçu le document, l'évalue à son tour en lui attribuant un score en fonction de son appréciation sur sa pertinence ; le système réajuste alors automatiquement le profil de l'utilisateur.

6.3 Filtrage hybride :

Le filtrage hybride est la combinaison des deux approches intégrant des fonctionnalités du filtrage collaboratif et du filtrage basé sur le contenu produisant ainsi une synergie.

En général, dans cette approche, les profils sont orientés contenu, et la comparaison entre ces profils donne lieu à la formation de communautés permettant le filtrage collaboratif. La façon dont ces deux approches s'articulent varie, mais les deux ont des atouts complémentaires. Les documents peuvent alors être acheminés vers d'autres utilisateurs en exploitant les critères de filtrage collaboratif (évaluation) et ceux basés sur le contenu (contenu des documents).

7. Recherche d'information versus Filtrage d'information :

La recherche d'information est étroitement liée au filtrage de l'information dans le sens où ils ont le même but qui est de retrouver l'information pertinente pour un certain utilisateur. La distinction entre la recherche d'information et le filtrage d'information n'est souvent pas claire [**Nouali 97**].

Un Système de Recherche d'Information (SRI) capitalise un volume important d'information et offre des outils permettant de localiser les informations pertinentes relatives à un besoin en information d'un utilisateur, exprimé à travers une requête. L'utilisateur joue un rôle particulièrement actif dans ces systèmes, du fait que la recherche est réalisée sur la base d'une requête, qu'il définit explicitement et qu'il soumet au SRI. Ce dernier retourne une liste ordonnée de documents susceptibles de répondre à cette requête.

Par contre les systèmes de filtrage d'information s'inscrivent dans le cadre plus général des systèmes d'accès personnalisé à l'information. Ces systèmes intègrent l'utilisateur d'une manière implicite, en tant que structure informationnelle dans le processus de sélection de l'information pertinente. Cette structure est souvent représentée à travers la notion de profil. Un profil peut comporter différents types d'information sur l'utilisateur, telles que ses préférences, ses centres d'intérêts, ses habitudes de recherche, etc. [**TEBRI 04**]

Bien que proches dans un certain nombre de fonctionnalités, recherche d'information et filtrage d'information s'opposent en un certain nombre de points, cette comparaison est résumée dans le tableau suivant :

	Recherche d'information	**Filtrage d'information**
Objectif	Trouver l'information Recherchée	Filtrer l'information non désirée
Livraison	Corpus statique ; sur demande (requête temporaire et adverbiale)	Flux dynamique ; livraison si compatibilité entre le besoin et le profil de l'utilisateur
Persistance	Des besoins à court terme	Des intérêts à long terme
Personnalisation	Non personnalisé	Profil d'utilisateur requis
Analyse du contenu	Utilise souvent des mots-clés	Différents et multiples dispositifs utilisés (ex : nombre d'occurrences des mots clés)
Fonctionnalités	Non personnalisé Non adaptatif au changement du profil de l'utilisateur Non dynamique A court terme	Personnalisé S'adapte au changement du profil de l'utilisateur Filtre dynamiquement l'information entrante A long terme

Tableau II.1 *Tableau comparatif des principes de recherche d'information et de filtrage d'information.*

8. Conclusion :

Le filtrage d'information se définit comme étant le processus qui vise à filtrer les informations d'un flux pour ne faire parvenir que celles qui intéressent l'utilisateur, contrairement à la recherche d'information où l'utilisateur à l'aide d'une requête sélectionne l'information pertinente à partir du flux.

Dans ce chapitre, nous avons étudié le principe de fonctionnement d'un système de filtrage d'information, ses caractéristiques ainsi que les différents types de filtrage (basé sur le contenu, hybride, collaboratif). Le chapitre suivant sera consacré au filtrage collaboratif qui a fait l'objet de notre contribution. L'intérêt du filtrage collaboratif est de déterminer pour chaque utilisateur une liste d'articles à recommander correspondant à ses goûts. Cette recommandation se base sur les notes fournies par les utilisateurs ayant des goûts similaires.

Chapitre III

Systèmes de filtrage collaboratif

1. Introduction :

Le but principal des systèmes de filtrage collaboratif est de générer pour chaque utilisateur des recommandations personnalisées susceptibles de l'intéresser en utilisant les préférences des autres utilisateurs. Jusqu'à présent, la majorité des méthodes de filtrage collaboratif fondent leur approche sur la prédiction de notes. Elles prennent toutes les notes disponibles en entrée et leur but étant de prédire les notes inconnues en sortie. Le système propose à l'utilisateur actif les articles dont les notes estimées sont les plus élevées. Les méthodes d'apprentissage ont naturellement été proposées pour considérer l'ensemble des notes et automatiser les traitements.

La diversité des applications du filtrage collaboratif a alors amené à la création de plusieurs types d'approches, répondant à des spécifications différentes. Les premières approches sont basées sur la notion des plus proches voisins [**Herlocker et al 99; Bell 07**].

Dans un premier temps, le système identifie les utilisateurs qui ressemblent à l'utilisateur actif. Leurs notes sont ensuite utilisées pour la prédiction et la recommandation. Ces méthodes sont simples et intuitives, cependant, elles sont beaucoup plus coûteuses pour brasser des millions de notes et faire de la recommandation en temps réel. Les approches qui génèrent un modèle permettent d'éviter ces problèmes. Elles supposent implicitement que les comportements utilisateurs (dont nous ne connaissons que la partie visible à travers les notes) peuvent être expliqués à travers les notes fournies. Chaque utilisateur de la base est alors vu comme une combinaison de ces comportements type. L'apprentissage est utilisé pour identifier ces derniers et faire des prédictions pour les articles non notés. Ces systèmes utilisent des adaptations de méthodes de réduction dimensionnelle linéaire à la matrice (utilisateur, article) [**Sarwar et al. 00; Srebro 03; Goldberg et al. 01**]. La difficulté majeure vient de la prise en compte des données manquantes. Les travaux de [**Srebro 03**] ont permis de développer un cadre élégant à la décomposition en valeurs singulières svd dans un tel contexte.

Des modèles probabilistes plus compliqués [**Marlin 03; Hofmann 04; Das et al 07; Salakhutdinov et al 07**] ont été proposés dans la littérature pour analyser et expliquer plus finement le comportement des utilisateurs.

Néanmoins, certains d'entre elles nécessitent une connaissance approfondie du domaine de l'apprentissage pour être mises en œuvre. Elles restent parmi les plus compétitives et la compréhension des comportements utilisateurs permet d'améliorer le développement du système, de l'interface utilisateur ou des ensembles d'articles similaires [**Hofmann 04; Polcicová 04**].

2. Les origines du filtrage collaboratif :

Le premier système de filtrage collaboratif a été proposé par David Golberg et ses collaborateurs chez Xerox en 1992 avec la mise en place du système de recommandation personnalisée Tapestry [**Goldberg et al 92**]. Deux ans plus tard, en 1994, Paul Resnick du MIT (Massachusetts Institute of Technology) et ses collaborateurs de l'université du Minnesota ont proposé l'architecture GroupLens pour recommander des articles dans les newsgroups [**Resnick et al 94**]. La librairie Amazon et plus particulièrement Greg Linden a popularisé le filtrage collaboratif avec sa fonction « les utilisateurs qui ont aimé ce livre ont aussi aimé tel autre livre ». En 1998, Brin et Page ont publié leur algorithme PageRank et lancé Google.
La même année chez Microsoft, John S. Breese et ses collaborateurs ont publié un article charnière, Empirical Analysis of Predictive Algorithms for Collaborative Filtering [**Breese 98**] dans lequel figure une comparaison détaillée des divers algorithmes de filtrage collaboratif. Dans les années 2000, les algorithmes de filtrage collaboratif étaient basés sur les réseaux bayésiens ou les réseaux de neurones avec une approche basée sur l'utilisateur.
En 2001, Amazon dépose un brevet introduisant le filtrage collaboratif basé sur l'article [**Linden 03**] ; ce type d'algorithme étant également publié la même année et de façon indépendante par la communauté GroupLens. En 2006, la compagnie Netflix annonce son challenge avec une récompense très attrayante, rendant ainsi disponible un ensemble de données réelles et volumineuses pour évaluer les systèmes de recommandation [**Thomas 11**].

3. Principe de fonctionnement du Filtrage collaboratif :

Le principe des systèmes de filtrage collaboratif repose sur les appréciations fournies par les utilisateurs sur des documents, sous forme de notes (valeurs entières). Dans un tel système, un utilisateur appartient à une communauté d'utilisateurs, les membres de cette dernière se partagent le même centre d'intérêt. Donc, pour recommander un document à un utilisateur donné, le système se base sur les évaluations des autres utilisateurs de la même communauté.

De plus, un utilisateur doit pouvoir bénéficier de ce que les autres ont déjà trouvé et évalué. Par exemple les utilisateurs qui veulent lire un livre ou regarder un film, demandent les avis de leurs amis. Donc, dans le filtrage collaboratif, la sélection des documents à proposer à un utilisateur ne dépend plus du contenu de document, mais des estimations faites par les membres de sa communauté.

La figure ci-dessous, présente un exemple des étapes de filtrage collaboratif, (où ***V*** et ***X*** représentent des évaluations données par l'utilisateur u_0 sur des documents), la valeur prédite pour qu'un document donné est proposé à l'utilisateur u_0, est calculée en rapprochant les évaluations passées de l'utilisateur des évaluations que d'autres utilisateurs (u_1, u_2, u_5 et u_7) de la communauté ont donné par le passé sur les mêmes documents **[Berrut 03]**.

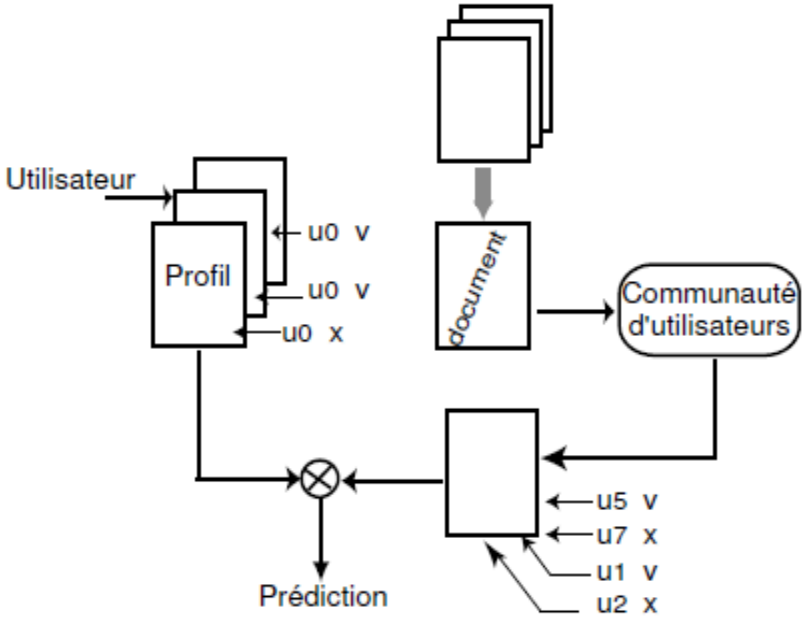


Figure III.1 *Principe du filtrage collaboratif.*

Parmi les systèmes commerciaux en ligne qui emploient la technique du filtrage collaboratif, nous trouvons :

- **Amazon** qui est notamment connu pour la recommandation des livres ;
- **Amie Street** qui permet aux utilisateurs de recommander des musiques. D'un autre côté, il leur permet de découvrir des musiques qui peuvent les intéresser en leurs fournissant des recommandations selon leurs préférences. Il permet également un système collectif de calcul de prix d'une musique ;
- **eBay** qui est un site d'e-commerce de ventes aux enchères permettant aux utilisateurs de vendre et d'acheter des biens. Il recommande aux utilisateurs des articles selon leur profil ;
- **Google News** qui est un site de journaux en ligne, rassemble des articles provenant de plus de 4,500 sources et regroupe les informations similaires pour les afficher en fonctions des intérêts de chaque utilisateur [**Das et al 07**] ;
- **Baynote** qui est une plateforme d'intelligence collective permettant aux entreprises de déployer des recommandations « orientées visiteurs », de faire de la recherche sociale

d'information et de fournir d'autres services à travers des canaux multiples tout en offrant un seul point d'entrée, d'essai et de contrôle. Baynote réclame faire 4 milliards de recommandations chaque mois ;

ChoiceStream qui est un service du genre SAS (Software As Service) fournit des recommandations sur des produits ainsi que des contenus informationnels aux consommateurs (utilisateurs). ChoiceStream utilise une approche probabiliste pour la personnalisation tout en combinant un ensemble de techniques statistiques telles que le filtrage collaboratif, les tableaux de corrélation multiple, l'analyse par cohorte, la corrélation des attributs, le filtrage sélectif et le multiple term scoring [**ChoiceStream 2009**].

4. Architecture générale :

Un système de filtrage collaboratif est composé de deux fonctionnalités fondamentales: le calcul de la proximité entre les utilisateurs et le calcul de la prédiction de l'évaluation qu'un utilisateur fera d'un document, auquel s'ajoute la fonctionnalité de mise à jour perpétuelle des profils d'utilisateurs, au fur et à mesure de la collecte de leurs évaluations **[Berrut 03]**.

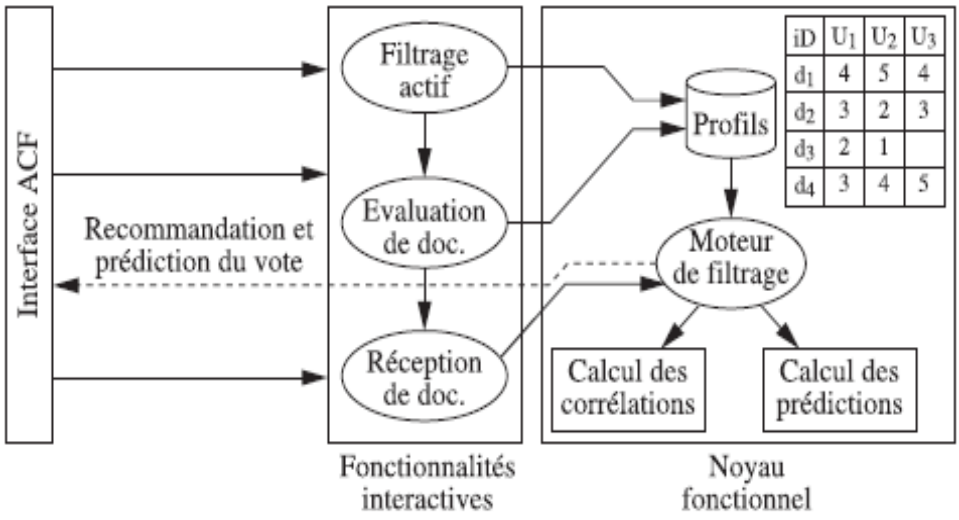


Figure III.2 *Architecture générale d'un système de filtrage collaboratif*

Dans ce type de système, où l'utilisateur contribue de façon décisive au bon fonctionnement du système dans son ensemble, on ne peut négliger les fonctionnalités interactives du système. Les fonctionnalités indispensables sont les suivantes :

- une interface permettant d'évaluer un document;
- une interface permettant de visualiser les documents reçus par filtrage.

D'autres fonctionnalités interactives peuvent exister, notamment celle permettant aux utilisateurs d'effectuer ce que l'on appelle du « filtrage actif » : le terme « filtrage actif » [**Maltz 95**] traduit le fait que l'utilisateur décide, de sa propre initiative, d'envoyer des documents à certains membres de la communauté. Cette possibilité peut s'avérer très utile lors de l'amorçage du système, pour faire croître les chances de recoupement des profils d'utilisateurs. En effet, un utilisateur qui reçoit un document envoyé par un autre est amené à l'évaluer lui aussi ; à la suite de cette évaluation, son propre profil et celui de l'auteur du filtrage actif se recouperont nécessairement.

5. Avantages du filtrage collaboratif :

L'approche collaborative :

- ✓ permet de traiter n'importe quelle forme de contenu et de diffuser des ressources pas nécessairement similaires à celles déjà reçues, où la sélection de documents ne s'appuie plus sur leurs contenus, mais sur une base de profils qui traduit les opinions que les utilisateurs ont émises sur les documents [**Berrut 03**].
- ✓ Permet aux utilisateurs de s'exprimer sur des concepts subjectifs tels que « goûts» et « qualité », en leur offrant la possibilité de s'entraider. Chacun des utilisateurs peut bénéficier des opinions que d'autres ont déjà émises sur un document.
- ✓ Offre une ouverture vers des recommandations inattendues, par la réception d'un document jugé intéressant par un collègue de la communauté donc, l'utilisateur peut enrichir son profil par un nouveau thème en retournant un avis de pertinence positif sur ce document ;

6. Inconvénients du filtrage collaboratif :

Le filtrage collaboratif souffre de certains problèmes, parmi lesquels :

- ✓ La masse critique qui est un problème lié à la taille de la population d'utilisateurs et de l'ensemble de documents à évaluer.
- ✓ Le démarrage à froid est un grand problème pour le filtrage collaboratif, le système doit avoir un minimum d'évaluations des utilisateurs, pour que le calcul de prédiction prenne tout son importance pour filtrer efficacement des documents pour un utilisateur, or les profils au départ sont vides et une période d'apprentissage plus au moins longue est toutefois nécessaire pour les constituer, période durant laquelle beaucoup d'utilisateurs peuvent abandonner le système.
- ✓ Un problème du système collaboratif est que sa performance dépend beaucoup de la distribution des évaluations (notes) données par utilisateurs. Si dans le système il y a

plusieurs articles qui ont été utilisés et évalués par très peu d'utilisateurs, ces articles seraient recommandés très rarement, même si ces utilisateurs ont donnée des notes très hautes pour ces articles. Ce problème est connu comme étant le problème de parcimonie (sparsity problem).

- ✓ Le problème des nouveaux documents qui ne peuvent être diffusés aux utilisateurs concernés du fait qu'ils n'ont pas encore été évalués.

7. Techniques de filtrage collaboratif :

Selon la classification de Breese et al. [**Breese 98**], le filtrage collaboratif se base sur deux grands types d'algorithmes pour calculer la prédiction : les algorithmes basés mémoires et les algorithmes basés modèles.

7.1 Algorithmes basés mémoires *:*

Les algorithmes basés mémoire consistent à utiliser la totalité des informations disponibles pour générer les prédictions. Généralement on distingue deux approches. Celles centrées utilisateurs et celles centrées articles. L'approche qui se réfère aux utilisateurs [**Resnick et al 94**] consiste à comparer les utilisateurs entre eux et de retrouver ceux ayant des goûts similaires ainsi les notes d'un utilisateur étant ensuite prédites selon son voisinage. Par contre l'approche qui se réfère aux articles [**Sarwar et al 01**] consiste à rapprocher les articles appréciés par les mêmes personnes et à prédire les notes des utilisateurs en fonction des articles les plus proches de ceux qu'ils ont déjà notés.

Afin de calculer une nouvelle prédiction, les algorithmes basés mémoire [**Resnick 94**] ; [**Nakamura 98**], utilisent la totalité de la base de données des évaluations fournies par les utilisateurs. Les évaluations de l'utilisateur actif sont prédites à partir d'informations partielles concernant l'utilisateur actif, et un ensemble de poids calculés à partir de la base de données des évaluations des autres utilisateurs. La valeur prédite de l'évaluation d'un document *j* pour un utilisateur *a* sera calculée par la formule :

$$P_{a,j} = \overline{v}_a + k \sum_{i=1}^{n} w(a,i)(v - \overline{v}_i)$$

Où *n* représente le nombre d'utilisateurs figurants dans la base de données qui ont un poids non nul, et *k* est un facteur de normalisation. Le poids *w (a, i)* détermine la distance (la proximité) entre l'utilisateur actif *a* et les autres utilisateurs $i \in [1..n]$, cette distance est calculée de façon variable, selon l'algorithme.

Le facteur de normalisation ***k*** est calculé de la manière suivante :

$$k = \frac{1}{\sum_{i=1}^{n} |w(a,i)|}$$

La distance de similarité entre deux utilisateurs peut être calculée selon différentes méthodes, une des plus utilisées est celle du coefficient de Pearson [**Resnick et al 94**]; [**Shardanand 95**] ou la méthode basé sur la similarité de vecteurs [**Breese 98**] ; [**Sarwar et al 01**].

Pour la méthode basée sur le coefficient de Pearson, le poids est calculé comme la corrélation entre les utilisateurs *a* et *i*, comme suit :

$$w(a,i) = \frac{\sum_{j} (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_{j} (v_{a,j} - \bar{v}_a)^2 (v_{i,j} - \bar{v}_i)^2}}$$

Avec : ***w (a, i)*** : La distance entre l'utilisateur actif ***a*** et l'utilisateur ***i*** ;

$v_{a,j}$: L'évaluation du document ***j*** par l'utilisateur actif ***a*** ;

v_i : L'évaluation moyenne de l'utilisateur ***i*** ;

$v_{i,j}$: L'évaluation du document ***j*** par l'utilisateur ***i*** ;

Les sommes sur les ***j*** concernent les documents pour lesquels ***a*** et ***i*** ont donné à la fois des évaluations.

Pour les méthodes basées sur la similarité des vecteurs, le poids est calculé comme un cosinus entre les vecteurs formés par les évaluations des utilisateurs, par la formule suivante :

$$w(a,i) = \sum_{j} \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}}$$

Et si I_i est l'ensemble des articles évalués par l'utilisateur ***i***, alors l'évaluation moyenne pour l'utilisateur i peut être définie comme suit :

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

Où $v_{i,j}$ représente l'évaluation du document j par l'utilisateur i ;

L'approche basée sur la mémoire a pour avantage la simplicité de l'implémentation et de l'intégration des nouvelles données dans le système. Cependant, cette approche a l'inconvénient d'être très dépendante de la quantité de notes des utilisateurs. En effet, si les données s'avèrent rares, il est difficile d'identifier des voisins fiables (à partir des articles co-notés) et par conséquent la performance du système décroît.

De plus, dans une situation de démarrage à froid, cette approche est incapable de tenir compte des nouveaux utilisateurs et/ou articles, récemment introduits au système. En effet, l'approche basée sur la mémoire nécessite la disponibilité des appréciations concernant ces utilisateurs et/ou ces articles pour pouvoir les intégrer parmi les recommandations.

En outre, l'approche basée sur la mémoire reste limitée dans la mesure où elle ne permet pas le passage à l'échelle. En effet, quand le nombre d'utilisateurs et d'articles présents dans le système devient important, la génération des recommandations requiert un temps de traitement très élevé [**Esslimani 10**].

7.2 Algorithmes basés modèles :

Les algorithmes basés modèle ont été intégrées aux systèmes de recommandation pour remédier aux problèmes des méthodes basées sur la mémoire, dont notamment : la non robustesse au manque de données ainsi que le non passage à l'échelle [**Sarwar et al 00**] [**Su et al 09**]. Pour faire face à ces deux problèmes, les méthodes basées sur un modèle utilisent notamment les techniques de réduction de dimensionnalité comme Factorisation Matricielle, le clustering dans le but d'écarter les utilisateurs ou les articles non représentatifs. Ainsi l'espace de représentation utilisateur-item est plus réduit et le taux de données manquantes est moins important comparé à l'espace de représentation original. Les voisins peuvent ainsi être calculés dans cet espace réduit, ce qui permet de garantir le passage à l'échelle.

Les algorithmes basés modèle utilisent la base de données des évaluations des utilisateurs pour créer ou apprendre un modèle de prédiction via un processus d'apprentissage, telles que : le clustering, les réseaux bayésiens, les arbres de décision …etc.

Les modèles construits sont par la suite utilisés pour générer les prédictions qui sont proposées à l'utilisateur actif lors de son interaction avec le système.

7.2.1 Clustering :

Le principe du modèle à base de clusters est de regrouper (en clusters) les utilisateurs ayant les mêmes goûts et les ressources portant sur les mêmes sujets, ou qui ont tendance à plaire aux mêmes personnes.

Dans le cadre du filtrage collaboratif (FC), le clustering a pour objectif de créer des clusters homogènes d'utilisateurs ou d'articles. Les prédictions sont par la suite calculées en prenant en considération les opinions des utilisateurs (en FC centré sur l'utilisateur) ou les notes des articles (en FC centré sur l'article) faisant partie des mêmes clusters.

Les méthodes de clustering les plus exploitées sont les méthodes de partitionnement dont k-means [**MacQueen 67**] est la plus populaire. Dans le cas d'un clustering d'utilisateurs [**Cho et al 02**], k-means consiste à créer *k* clusters telle que la distance entre utilisateurs intracluster est faible alors que la distance inter-cluster est forte. En d'autres termes, chaque cluster créé doit comprendre des utilisateurs ayant des appréciations similaires.

7.2.2 Modèles probabilistes :

De nombreux modèles probabilistes ont été proposés pour le filtrage collaboratif, ces modèles visent à représenter le calcul des prédictions sous forme de distributions de probabilité [**Schafer et al 07**].

Supposons que les évaluations se fassent sur une échelle d'entiers de *0* à *m*.
Alors la valeur prédite sera :

$$p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} \Pr(v_{a,j} = i \,|\, v_{a,k,}\, k \in I_a) i$$

Avec : I_a : Ensemble des documents appréciés par l'utilisateur actif ***a*** ;

0,.., m : Ensemble des valeurs possibles des appréciations ;

La probabilité exprimée dans la formule, s'interprète comme étant la probabilité que l'utilisateur *a* attribué à l'évaluation *i* au document *j*, sachant les évaluations qu'il a déjà attribuées sur les autres documents.

Les premières méthodes probabilistes proposées pour le FC sont basé sur les réseaux bayésiens qui exploitent les arbres de décision. Le FC est ainsi perçu comme un réseau bayésien où chaque article représente un nœud. Les états de chaque nœud correspondent aux valeurs possibles de note. Ces valeurs comprennent aussi l'état "pas de note" correspondant à une note manquante. Ainsi, pour prédire ces notes manquantes, un algorithme d'apprentissage de réseaux bayésiens est appliqué. Dans les réseaux résultant de cet apprentissage, chaque article dispose d'un article parent à travers un arbre de décision qui définit les probabilités conditionnelles qu'un article soit apprécié ou non par l'utilisateur.

Il existe plusieurs modèles probabilistes qui modélisent les comportements des utilisateurs à l'aide d'une variable latente pouvant prendre *K* valeurs, où chaque valeur correspond à un comportement type.

Dans la suite, *Z* est une variable latente dans *{1,...,K}*, *U* et *u* sont une variable et un indice d'utilisateurs, *Y* et *y* sont une variable et un indice d'articles, *R* et *r* sont une variable note et une note dans ***{1,...,V}***. Avec ces notations, chaque utilisateur *u* est représenté par une distribution *p(Z|U = u)*, modélisant à quel point chaque comportement type intervient dans le comportement de *u*. A chaque comportement type *z* et à chaque article ***y*** est associée une distribution multinomiale *p(R|Z = z, Y =y)* qui modélise les goûts du comportement type *z* pour l'article *y*. D'un point de vue génératif, la note attribuée par l'utilisateur *u* à l'article ***y*** est générée de la façon suivante : un comportement type ***z*** est choisi suivant la distribution *p(Z|U = u)*, puis la note ***r*** est choisie suivant la distribution *p(R|Z = z, Y = y)*. Lorsque les distributions sont connues, la prédiction de notes pour l'utilisateur u et l'article ***y*** consiste à calculer les *p(R = r|U = u, Y = y)* pour toutes les notes ***r***, et à prédire la note médiane $\hat{r}$ telle que $\hat{r}$ *= {r|p(R = r|U = u, Y = y)≤ 1/2, p(R = r|U =u, Y = y) ≥1/2}*.

[**Hofmann 04**] a d'abord proposé le modèle d'aspects, dans lequel il considère les distributions *p(Z|U = u)* et *p(R|U = u, Y = y)* comme les paramètres du modèle à apprendre. L'apprentissage se fait à l'aide d'un algorithme EM sur le principe du maximum de vraisemblance. Mais le fait de considérer ces distributions comme des paramètres pose problème : les *p(Z|U = u)* ne peuvent être déterminés que pour les utilisateurs qui étaient présents au moment de l'apprentissage. Le modèle d'aspects ne peut donc pas faire de prédictions pour de nouveaux utilisateurs, ce qui rend impossible son utilisation dans un cadre en ligne. Une autre conséquence indésirable est que le nombre de paramètres augmente avec le nombre d'utilisateurs, augmentant ainsi le temps de l'apprentissage. Une réponse possible à ces problèmes est de considérer les distributions *p(Z|U = u)* non plus comme des paramètres, mais comme des variables aléatoires suivant des distributions paramétrées. Dans le cas particulier de distributions de Dirichlet, on se retrouve avec le modèle URP proposé par [**Marlin 04b**]. Mentionnons également le modèle d'attitudes de [**Marlin 04a**], qui est une amélioration du modèle URP permettant aux différents comportements type d'interagir lors de la génération des notes pour chaque article. En termes de prédiction de notes, le modèle URP et le modèle d'attitudes sont les approches les plus performantes à ce jour parmi les modèles probabilistes [**Marlin 04a**].

7.2.3 Factorisation Matricielle :

Les approches à base de factorisation matricielle pour le filtrage collaboratif reposent sur la décomposition de matrices comportant des "trous" (matrice creuse), c'est à dire que les valeurs de certaines entrées sont inconnues. Ces trous correspondent aux notes manquantes

que l'on veut prédire. Les méthodes de factorisation classiques n'étant capables de factoriser que des matrices pleines, la difficulté principale consiste à les adapter au cas de données manquantes. Une fois calculée, la factorisation permet de reconstruire une matrice pleine, contenant les entrées initiales ainsi que les prédictions pour les entrées manquantes [**Jean et al 07**].

L'objectif de ces approches est de représenter les données dans un espace de dimension inférieure à celles de départ et d'offrir un ensemble de méthodes parmi lesquelles : la décomposition en valeurs singulières (*DVS*), décomposition en valeurs singulières pondérées (*DVSP*), factorisation matricielle non-négatives (*FMN*) et factorisation matricielle maximisant la marge (*FMMM*).

7.2.4 Décomposition en valeurs singulières (DVS) [Jean 08] :

La décomposition en valeurs singulières *(DVS)* est une technique fondamentale en algèbre linéaire. Elle permet de factoriser une matrice *Y* de taille *(n X p)* en un produit $U\sum V^T$, où les matrices *U* et *V* sont des matrices orthogonales contenant les vecteurs singuliers respectivement gauches et droits de *Y*, $\sum$ est une matrice diagonale de taille *(p X p)* contenant les valeurs singulières de *Y*.

La solution standard pour une décomposition en *DVS* est de présenter $\sum = diag(\sigma_1,.....,\sigma_d)$ avec σ_i la i^{ème} plus grande valeur propre de YY^T, les colonnes de *U* sont définies comme des vecteurs propres de YY^T et celles de *V* comme des vecteurs propres de Y^TY.

La *DVS* permet de déterminer la matrice de rang $k<P$ optimale pour la norme de Frobenius, c'est–à–dire la matrice $\tilde{Y}$ de rang *k* qui minimise $\left\|Y-Y\right\|_{Fro}^2$ avec :

$$\left\|Y-Y\right\|_{Fro}^2=\sum_{ij}\left\|Y_{ij}-\tilde{Y}_{ij}\right\|^2$$

En effet, la solution s'exprime en fonction des matrices trouvées par la décomposition de *Y*. Elle est donnée par la formule suivante :

$$\tilde{Y}=U_k\sum_k V_k$$

Avec $\sum_k$, la sous matrice *k X k* en ne gardant que les *k* premiers éléments de la diagonale et en tronquant le reste. U_k et V_k sont obtenues en prenant les *k* premières colonnes de *U* et *V* et en tronquant le reste.

7.2.5 Décomposition en valeurs singulières pondérées (DVSP) [Nguyen 09]:

La méthode *DVS* ne peut pas être appliquée directement en filtrage collaboratif. Soit *Y* la matrice utilisateur-article qui contient les notes fournies par les utilisateurs et celles manquantes. Srebro et Jaakola [**Srebro et al 03**] ont proposé d'étendre cette méthode pour le filtrage collaboratif en cherchant une matrice qui minimise la norme de Frobenius pondérée:

$$L(Y,\tilde{Y}) = \sum_{ij} \left\| W_{ij}(Y_{ij} - \tilde{Y}_{ij}) \right\|$$

Où W_{ij} est l'indicateur valant **0** si Y_{ij} est une donnée manquante, sinon 1.

Ces auteurs proposent aussi une méthode de type espérance-maximisation (EM) permettant de minimiser la norme de Frobenius pondérée. En premier, les notes manquantes sont initialisées aléatoirement. L'algorithme alterne ensuite deux étapes jusqu'à convergence. La première consiste à trouver la matrice de rang *k* qui se rapproche le plus de la matrice de notes obtenues.

La deuxième consiste à estimer de nouveau les notes qui n'ont pas été renseignées. L'algorithme suivant détaille plus formellement ces instructions. Avec *W* la matrice d'éléments W_{ij}, 1 la matrice *n X p* contenant que des 1 et $A \otimes B$ le produit de Schur (produit terme à terme) des matrices *A* et *B*. SVD_K donne l'approximation au rang *k* par la méthode *DVS*.

Algorithme de la DVS pondérée pour le filtrage collaboratif

Entrées : La matrice utilisateur-article ***Y***

La matrice ***W*** indicatrice de notes manquantes.

1 : Initialiser aléatoirement $\tilde{Y}$

2 : **répéter**

3 $X \leftarrow W \otimes Y + (1-W) \otimes \tilde{Y}$

4 $[U_k, \sum_k, V_k] = SVD_k(X)$

5 $\tilde{Y} \leftarrow U_k \Sigma_k V_k^T$

6 : **jusqu'à** convergence de la norme de Frobenius de $W \otimes Y - W \otimes U_k \Sigma_k V_k$

La *DVS* est mise en œuvre dans de nombreuses bibliothèques de calcul, ce qui facilite grandement son utilisation ainsi que celle de la DVS pondérée.

Les auteurs remarquent que le nombre d'itérations nécessaires pour la convergence de l'algorithme et la qualité de la solution trouvée dépendent fortement de la proportion d'entrées manquantes.

B. Marlin [**Marlin 04a**] a proposé une manière simple pour faire des recommandations à des utilisateurs absents lors de l'apprentissage. L'arrivée d'un nouvel utilisateur correspond à un vecteur de notes ***y*** et à un vecteur *W* dont l'élément W_i indique si y_i est une note présente ou non. Le vecteur *y* correspond à une nouvelle ligne de la matrice *Y*. Pour prédire les notes manquantes, il suffit de trouver un modèle utilisateur c'est–à–dire une nouvelle ligne u' pour ***U***. Pour y parvenir, il suffit de minimiser l'erreur suivante :

$$\left\| W \otimes Y - W \otimes u' \Sigma_k V_k^T \right\|$$

Cela revient à résoudre un problème de régression, puisque cette fois-ci la représentation vectorielle des articles est donnée et qu'il faut apprendre un modèle pour reconstituer les notes présentes. Dans [**Marlin 04a**], une méthode itérative a été proposée et décrite par l'algorithme ci-dessous :

Algorithme de la DVS pondérée pour la prise en compte des nouveaux utilisateurs proposé par B.Marlin.

Entrées : Le vecteur de notes ***Y***

Le vecteur ***W*** indiquant les notes manquantes.

Les matrices ∑ et V apprises par SVD

1 : Initialiser aléatoirement $\tilde{\tilde{Y}}$

2 : **répéter**

3 $X \leftarrow W \otimes Y + (1 - W) \otimes \tilde{\tilde{Y}}$

4 $u' \leftarrow x V_k \Sigma_k^{-1}$

5 $\tilde{\tilde{Y}} \leftarrow u' V_k \Sigma_k^T$

6 : **jusqu'à** convergence de la norme de Frobenius de $W \otimes Y - W \otimes u' \Sigma_k V_k^T$

Elle alterne une étape d'estimation des notes manquantes (instructions 3 et 5) et une étape d'estimation du vecteur utilisateur (instruction 4). L'algorithme nécessite l'inversion de la matrice diagonale $\sum_k$ et des produits matriciels.

7.2.6 Factorisation matricielle non-négatives pour le filtrage collaboratif :

La factorisation en matrices non négatives (*FMN*) est une méthode générale de décomposition matricielle, introduite par [**Lee et Seung 99**]. Elle permet d'approximer toute matrice Y de taille $(n \times P)$ et dont les éléments sont tous positifs, grâce à une décomposition de la forme $Y \approx BC$, où B et C sont des matrices $(n \times k)$ et $(k \times P)$. La matrice Y contient les vecteurs réels de dimension P, la matrice B contient les vecteurs correspondants dans un espace de dimension $k < P$, et la matrice de passage C contient les vecteurs de base.

L'originalité de la FMN réside dans les contraintes de non-négativité qu'elle impose à B et C ; c'est à dire que leurs éléments doivent être tous positifs. Ces contraintes font que les vecteurs de base comportent beaucoup de 0 et que leurs parties non nulles se chevauchent rarement. La représentation d'un objet (décrit par un vecteur de réels positifs) comme une somme de ces vecteurs de base, correspond alors à l'intuition d'une décomposition par parties.

La FMN a été appliquée avec succès, notamment en reconnaissance des visages [**Lee et Seung 99**] et en classification de documents textuels (**[Xu 03]**, **[Berry 04]**).

La FMN a été utilisée dans un grand nombre de domaines. Citons par exemple l'analyse de documents textuels [**Shahnaz et al 06**; **Pauca et al 04**], le traitement d'image ou la biologie [**Devarajan 08**].

Déterminer les matrices B et C revient à minimiser la distance entre la matrice initiale et le produit BC; plus précisément, il faut minimiser la norme de Frobenius $\|Y - BC\|^2 = \sum_{ij} (Y_{ij} - (BC)_{ij})^2$ sous les contraintes de non-négativité.

C'est un problème d'optimisation non trivial, que [**Lee et Seung 00**] proposent de résoudre en initialisant B et C aléatoirement, puis en alternant les mises à jour suivantes :

$$C_{ij} \leftarrow C_{ij} \otimes \frac{(B^T Y)_{ij}}{(B^T BC)_{ij}} \qquad B_{ij} \leftarrow B_{ij} \otimes \frac{(Y^T C)_{ij}}{(BCC^T)_{ij}}$$

Les auteurs montrent que leur algorithme converge vers un minimum local. Nous remarquons qu'à chaque étape, la complexité est en $O(npKt)$, où t est le nombre maximal d'itérations.

[Jean et al 06] appliquent la FMN au filtrage collaboratif, motivés par l'idée de pouvoir représenter les utilisateurs comme des sommes de comportements types. Ces derniers étant alors directement interprétables et correspondent à des utilisateurs imaginaires ayant noté des sous-ensembles d'articles.

Ces mêmes auteurs ont proposé une méthode simple inspirée de [**Srebro et al 03**], qui est décrite par l'algorithme ci-après. Elle se fonde sur une méthode de factorisation matricielle

non négative FMN_K. C'est en fait un cas particulier de EM (espérance-maximisation) [**Dempster et al 77**; **Collins 97**] dans le cadre des données incomplètes. Elle alterne une étape d'estimation des données manquantes par la matrice approchée, puis calcule à nouveau la factorisation matricielle non négative. Il est montré que l'algorithme non seulement converge mais la limite est au pire un point stationnaire.

Pour éviter ces points (et obtenir un minimum), les auteurs lancent plusieurs fois l'algorithme avec différentes initialisations. Comme préconisé dans [**Srebro et al 03**], ils commencèrent par chercher une approximation matricielle à un rang plus élevé en le diminuant au fur à mesure des itérations jusqu'à atteindre la valeur de ***k***.

Algorithme générique pour le filtrage collaboratif basé sur FMN.

1 : $t \leftarrow \mathbf{0}$

2 : Initialiser aléatoirement ***B*** et ***C***

3 : **répéter**

4 $\quad Y \leftarrow W \otimes Y - (1-W)BC$

5 $\quad BC \leftarrow FMN_k(Y)$

6 : **jusqu'à** convergence

7.2.7 Factorisation en matrices non-négatives généralisée (FMNG) :

La factorisation en matrices non-négatives généralisée (FMNG) de [**Dhillon et Sra 06**] permet d'approximer une matrice non-négative *Y* par un produit de matrices non négatives *BC* : $Y \approx BC$. La qualité de l'approximation *BC* est mesurée par des divergences de Bregman. Considérons une fonction convexe et dérivable φ: $\varphi : R \to R$. La divergence de Bregman D_φ associée à φ est définie comme $D_\varphi = \varphi(x) - \varphi(y) - \nabla(y)(x-y)$.

Pour calculer la divergence entre deux matrices, il est écrit simplement $D_\varphi(x,y) = \sum_{ij} D_\varphi(x_{ij}, y_{ij})$. Les divergences de Bregman sont des fonctions asymétriques, et la FMNG permet de résoudre les problèmes d'optimisation suivants **:**

$$\min_{B\geq 0, C\geq 0} D_\varphi(BC, Y) + g(B) + r(C)$$

$$\min_{B\geq 0, C\geq 0} D_\varphi(Y, BC) + g(B) + r(C)$$

Où *g* et *r* sont deux fonctions de pénalisation des matrices *B* et *C*, permettant de régulariser la factorisation. En général, les divergences de Bregman ne sont pas convexes en *B* et *C* simultanément, ce qui les rend difficiles à minimiser. Les auteurs [**Dhillon et Sra 06**] proposent une stratégie de minimisation de $D_\varphi(BC; Y)$ très similaire à l'algorithme EM (espérance-maximisation). En effet les auteurs ne minimisent pas directement $D_\varphi(BC; Y)$, mais plutôt une borne supérieure de la divergence. Dans le cas général, l'algorithme prend la forme d'une minimisation alternée après une initialisation aléatoire de *B* et *C* qui sont mises à jour alternativement jusqu'à ce que la fonction de coût n'évolue plus.

Dans le cas de la norme de Frobenius $D_\varphi(Y, BC) = \|W \otimes (Y - BC)\|^2$, les auteurs proposent l'algorithme ci-après pour déterminer les matrices *B* et *C*.

Algorithme : Factorisation en matrices non-négatives généralisée.

Entrée : W, Y

1 : **Initialisation :** $B \geq 0, C \geq 0$ aléatoires

3 : **répéter**

4 : $B \leftarrow B \otimes \dfrac{(W \otimes W \otimes Y)C^T}{(W \otimes W \otimes (BC))C^T}$

5 : $C \leftarrow C \otimes \dfrac{B^T(W \otimes W \otimes Y)}{B^T(W \otimes W \otimes (BC))}$

6 : **jusqu'à** convergence de $\|W \otimes (Y - BC)\|^2$

7 : **Sortie :** *B, C.*

7.2.8 Factorisation Matricielle Maximisant la Marge (FMMM) :

[**Rennie et al 05**] proposent d'apprendre une matrice de scores *S* de taille *(n × p)* et de rang *k*, ainsi qu'une matrice de seuils θ de taille *(n × (k − 1))*. La note prédite pour l'utilisateur *i* et l'article *j* ne dépend que de la position du score S_{ij} par rapport aux seuils $\{\theta_{ir}\}_{r=1\ldots k-1}$. La FMMM se place dans une optique légèrement différente de celles des méthodes de factorisation précédentes, puisqu'elle ne tente pas d'approximer directement la matrice de notes *Y*. Notons enfin que cette méthode est la plus performante de la littérature à ce jour en terme de prédiction de notes sur des bases standards de filtrage collaboratif [**DeCoste 06**].

8. Métriques d'évaluation :

Afin d'évaluer la qualité de prédictions d'un algorithme de filtrage collaboratif, il existe plusieurs mesures d'évaluations, parmi lesquelles :

8.1 Root Mean Squared Error (RMSE):

La mesure RMSE permet de déterminer l'erreur réalisée entre la note prédite par le système et la note réelle donnée par l'utilisateur [**Jannach et al 11**]. La mesure RMSE est une mesure d'erreur, plus sa valeur est faible, moins l'erreur est importante.

$$RMSE = \sqrt{\frac{\sum_{j=1}^{N} (V_{uj} - P_{uj})^2}{N}}$$

Où V_{uj} : Représente la note réelle de l'utilisateur u sur l'article j,

p_{uj} : La note prédite par le système et n le nombre total de mesures prédites.

N : L'ensemble des évaluations réelles attribuées par l'utilisateur *u*.

8.2 Mean Absolute Errror (MAE):

MAE, l'abréviation Mean Absolute Error **[Herlocker et al 04]** est également une mesure d'erreur permettant de calculer la moyenne des erreurs absolues entre les prédictions et les valeurs réelles fournies par les utilisateurs. En d'autres termes, elle permet de déterminer la qualité de ces prédictions. Rappelons que les prédictions sont générées à partir d'un ensemble d'apprentissage.

La mesure MAE se calcule de la manière suivante :

$$MAE = \frac{\sum_{j=1}^{N} |V_{uj} - P_{uj}|}{N}$$

Où $\boldsymbol{V_{uj}}$: Représente la mesure réelle de l'utilisateur u sur l'article j,

$\boldsymbol{p_{uj}}$: La mesure prédite par le système,

$\boldsymbol{N}$: Le nombre total de mesures prédites.

8.3 Normalized Mean Absolute Error (NMAE):

NMAE (Normalized Mean Absolute Error) permet de normaliser MAE et comparer différents algorithmes dans le cas où les échelles de votes ne sont pas les mêmes (par exemple comparaison d'un algorithme avec des votes 2 [1; 5] avec un autre algorithme dont les notes s'échelonnent sur l'intervalle [1; 7]).

NMAE est calculée de la façon suivante :

$$NMAE = \frac{MAE}{V_{\max} - V_{\min}}$$

8.4 High Mean Absolute Errror (HMAE):

La mesure HMAE est définie comme étant MAE obtenue uniquement sur les votes élevés. On considérera par exemple que sur une échelle de votes de 1 à 5, les votes élevés sont les valeurs 4 et 5. Cette métrique évalue la qualité des prédictions véritablement importantes, c'est-à-dire celles qui figureront potentiellement dans le top-N. En effet, il est relativement peu important que la fonction $f_{predict}$ estime mal les articles de votes faibles, puisque ceux-ci ne seront jamais suggérés par le système et s'avèrent inintéressants pour l'utilisateur courant. En revanche, l'erreur sur les votes élevés doit être la plus faible possible afin que le système soit capable de suggérer les bons articles dans le bon ordre de préférence **[Thomas 11]**.

9. Conclusion :

Le filtrage collaboratif (FC) est une technique utilisée par certains systèmes de recommandation. Il permet de faire des prédictions automatiques (filtrage) sur les intérêts d'un utilisateur par la collecte des préférences de nombreux utilisateurs (collaborateurs). Ces systèmes utilisent les notations des utilisateurs sur les articles dans le but de prédire leurs préférences sur les articles non notés.

Nous avons présenté dans ce chapitre un état de l'art sur les systèmes de filtrage Collaboratif, ses origines, son principe de fonctionnement, son architecture générale, ses avantages et inconvénients, ses algorithmes de calcul de prédiction de notes, à savoir ceux basés modèle et basés mémoire ainsi que leurs métriques d'évaluation.

Chapitre IV : Les méthodes d'apprentissages d'ordonnancement pour la recherche et le filtrage d'information

1. Introduction :

La majorité des recherches en apprentissage se sont intéressées aux problèmes issus de la classification et de la régression [**Nguyen 09**]. L'objectif est d'apprendre une fonction de prédiction, qui va induire, pour chaque entrée, une valeur désirée correspondante en sortie.

À partir de la fin des années 2000, un nouveau cadre suscite un intérêt croissant dans la communauté: l'apprentissage de fonctions d'ordonnancement. Ces fonctions considèrent plusieurs entrées, les comparent, et les renvoient sous la forme d'une liste ordonnée. L'ordre prédit doit être en accord avec une notion de préférence, spécifique au problème traité.

L'ordonnancement est le problème principal pour beaucoup d'applications de recherche d'information telles que la recherche Web, le filtrage collaboratif et la récupération de documents.

En recherche d'information, l'exemple le plus courant est celui des moteurs de recherche comme www.google.com qui présentent à l'utilisateur une liste de documents triés par ordre de pertinence, et non pas un ensemble de documents tous jugés pertinents et présentés sans ordre particulier.

En filtrage d'information, citons l'exemple des systèmes de recommandation, dont le but est de recommander des articles susceptibles d'intéresser un utilisateur. Du point de vue de la recommandation, trier les articles par ordre de pertinence semble plus approprié que prédire une note ou une classe pour chaque article [**Jean 08**].

L'apprentissage d'une fonction d'ordonnancement peut être considéré comme l'apprentissage d'une fonction de score : une fonction à valeurs réelles, qui prend en entrée un élément d'un ensemble à ordonner. L'ordre est ensuite prédit en triant les éléments selon les scores croissants ou décroissants. Contrairement aux cas de la classification ou de la régression, le score prédit pour une entrée donnée peut être arbitraire : seuls les scores relatifs entre les éléments sont importants. Pour réaliser l'apprentissage de fonctions de score, il est donc nécessaire de définir de nouveaux critères d'erreurs, des algorithmes pour optimiser l'erreur, et une théorie permettant de montrer que la fonction apprise sera performante sur les données qu'elle observera dans le futur [**Nicolas 06**].

Dans ce chapitre, nous présentons l'apprentissage automatique, l'apprentissage supervisé, l'ordonnancement dans la recherche d'information où nous avons décris l'ordonnancement d'instances et leur relation avec la classification binaire ainsi que le cas particulier de cet ordonnancement (biparti) et le cadre formel de l'apprentissage supervisé de fonctions d'ordonnancement. Nous survolons, ensuite, les principales approches telles que :

l'approche pointwise, pairwise et listwise. Nous évoquons enfin les méthodes d'ordonnancement pour le filtrage collaboratif.

2. L'apprentissage automatique (machine learning) :

Appelé aussi apprentissage statistique, est un type d'intelligence artificielle où un algorithme modifie son comportement automatiquement de manière à améliorer sa performance sur une tâche à partir d'un ensemble de données. Nous qualifions ce processus d'apprentissage du fait que l'algorithme est optimisé à partir d'un ensemble d'observations et qui essaye d'en extraire les régularités statistiques [**François 09**]. Il existe plusieurs types d'apprentissage : supervisé, semi-supervisé et non supervisé. Le type d'apprentissage objet principal de ce mémoire est l'apprentissage supervisé.

3. Présentation de l'apprentissage supervisé :

En apprentissage supervisé, on veut apprendre la relation probabiliste (ou la distribution jointe) $p(x,y)$ entre les exemples (ou formes d'entrée) $x \in X$ et les sorties désirées $y \in Y$. Cet apprentissage se réalise habituellement à l'aide d'exemples étiquetés $\{(x_i,y_i)/\ i=1,\ldots,n\}$ issus de la probabilité jointe $p(x,y)$. L'apprentissage supervisé inclus notamment la **classification** (avec Y un ensemble discret) et la **régression** ($Y \subset R^N$).

Un exemple est un couple composé d'une observation et d'une sortie désirée. Les observations possèdent une représentation numérique dans un espace vectoriel X, typiquement $X \subset R_d$ pour d fixé. La réponse sera appelée *sortie désirée*, et elle est supposée faire partie d'un ensemble de sortie $Y \subset R$. Lorsqu'on est devant un problème de classification (Y discret), la fonction de prédiction f est appelé un classifieur. Lorsque Y est continu, la fonction f est une fonction de régression. Un couple (x, y) désignera un élément de $X \times Y$, et $S = \left((x_1, y_1),\ldots(x_m, y_m) \in \left(X \times Y\right)^m\right)$ dénotera un ensemble d'exemples, un ensemble d'apprentissage ou un ensemble de test selon les cas. Enfin, un algorithme d'apprentissage est une fonction, qui, pour chaque ensemble d'apprentissage S, choisit une fonction de prédiction dans un ensemble prédéfini F. Lorsque le contexte est clair, la sortie de l'algorithme d'apprentissage considéré sera notée f_S [**Jean 08]**.

3.1 Fonctions d'erreur (*risque* ou *coût) :*

Etant donnée une fonction de prédiction f, l'accord entre la prédiction $f(x)$ et la sortie désirée y pour un couple (x, y) est mesuré grâce à une fonction de risque instantané (ou d'erreur instantanée ou de coût instantané).

Une fonction $L : Y \times Y \rightarrow R_+$. Intuitivement, $L(f(x), y)$ mesure la proximité entre la réponse prédite et la réponse réelle. C'est donc généralement une distance sur l'ensemble Y.

En classification, l'erreur instantanée généralement considérée est :

$$L_r(f(x), y) = [[f(x) \neq y]]$$

Ou **[[P]]** vaut 1 si le prédicat **P** est vrai et sinon 0.

En régression, la fonction d'erreur instantanée couramment utilisée est la distance sur R définie par le carré de la valeur absolue :

$$L_r(f(x), y) = |f(x) - y|^2$$

3.2 Généralisation et les ensembles de données :

Le but principal de l'apprentissage automatique est la **généralisation**. En effet, il est facile de construire un modèle qui apprendra par cœur l'ensemble d'entraînement, c'est-à-dire de trouver une fonction f telle que $L(f|S) \approx 0$. Cet ensemble ne représente qu'un échantillon d'un processus inconnu et la véritable difficulté est de trouver une fonction f qui saura être performante sur des données de ce processus auxquelles elle n'a pas été exposée durant son entraînement.

Introduisons l'**ensemble de test** (*test set*) S_{test}, qui est semblable à l'ensemble d'entraînement et contient des observations issues du même processus. Les données de l'ensemble de test sont par contre distinctes et ne sont utilisées que pour évaluer la performance de généralisation du modèle. Puisque S et S_{test} sont indépendants, on peut utiliser S pour trouver une fonction f et l'évaluer de manière non baisée en calculant son erreur sur S_{test} avec fonction d'erreur [**François 09**].

4. Ordonnancement et recherche d'Information :

L'ordonnancement est une technique couramment utilisée au sein de la recherche d'Information, les systèmes de recherche documentaire (SRD) prennent en entrée une requête utilisateur et renvoient une liste ordonnée de documents. Liste qui présente en premier les documents pertinents pour la requête. Elle est déterminée par une fonction d'ordonnancement.

En apprentissage, l'ordonnancement désigne la capacité d'apprendre à créer des listes ordonnées d'objets pour une requête de l'utilisateur. Les fonctions en ordonnancement ne cherchent donc plus à prédire une sortie par rapport à une entrée, mais à comparer les entrées entre elles et à les retourner sous forme de liste ordonnée [**Dammak 09**].

4.1 Ordonnancement d'instances :

Le problème d'ordonnancement d'instances est défini par une relation d'ordre $\succ$ sur l'espace d'instances *X*, permettant d'ordonner x_1 et x_2 pour toute paire d'instances (x_1, x_2) dans *X*. Une manière plus simple et naturelle de modéliser la relation d'ordre $\succ$ est d'utiliser une fonction $h: X \rightarrow \mathbb{R}$ qui attribue un score réel à toute instance $x \in X$. L'ordre entre les instances se déduit alors de la comparaison des valeurs de *h*. Ainsi, $h(x_1) > h(x_2)$ signifie que x_1 est ordonné au dessus de x_2.

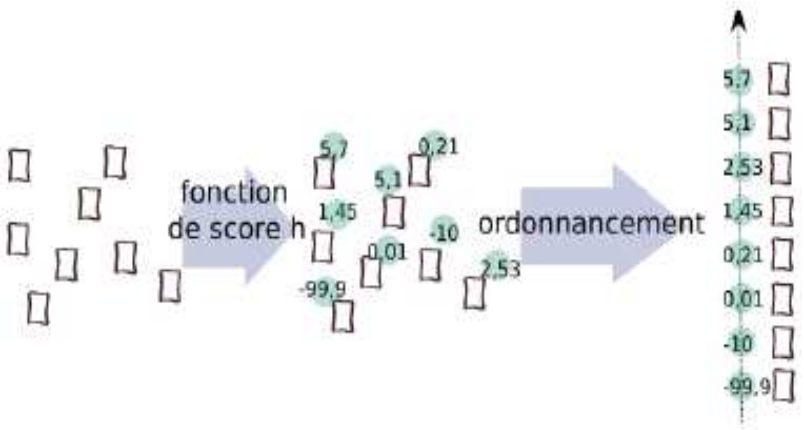


Figure IV.1 *Ordonnancement avec une fonction score* [**Nguyen 09**]

Dans la modélisation de la relation d'ordre, [**Jean et al 07**] formulent l'ordonnancement d'instances comme l'apprentissage d'une fonction de score, ainsi en classification et en régression. En revanche, ils soulignent une différence importante : en classification et en régression, les fonctions apprises donnent directement les prédictions attendues. En ordonnancement d'instances, les fonctions apprises renvoient des scores dont les valeurs absolues ne sont pas importantes en soi. En effet, ces valeurs ne servent qu'à comparer les instances les unes avec les autres. Ce sont donc les valeurs relatives des scores qui sont importantes.

4.1.1 Ordre souhaité induit par des scores :

A partir d'un ensemble d'observations $X = (x_1;\ldots x_m)$, [**Jean 08**] suppose que l'ordre désiré est induit par des scores $Y = (y_1;\ldots; y_m)$. L'ordre obtenu par ces scores est partiellement strict sur l'ensemble des entrées *X* :

Avec x_i ordonné au dessus de x_j si $y_i > y_j$

C'est le cas par exemple en routage d'information, où à chaque document est attribué un jugement de valeur indiquant sa pertinence par rapport à une thématique fixée *r* : *y=1* si le

document est pertinent par rapport à *r* sinon *y=-1*. C'est également le cas en filtrage collaboratif, où chaque utilisateur peut attribuer à chaque article un jugement de valeur exprimant ses préférences sur une échelle de notes : *y=5* s'il l'aimait beaucoup, *y*=1 dans le cas contraire. Dans les deux cas, des documents ou des articles possédants des scores différents peuvent être ordonnés les uns par rapport aux autres.

L'auteur suppose aussi qu'une partie des exemples $(x_i; y_i)$ est connue et disponible pour l'apprentissage. Le but de l'ordonnancement d'instances est d'apprendre une fonction de score qui doit retrouver un ordre souhaité à partir des exemples d'apprentissage.

Ainsi, il retrouve une tâche similaire à la classification et à la régression, où le but est d'apprendre une fonction à partir de quelques exemples afin de retrouver les sorties pour de nouvelles observations. En revanche, le but n'est plus de prédire les scores des exemples non observés, mais de prédire l'ordre entre les instances. Pour tenir compte de cette différence, l'auteur adapte les notions d'erreurs d'apprentissage au cadre de l'ordonnancement d'instances.

4.1.2 Fonction d'erreur d'ordonnancement :

Pour un ensemble donné d'instances, cette erreur permet de comparer l'ordre induit par la fonction score et celui souhaité (l'ordre induit par la valeur des étiquettes). Elle mesure ainsi à quel degré le premier ordre diffère du deuxième.

La fonction d'erreur d'ordonnancement est une fonction de la forme :

$$L_0 : \mathbb{R}^m \; X \; \mathbb{R}^m \longrightarrow \mathbb{R}^+$$

Autrement dit, la fonction va mesurer l'accord entre l'ordre induit par les scores renvoyés par la fonction et les jugements de pertinence [**Nicolas 06**].

4.1.3 Erreur de classification des paires cruciales :

L'erreur d'ordonnancement par paires d'entrées (x_i, x_j) telles que $y_i > y_j$, est appelées paires cruciales. Cette erreur notée L_{cp} estime la proportion de paires critiques sur lesquelles l'ordre prédit par *h* n'est pas l'ordre souhaité [**Cohen et al 97**].

$$L_{cp}(f(x), y) = \frac{1}{\sum_{ij} [[y_i > y_j]]} \sum_{y_i > y_j} [[h(x_i) \leq h(x_j)]]$$

Où $f(X) = (h(x_1);\ldots h(x_m))$ et $Y = (y_1;\ldots;y_m)$.

La motivation de cette fonction d'erreur vient d'une interprétation de l'ordonnancement en terme de reproduction d'une relation binaire d'ordre partiel : l'ordre partiel strict induit par

les scores y_i est une relation binaire, définie sur l'ensemble d'entrées considéré X. La fonction de score, elle aussi, permet de définir une relation binaire sur X. Il est alors naturel de mesurer l'erreur de la fonction de score par la comparaison d'une part de la relation binaire souhaitée, et d'autre part de la relation induite par la fonction de score.

Il est à noter que cette fonction a une relation avec la classification binaire intéressante, où le principe est que les scores induisent une relation binaire sur X. Une relation binaire considère une paire d'entrées, et a une valeur booléenne, par exemple 1 ou −1.

En considérant deux entrées x_i et x_j, la relation binaire induite par les scores vaut 1 si $y_i > y_j$, sinon −1. Ainsi, une relation binaire peut être vue comme un classifieur de paires d'entrées. Considérant ce point de vue, l'erreur L_{cp} peut être vue comme l'erreur de classification des paires cruciales ($y_i > y_j$) par la relation binaire induite sur X par la fonction de score : l'erreur d'ordonnancement mesurée par L_{cp} est une erreur de classification des paires cruciales.

Cette relation entre L_{cp} et la classification binaire a amené la communauté à s'intéresser principalement à cette fonction comme erreur d'ordonnancement [**Nicolas 06**].

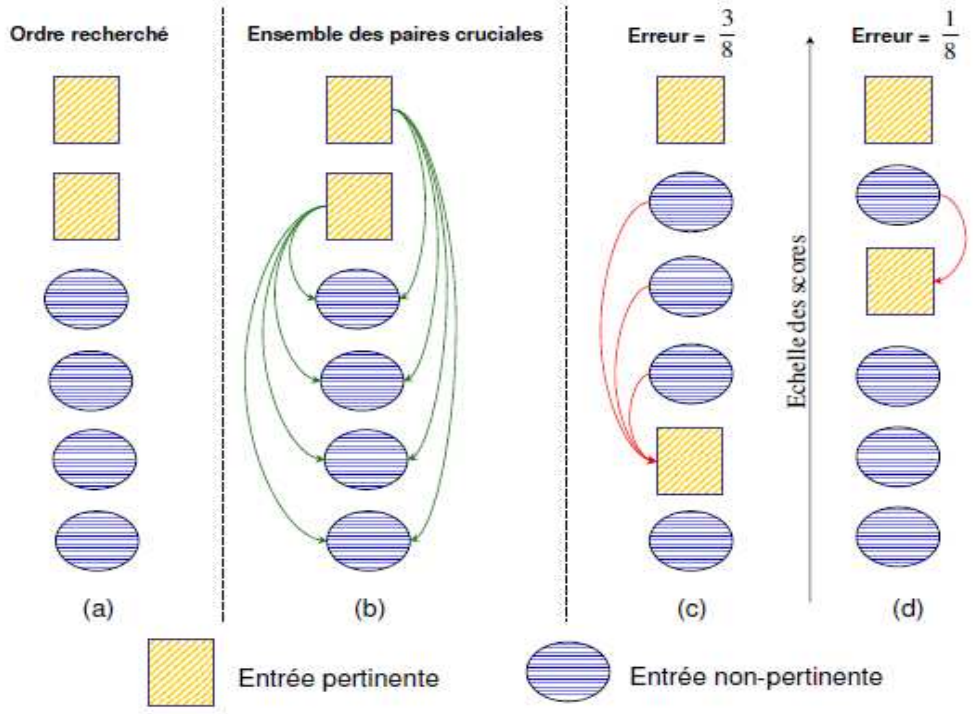


Figure IV.2 *Erreur de classification des paires cruciales* [**Nicolas 06**]

4.1.4 Relation entre ordonnancement d'instances et classification binaire :

Dans le cas de l'ordonnancement d'instance, la relation avec la classification binaire est immédiate. Le risque empirique sur un ensemble d'apprentissage $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ s''ecrit :

$$R_m^{OI}(h, S) = \frac{1}{\sum_{ij} [[\, y_i > y_j \,]]} \sum_{y_i > y_j} [[\, h(x_i) \leq h(x_j) \,]]$$

Notons h_{cp} le classifieur de paires associé à la fonction *h*, défini comme :

$$h_{cp}(x_i, x_j) = signe(h(x_i) - h(x_j))$$

Où *h* une fonction de score qui permet d'ordonner les instances de *X*.

La fonction $t \rightarrow signe(t)$ vaut −1 si $t \leq 0$ sinon 1, et considérons *T*, une fonction qui transforme l'ensemble d'apprentissage S en un nouvel ensemble d'apprentissage composé des paires cruciales :

$$T(S) = \left\{ \left((x_i, x_j), 1 \right) \middle| (i, j) \in \{1, ..., m\}^2 \, t.q. \, y_i > y_j \right\}$$

Il est alors aisé de voir que :

$$\frac{1}{\sum_{ij} [[\, y_i > y_j \,]]} \sum_{y_i > y_j} [[\, h(x_i) \leq h(x_j) \,]] = \frac{1}{|T(S)|} \sum_{((x_i, x_j), 1) \in T(s)} [[\, h_{cp}(x_i, x_j) \neq 1 \,]]$$

Ainsi, l'erreur empirique d'ordonnancement de *h* dans le cas de l'ordonnancement d'instances est l'erreur du classifieur de paires h_{cp} sur l'ensemble d'apprentissage transformé *T(S)*.

4.2 Ordonnancement d'instances linéaire :

Une fonction de score linéaire est de la forme $h(x) = b^T x$ avec $x \in R^S$ où $b \in R^S$ est un vecteur de paramètres. Notons h_{cp} le classifieur de paires associé à *h*, et $(x_i; x_j)$ une paire d'observations à classer. Il est important de savoir laquelle des deux instances est ordonnée au dessus de l'autre. La prédiction s'écrit :

$$h_{cp}(x_i, x_j) = b^T x_i - b^T x_j = b^T (x_i - x_j)$$

produit scalaire entre *b* et le vecteur $(x_i - x_j)$.

Afin de classer la paire $(x_i; x_j)$ avec le classifieur de paires h_{cp} revient à classer le vecteur $(x_i - x_j)$ avec le classifieur $h(x) = signe(b^T x)$. Ainsi, apprendre le classifieur de paires h_{cp} sur l'ensemble de paires *T(S)* revient à apprendre le classifieur binaire h_c sur l'ensemble d'exemples transformés *T`(S)* : $T'(S) = \left\{ \left((x_i - x_j), 1 \right) \middle| (i, j) \in \{1, ..., m\} t.q. \, y_i > y_j \right\}$

En conséquence, n'importe quel algorithme de classification linéaire peut être utilisé pour apprendre une fonction de score linéaire pour un problème d'ordonnancement. Il suffit d'utiliser, comme ensemble d'exemples, celui d'exemples transformés *T`(S)* construit à partir de *S*. C'est une propriété très importante d'un point de vue algorithmique, puisque tout algorithme de classification binaire peut être utilisé tel quel pour l'ordonnancement, sans modification. La seule différence entre ordonnancement d'instances et classification c'est qu'en ordonnancement, ce n'est pas l'ensemble d'exemples *S* qui sert de base d'apprentissage, mais l'ensemble d'exemples transformés *T`(S)* [**Jean 08**].

4.3 Ordonnancement biparti [**Amini 07**] *:*

L'ordonnancement biparti, est un cas particulier de l'ordonnancement d'instances. Il s'agit d'apprendre à partir d'un ensemble $S = \{x_i, y_i\}_i^n$ de *n* exemples $x_i \in R^S$ muni d'un jugement de préférence y_i. Dans le cas biparti, il est considéré un jugement de valeur binaire, qui reflète, pour un profil utilisateur donné, soit la pertinence d'un exemple *(y = 1)* soit sa non-pertinence *(y = −1)*. L'apprentissage consiste à trouver une fonction $h : R^S \rightarrow R$ qui donne un score plus élevé aux exemples pertinents qu'aux exemples non-pertinents.

Sur l'ensemble d'apprentissage $S = \{(x_1, y_1), ..., (x_n, y_n)\}$, en notant *k* le nombre d'exemples positifs (i.e. étiquetés 1) et *p* le nombre d'exemples négatifs, l'erreur d'ordonnancement empirique s'écrit alors : $R_{OI}^n(h, S) = \frac{1}{kp} \sum_{i:y_i=+1} \sum_{j:y_i=-1} [[\, h(x_i) \leq h(x_j)]]$

4.4 Relation avec l'aire sous la courbe ROC (Receiver Operator Characteristic):

Il s'avère que l'expression de R^{OI} *(h, S)* dans le cas biparti correspond à une quantité connue depuis longtemps, qui s'appelle aire sous la courbe ROC [**Cortes and Mohri 04**]. La courbe ROC permet de visualiser l'évolution du taux des faux pertinents *(FP)* par rapport aux taux des vrais pertinents *(TP)*. Pour une fonction score *h(x)*, la pertinence d'une instance peut être estimée en comparant son score à un seuil *b*.

Le classifieur obtenu a la forme *f (x) = signe (h(x) + b)*. La courbe est obtenue en faisant varier le seuil *b*. D'une façon générale, l'aire sous cette courbe est un indicateur de la capacité de la fonction réelle à ordonner les exemples positifs (Pertinents) au dessus des négatifs (non-pertinents) [**Nguyen 09**].

4.5 Apprentissage d'une fonction d'ordonnancement :

L'apprentissage d'une fonction d'ordonnancement (Learning To Rank) est une approche d'apprentissage automatique, qui construit automatiquement une fonction d'ordonnancement à partir de données d'apprentissage.

En Learning to Rank, il s'agit de considérer une collection de couples (requête, document) pour lesquels la pertinence est connue. Une fonction d'ordonnancement est apprise à l'aide d'un algorithme d'apprentissage sur ce jeu de données. Cette fonction est ensuite utilisée pour prédire la pertinence de nouvelles paires (requête, document) et fournir un classement [**Léa 10**].

La figure IV.3 ci-dessous montre le cadre générale de l'apprentissage d'une fonction d'ordonnancement. La création de la base d'apprentissage est très similaire à la création de la base de test pour l'évaluation. Supposons qu'il existe une collection de *n* requêtes pour l'apprentissage, notées par $q_i(i=1,...,n)$, leurs documents associés représentés par des vecteurs de caractéristiques $X^{(i)} = \{x_j^{(i)}\}_{j=1}^{m}$ (où *m(i)* est le nombre de documents associés à la requête q_i), et $Y = \left(y^{(1)},...y^{(m^{(i)})}\right)$ les jugements de pertinence (Veuillez distinguer le jugement pour l'évaluation et le jugement pour construire l'ensemble d'apprentissage, bien que le processus puisse être très semblable). Le rendement du modèle d'ordonnancement peut prévoir l'étiquette réelle dans l'apprentissage le plus exactement possible, en termes de fonction d'erreur. Dans la phase de test, quand une nouvelle requête entre, le modèle appris dans la phase d'apprentissage est appliqué pour assortir les documents et renvoyer la liste ordonnée correspondante à l'utilisateur comme réponse à sa requête.

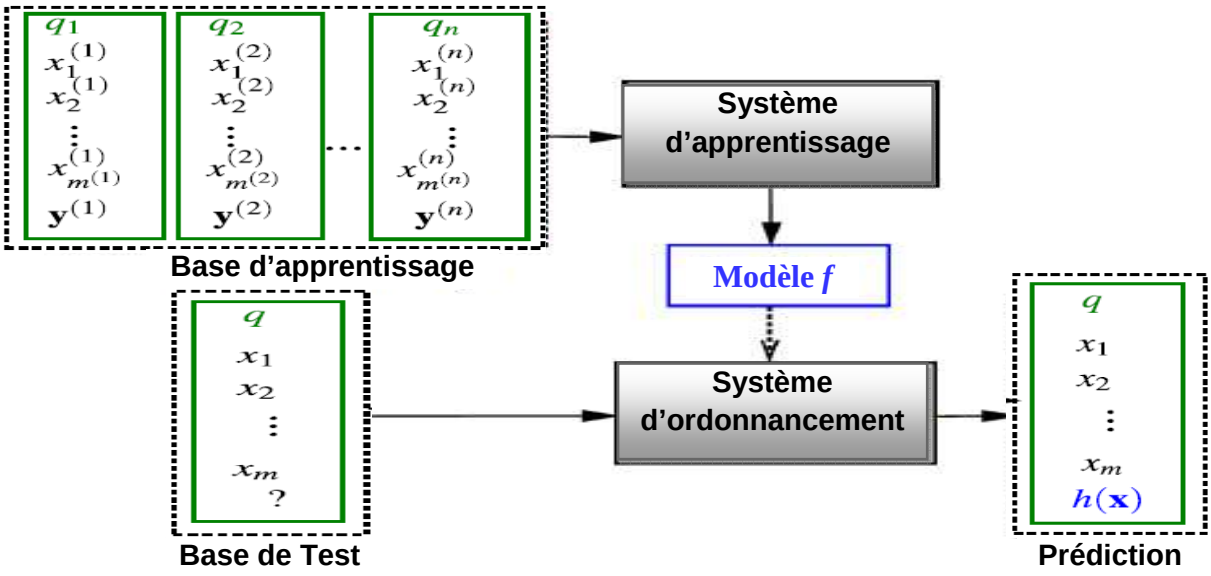


Figure IV.3 *Le cadre générale de l'apprentissage d'une fonction d'ordonnancement* [**Lui 09**]

Plusieurs algorithmes d'ordonnancements peuvent s'insérer dans le cadre ci-dessus. Afin de les mieux comprendre, Tie-Yan-Lui [**Lui 09**] en propose d'effectuer une catégorisation. En particulier, nous les groupons selon trois approches : l'approche pointwise, pairwise et listwise. Ces dernières modélisent le processus d'ordonnancement sous différentes manières. C'est-à-dire, ils peuvent définir divers espaces d'entrée et de sortie, en utilisant différentes fonctions d'ordonnancements et de coûts. Le tableau ci-dessous illustre une comparaison entre ces trois approches :

Catégorie	**Pointwise**	
	Régression	**Classification**
Espace d'entrée	Un seul document comme instance d'apprentissage	
Espace de sortie	Variable à prédire ***Y*** est un score (note) de pertinence	-Dichotomique {-1,1} (non pertinent, pertinent) - Présenter plusieurs classes (non pertinent, pertinent faible, pertinence moyenne, pertinence forte)
Fonction d'ordonnancement	$f(x_j)$	$f(x_j)$
Fonction d'erreur	$L(f(x_j),y_j)=\left\|f(x_j)-y_j\right\|^2$	$L(f(x_j),y_j)=[[f(x_j)\neq y_j]]$
Catégorie	**Pairwise**	**Listwise**
Espace d'entrée	Paire de documents (x_u,x_v)	Liste de documents $X=\{x_j\}_{j=1}^{m}$
Espace de sortie	Préférence y_{uv}	Liste ordonnée documents π_y
Fonction d'ordonnancement	$I_{\{f(x_u)>f(x_u)\}}-1$	$Sort\circ f(x)$
Fonction d'erreur	$L(f;x_u,x_v,y_{u,v})$	$L(f;X,\pi_y)$

Tableau IV.1 *Résumé des approches de l'apprentissage d'une fonction d'ordonnancement*

4.5.1 Approche pointwise :

L'approche pointwise essaye d'utiliser simplement les méthodes d'apprentissage automatique par leurs applications directes au problème d'ordonnancement. Cette approche apprend une fonction d'ordonnancement par la minimisation d'une fonction d'erreur qui est défini par le jugement de pertinence du document individuel et le score d'ordonnancement pourrait être basé sur la régression, de la classification ou de la régression ordinale.

Cette approche suppose que le degré exact de la pertinence de chaque document est l'objectif de la prédiction. Cependant, cette hypothèse n'est pas nécessaire du fait que l'objectif est de produire une liste ordonnée des documents et le score individuel des documents reste non important.

Régression ordinale :

La régression ordinale peut être considéré comme une classification multi–classes avec un ordre total sur les étiquettes, où les étiquettes prennent des valeurs dans un espace *Y* où il existe un ordre total. Sans perte de généralité, nous supposons *Y = {1; … ; K}*.

Une approche courante est d'apprendre une fonction réelle ainsi que K-1 seuils $\theta_1,...,\theta_{k-1}$ pour déterminer l'étiquette. Cela revient à partitionner l'ensemble des réels et à associer à chaque partition une étiquette. La prédiction se fait alors en deux étapes : on attribue un score *h(x)* à une instance x et on lui donne ensuite l'étiquette associée à la partition dans laquelle le score se trouve.

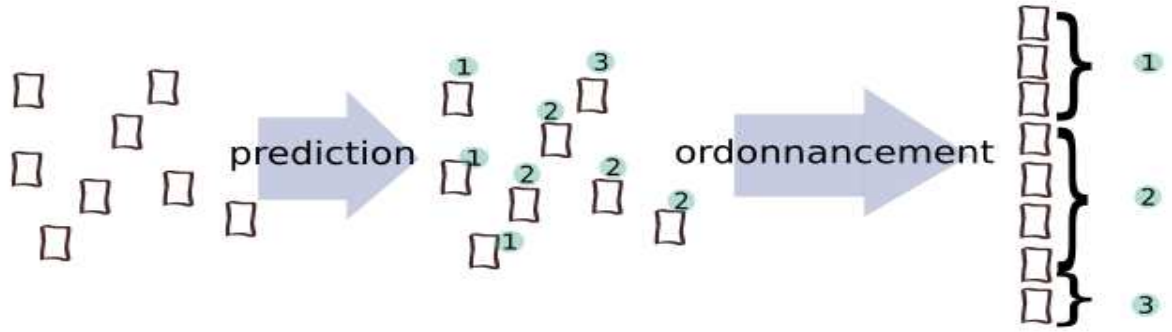


Figure IV.4 *Ordonnancement dans le cadre de la régression ordinale*

Le but de la régression ordinale est de prédire l'étiquette mais en cas d'erreur, l'étiquette prédite doit être la plus proche possible de la vraie étiquette [**Nguyen 09**]. Par exemple, pour une étiquette valant 5, il est préférable de prédire 4 au lieu de 3.

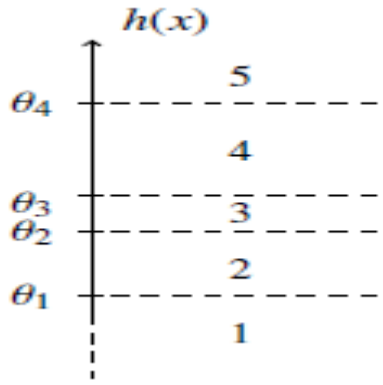


Figure IV.5 *Exemple de fonction de décision en régression ordinale*

4.5.2 Approche pairwise :

Contrairement à l'approche pointwise qui met l'accent sur la prédiction du degré de pertinence de chaque document individuel, l'approche pairwise se concentre sur l'ordre relatif entre chaque paire de documents. Les algorithmes d'ordonnancement fondés sur l'approche pairwise tentent d'attribuer des notes plus élevées aux documents auxquels sont attribuées des étiquettes (labels) de pertinence plus élevées par des évaluateurs humains. La plupart de ces approches tentent de minimiser le nombre de paires de documents qui sont mal ordonnés. Pour ce faire, le problème d'ordonnancement est généralement réduit à une classification binaire sur des paires de documents ; l'objectif de l'apprentissage vise alors à minimiser le nombre de paires de documents classés à tort.

Beaucoup d'algorithmes d'ordonnancement pairwise ont été proposés dans la littérature de l'apprentissage de fonction d'ordonnancement. Ils utilisent les réseaux neurones, boosting, SVM, et d'autres méthodes d'apprentissages afin d'établir le modèle d'ordonnancement. Parmi ces algorithmes on trouve:

RankSVM :

RankSVM [**Herbrich et al 00**] est un algorithme de la famille SVM (*Support Vector Machine*) [**Vapnik 00**] pour l'ordonnancement des paires d'exemples. La fonction de score *H* pour chaque couple document-requête (*d, q*) est linéaire dans un espace de caractéristiques, sous-ensemble de R^n :

$$H(d,q) = W^T x(d,q)$$

Où *W* est un vecteur des poids à estimer et *x*(*d, q*) est une description vectorielle du document *d* et de la requête *q*, *n* est la dimension du vecteur de paramètre *W*. Pour une requête q_i et une paire de documents $(d_j, d_k) \in X_q^1 \; x \; X_q^0$, nous notons $x_{ijk} \equiv W^T (x_{ij} - x_{ik})$ avec x_{ij} et x_{ik} les représentations vectorielles des couples (d_j, q_i) et (d_k, q_i). Pour une base d'apprentissage de *N* paires (x_{i1}, x_{i0}), où x_{i1} est à classer au-dessus de x_{i0}, RankSVM résout un problème d'optimisation quadratique (QP) :

$$\min_W \sum_i^N \left[1 - H(x_{i1}) + H(x_{i0})\right]_+ + \lambda \|W\|^2$$

où $[z]_+ = \max(0, z)$, dénommée fonction *hinge*, est une fonction convexe (*cf.* figure VI.6).
Autrement dit RankSVM utilise la fonction $[z]_+$ pour approximer *PairLoss*. Le deuxième terme de la fonction est un terme de régularisation qui améliore la généralisation sur de

nouveaux exemples. λ est un hyper-paramètre de RankSVM qui permet de pondérer le poids des deux termes de la somme.

Le problème de minimisation sous contrainte équivalent est le suivant :

$$\min_{W,\{\xi_{ijk}\}} \quad \frac{1}{2}\|W\|^2 + C\sum_{ijk}\xi_{ijk}$$

$$s.c \quad W^T x_{ijk} \geq 1 - \xi_{ijk} \; \forall i = 1,..., \quad Q; j = 1,..., R_i; k = 1,..., NR_i$$

$$\xi_{ijk} \geq 0, \forall i,..., Q; j = 1,..., R_i; k = 1,..., NR_i$$

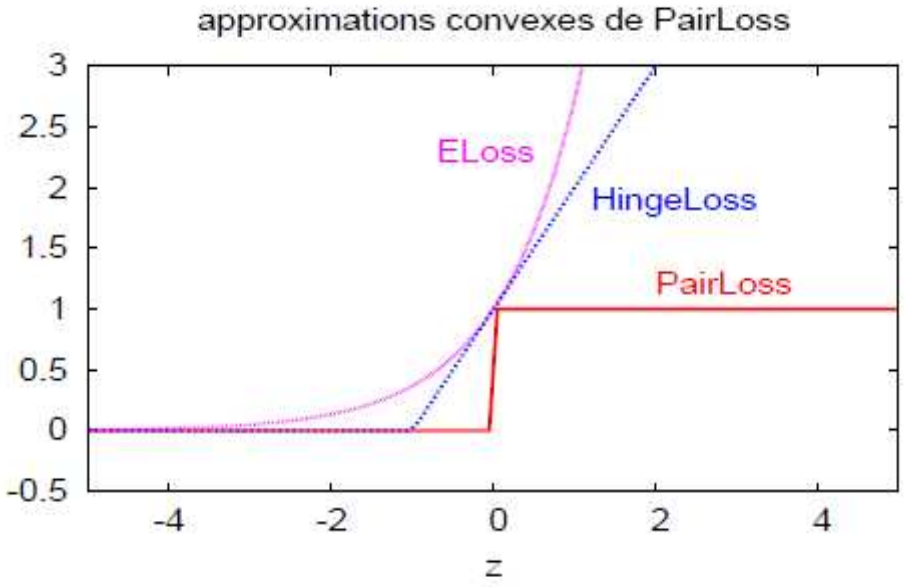


Figure IV.6 *La courbe d'erreur PairLoss et deux approximations convexes : l'erreur hinge (HingeLoss utilisée en RankSVM) et l'erreur exponentielle (ELoss utilisée en Rank-Boost).*

$z = H(x0) - H(x1)$, *PairLoss*(z) = signe(z), *HingeLoss*(z) = [1 + z]+ =max(0, 1 + z),

et *ELoss*(z) = e^z

Où $C = \frac{1}{2\lambda}$, Q est le nombre de requêtes, R_i est le nombre de documents pertinents pour la requête q_i et NR_i le nombre de documents non pertinents. La somme des variables $\{\xi_i\}, \xi_i \geq 0 \forall i = 1,..., N$ donne une borne supérieure de nombre de paires mal ordonnées $(N = \sum_{i=1}^{Q} R_i.NR_i)$.

Le problème dual équivalent est le suivant :

$$\max_{\alpha} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{Q}\sum_{j,j'=1}^{R_i}\sum_{k,k'=1}^{NR_i}\alpha_{i,j,k}\alpha_{i,j',k'}x_{i,j,k}^T x_{i,j',k'}$$

$$S.C. \quad \sum_{i=1}^{N}\alpha_i = 0$$

$$0 \leq \alpha_i \leq C, \forall i = 1,..., N$$

L'apprentissage d'un modèle RankSVM revient alors à résoudre un problème d'optimisation quadratique (QP) comme dans le cas SVM habituel. Plusieurs méthodes d'optimisation peuvent être utilisées : soit des méthodes génériques pour des problèmes QP, soit des méthodes spécifiques aux SVM développées par la communauté d'apprentissage statistique [**Huyen 08**].

RankBoost :

RankBoost [**Freund et al 03**] est un algorithme de type *boosting* dédié à la tâche d'ordonnancement, a pour but d'estimer une fonction de score pour chaque document-requête (*d, q*) en minimisant itérativement l'approximation exponentielle suivante de *PairLoss* :

$$ELoss(H, D) = \sum_{(x_1 x_0)} D(x_1, x_0) \exp[H(x_0) - H(x_1)]$$

Où $D(x1, x0)$ pondère l'importance de la paire (x_1, x_0) pendant la phase d'apprentissage.
Plus cette quantité est élevée, plus la paire est difficile à ordonner. RankBoost va déterminer itérativement une fonction $H_T = \sum_{t=1}^{T} \alpha_t h_t$, en cherchant à l'itération *t* une fonction h_t et son poids α_t qui minimisent $ELoss(h_t, D_t)$. A l'´etape *t*, D_t est une pondération des paires qui met plus de poids sur celles pour lesquelles la fonction $H_{t-1} = \sum_{k=1}^{t-1} \alpha_k h_k$ se trompe.

Définissons le coût à l'itération *t* par :

$$Z_t = ELoss(h_t, \alpha_t, D_t) = \sum_{(x_1 x_0)} D_t(x_1, x_0) \exp[\alpha_t (h_t(x_0) - h_t(x_1))]$$

Il est aisé de montrer que pour tout *T*, $PairLoss(H_T) \leq \prod_{t=1}^{T} Z_t$ [**Freund et al 03**]. Cette borne montre que l'on peut minimiser $PairLoss(H_T)$ itérativement en minimisant Z_t à chaque itération.

4.5.3 Approche listwise :

L'approche listwise essaye d'apprendre un modèle qui prédit l'ordonnancement réel des documents associés à une requête. Cette approche résout le problème d'ordonnancement par la minimisation d'une fonction d'erreur définie sur la liste prédite et la liste d'apprentissage.

Selon les fonctions d'erreurs utilisées, l'approche peut être divisée en deux sous-catégories :

Pour la première sous-catégorie, la fonction d'erreur est explicitement liée à des mesures d'évaluation. En raison de la forte relation entre les fonctions d'erreurs et les mesures d'évaluation, les algorithmes utilisés sont appelées méthodes d'optimisation directes.

Pour la deuxième sous-catégorie, la fonction d'erreur n'est pas explicitement liée à des mesures d'évaluation. Plusieurs méthodes ont été proposées parmi lesquelles, ListNet et ListMLE qui sont des méthodes probabilistes, proposées respectivement par Cao et al. **[Cao et al 07]** et Xia et al. [**Xia et al 08**].

ListNet :

ListNet, définie une fonction d'erreur qui utilise la distribution de probabilité sur les permutations. Actuellement, les distributions de probabilité sur les permutations ont été bien étudiées dans le domaine de la théorie des probabilités. Beaucoup de modèles ont été proposées pour représenter les distributions de probabilité de permutation, tels que le modèle Plackett-Luce [**Luce 59**, **Plackett 75**] et le modèle Mallows [**Mallows 75**].

ListNet est juste un exemple, montrant comment appliquer le modèle Plackett-Luce à l'apprentissage de fonction d'ordonnancement. Étant donné les scores d'ordonnancements des documents fournis par la fonction *f* en sortie ($S = \{S_j\}_{j=1}^{m}$, où $S_j=f(x_j)$), le modèle Plackett-Luce définit une probabilité pour chaque permutation π possible des documents:

$$P(\pi|S)=\prod_{j=1}^{m}\frac{\varphi(S_{\pi^{-1}(j)})}{\sum_{u=1}^{m}\varphi(S_{\pi^{-1}(u)})}$$

Où $\pi^{-1}(j)$ désigne le document classé à la j'i^{ème} position de π permutation et φ est une fonction de transformation, qui peut être linéaire ou exponentielle.

ListNet définit premièrement la distribution de probabilité de permutation basée sur les scores attribués par la fonction *f*. Ainsi qu'elle définit une autre distribution de probabilité de permutation $P_{y(\pi)}$ basé sur les étiquettes réelles qui représentent le degré de pertinence des documents. Il peut être substitué directement dans le modèle Plackett-Luce afin d'obtenir une distribution de probabilité. L'algorithme ListNet optimise l'entropie croisée entre la distribution des rangs des listes prédite et désirée, où la fonction d'erreur est définit de la manière suivante :

$$L(y_i,F(x_i;\theta))=-\sum_{g\in G_i^k}P_{Y_i}(g)\log P_{F(x_i;\theta)}(g)$$

$$=-\sum_{g\in G_i^k}\prod_{j=1}^{k}\frac{\exp(y_{ij})}{\sum_{l=j}^{n_i}\exp(y_{il})}\log\prod_{j=1}^{k}\frac{\exp(f(x_{ij};\theta))}{\sum_{l=j}^{n_i}\exp(f(x_{il};\theta))}$$

ListNet utilise le gradient décent pour optimiser cette fonction d'erreur

$$\frac{\partial L(\theta)}{\partial \theta} = \sum_{i=1}^{m} \frac{\partial L(y_i, F(x_i, \theta))}{\partial \theta}$$

$$\frac{\partial L(y_i, F(x_i, \theta))}{\partial \theta} = -\sum_{g \in G^k} \left(\frac{P_{Y_i}(g)}{P_{F(x_i;\theta)}(g)} \right) \left(\frac{\partial P_{F(x_i;\theta)}(g)}{\partial \theta} \right)$$

ListMLE :

Elle base sur une distribution exponentielle des scores de pertinence et maximise la vraisemblance entre les distributions des scores de la liste de candidats prédite par le système et les jugements de la liste désirée.

5. Apprentissage semi-supervisé et actif de fonctions d'ordonnancement :

Dans le formalisme de la tâche d'ordonnancement semi-supervisé, l'ensemble d'apprentissage est constitué d'instances étiquetées dont l'étiquette reflète un jugement de pertinence ainsi qu'un ensemble d'instances non étiquetées. Il s'agit ainsi d'apprendre une fonction score à partir d'une base d'instances étiquetées.

Dans la littérature, il existe deux paradigmes en apprentissage semi-supervisé : l'apprentissage inductif et transductif. Le cadre transductif s'intéresse uniquement à étiqueter les instances non-étiquetées de la base d'apprentissage. Par conséquent, l'évaluation se fait suivant l'habilité du modèle à s'acquitter de sa tâche sur les instances non-étiquetées de la base d'apprentissage. Le cadre inductif a un tout autre objectif : le but est de pouvoir ordonner n'importe quel ensemble de données. L'inférence est donc au cœur du paradigme inductif.

L'apprentissage actif vise à réduire le coût d'étiquetage en sélectionnant de façon intelligente les exemples à étiqueter. L'objectif est d'obtenir des bonnes performances avec le moins possible d'exemples d'apprentissage. Il existe deux approches différentes : les approches constructives et les approches sélectives [**Settles 09**].

Les approches constructives consistent à demander des étiquettes pour n'importe quelles données de l'espace de départ. Par contre, l'approche sélective permet de résoudre ce problème en se focalisant uniquement sur les données non-étiquetées.

Algorithme :

Entrée

Un petit ensemble de données étiquetés S_L

Un large ensemble de données non-étiquetés S_U

Répéter

Entrainement du modèle sur S_L

Utilisation de modèle pour tester S_U

Selection de l'exemple le plus utile de S_U

Interrogation d'un expert humain pour l'étiqueter

Addition de l'exemple étiqueté à S_L

Jusqu'à modèle atteigne un certain niveau de performance ou un certain nombre de requêtes.

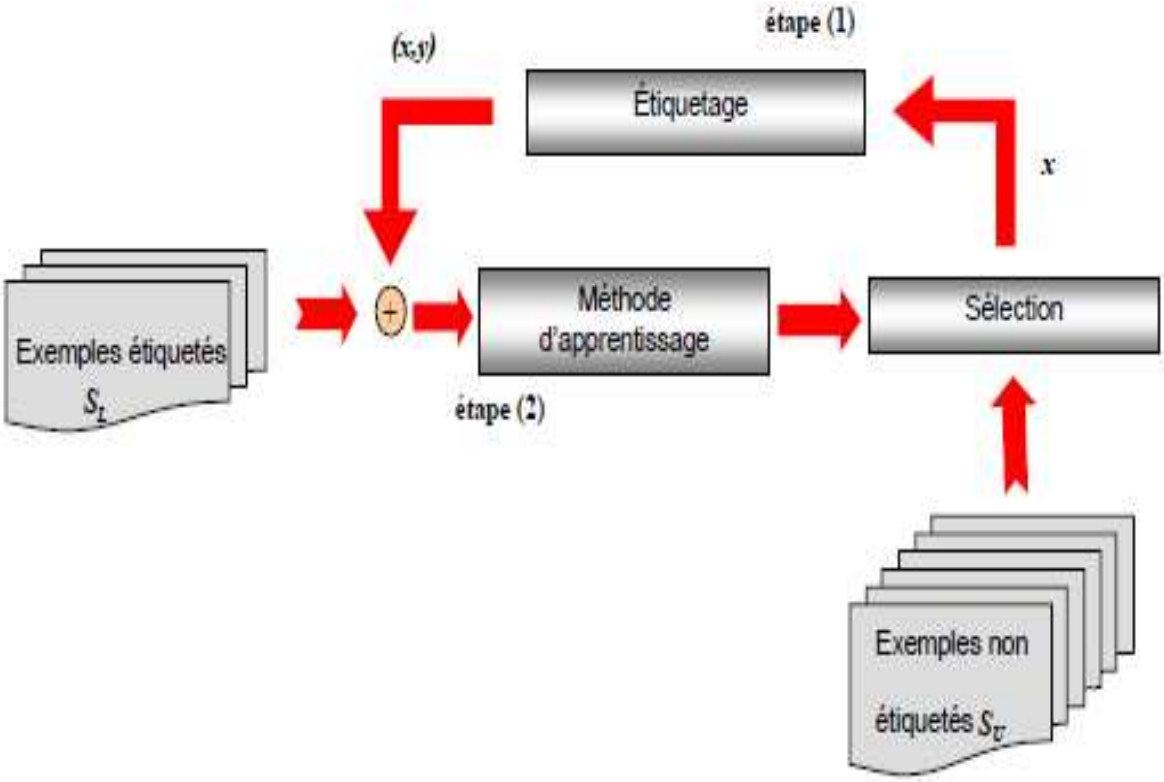


Figure IV.7 *Approche sélective en apprentissage actif* [**Nguyen 09**]

6. Ordonnancement pour le filtrage collaboratif :

Actuellement et afin d'améliorer la qualité de la recommandation, l'attention de la recherche dans le domaine du filtrage collaboratif a déplacé le problème de prédiction de note en un problème de prédiction d'ordre, qui permet d'ordonner directement les articles, sans essayer de prédire les notes d'un utilisateur sur les articles non notés comme une étape intermédiaire. Il existe des méthodes qui utilisent la préférence par paires (pairwise) d'articles des utilisateurs. Parmi elles l'ordonnancement collaboratif avec la borne exponentielle **[Jean el al 07],** l'EigenRank [**Liu et al 08**] et probabilistic latent preference analysis [**Liu et al 09**]. D'autres méthodes ont été proposées telles que CoFiRank [**Weimer et al 07**] qui optimise directement la mesure d'ordonnancement NDCG (Normalized Discounted Cumulative Gain) et ListRank-MF [**Shi 10**] qui consiste à combiner une des méthodes de l'approche listewise avec la méthode de factorisation matricielle probabiliste FMP.

6.1 Méthode EigenRank :

Le principe de cette méthode est de mesurer la similarité entre les utilisateurs en fonction de la corrélation entre leurs ordonnancements des articles plutôt que les valeurs de notation et de proposer de nouveaux algorithmes de filtrage de collaboration pour l'ordonnancement des articles bases sur les préférences des utilisateurs similaires.

6.2 Méthode pLPA (probabilistic latent preference analysis) :

Le modèle pLPA : aborde le problème d'ordonnancement des articles directement par la modélisation des préférences par paires sur les articles en utilisant un modèle classe latente basé sur le modèle de Bradley-Terry pour les comparaisons par paires.

6.3 Ordonnancement collaboratif avec la borne exponentielle :

Dans cette méthode, les auteurs définissent une erreur d'ordonnancement qui prend en compte les préférences par paires d'articles, où ils définissent une fonction de coût *D* comme étant le nombre d'erreurs de prédiction sur les préférences par paires de la base d'apprentissage $(a, b) \in y_x$ tel que :

$$y_x = \{(a,b) | a \in y, b \in y, r_x(a) > r_x(b)\}$$

où $r_{x(a)}$ est la note fournie par l'utilisateur *x* pour l'article *a*. Chaque préférence par paire $(a, b) \in y_x$ exprime que l'utilisateur x préfère l'article *a* à l'article *b*.

$$D(U,I) = \sum_x \sum_{(a,b)\in y_x} [[(UI)_{xa} \le (UI)_{xb}]]$$

où [[***pr***]] vaut 1 si ***pr*** est vraie, sinon 0. Le but de l'apprentissage est de déterminer les matrices de paramètres *U* et *I* qui minimisent cette erreur. C'est un problème d'optimisation difficile car la fonction *D* n'est pas dérivable. Grâce à l'inégalité $[[x \le 0]] \le e^{-x}$, nous proposons de borner *D* de la façon suivante :

$$D(U,I) \le \underbrace{\sum_x \sum_{(a,b)\in y_x} e^{(UI)_{xa} - (UI)_{xb}}}_{\mathcal{E}(U,I)}$$

La fonction $\mathcal{E}$ n'est pas convexe en *U* et *I* simultanément, en revanche elle est convexe en chacune des matrices séparément. Afin de pouvoir régulariser la factorisation, ils ont ajouté également deux coefficients $\mu_U, \mu_I \ge 0$ pénalisant les normes de *U* et *I*. Finalement, ils ont obtenu la fonction R :

$$R(U,I) = \sum_x \sum_{(a,b)\in y_x} e^{(UI)_{xa} - (UI)_{xb}} + \mu_U \|U\|^2 + \mu_I \|I\|^2$$

Puis ils ont construis un algorithme efficace qui optimise cette fonction d'erreur, où ils ont adopté une stratégie de minimisation alternée, qui consiste à fixer alternativement une des deux matrices et à minimiser *R* par rapport à l'autre grâce à la méthode du gradient conjugué. Les formules qui permettent de calculer les gradients de *R* par rapport à *U* et à *I*, qui sont nécessaires à l'optimisation sont :

$$\frac{\partial R}{\partial I_{jd}} = \sum_x \left[\sum_a U_{xd} e^{(I^T u_x)_j - (I^T u_x)_a} \delta_{ja}^x - \sum_b U_{xd} e^{(I^T u_x)_b - (I^T u_x)_j} \delta_{bj}^x \right] + 2\mu_I I_{jd}$$

$$\frac{\partial R}{\partial U_x} = \sum_{(a,b)\in y_x} (i_b - i_a) e^{(I^T u_x)_b - (I^T u_x)_a} + 2\mu_U u_x$$

où u_x est la *x*-ième ligne de *U*, i_a est la a-i^{ème} colonne de *I*, $(I^T u_x)_j$ est la *j*-ième composante du vecteur $I^T u_x$, et δ_{ja}^x vaut 1 si $(j, a) \in y_x$, 0 sinon **[Jean et al 07]**.

6.4 Méthode CoFiRank :

La méthode CoFiRank à pour principe, de chercher en priorité à ordonner correctement les articles en haut de la liste, sans accorder une importance égale à tous les articles. Cette propriété est désirable notamment en filtrage collaboratif, où pour un utilisateur donné il est

important qu'un article favori se retrouve en haut de la liste (donc parmi les articles recommandés). En revanche si un article en bas de la liste est mal ordonné, cela n'a pas d'importance.

Pour atteindre ce but, les auteurs [**Weimer et al 07**] s'intéressent au critère DCG (Discounted Cumulative Gains [**Kalervo et al 00**]) et de sa variante normalisée NDCG.

Concentrons-nous sur le critère DCG. Soit $y \in \{1;...;r\}^r$ un vecteur de notes et π une permutation de *y*. Notons π_i la position de l'article i dans la permutation. Soit $k \in N$ un seuil, et π_s la permutation de *y* dans l'ordre décroissant de ses éléments. Le score DCG@k associé à la permutation π vaut :

$$DCG@k(y,\pi) = \sum_{i=1}^{k} \frac{2^{y_{\pi_i}} - 1}{\log(i+2)}$$

Ce score est maximal pour $\pi = \pi_s$. Le seuil k représente le nombre d'articles qu'un utilisateur est prêt à considérer. Remarquons que le score DCG repose sur les permutations du vecteur de notes *y*, et ne dépend pas des valeurs des notes. De plus les auteurs montrent que les positions en haut de la liste ont plus d'influence sur le score que les positions en bas de la liste. Autrement dit l'optimisation de ce critère permet de déterminer une permutation des éléments respectant l'ordre défini par *y*, et favorisant les éléments en haut de la liste.

Dans le cadre de l'ordonnancement, nous ne prédisons pas des permutations mais des scores qui induisent un ordre entre les observations. Ainsi les auteurs définissent une matrice *F* de taille *(n X m)*, dont les éléments F_i vont permettre d'ordonner l'article *i* pour l'utilisateur. Pour mesurer la qualité de l'ordonnancement induit par *F*, les auteurs définissent l'erreur :

$$R(F,Y) = \sum_{l=1}^{n} DCG@k(\Pi_l, Y_l)$$

où Π_l : est la permutation de F_l: dans l'ordre décroissant de ses éléments. Autrement dit en optimisant cette fonction, nous cherchons la matrice de scores F permettant d'ordonner les articles pour chaque utilisateur en favorisant les articles en haut de la liste.

L'objectif de CoFiRank est donc de maximiser $R(F; Y_{test})$ en ne connaissant que la quantité $R(F; Y_{app})$.

En pratique la fonction $R(F; Y)$ est difficile à optimiser, et les auteurs déterminent d'abord une borne inférieure de *R*, qu'ils optimisent à l'aide des bundle méthodes. Après optimisation, la matrice *F* permet d'ordonner les articles pour chaque utilisateur. Les auteurs

montrent que leur algorithme peut également être utilisé dans le cadre de la régression multi-tâches. Autrement dit CoFiRank est également capable de prédire des notes. Expérimentalement, les auteurs valident leur algorithme sur plusieurs bases standard de filtrage collaboratif.

6.5 Méthode ListRank-MF :

La méthode ListRank-MF consiste à combiner la méthode de cross-entropy de top one probabilités avec la méthode de factorisation matricielle probabiliste *(FMP)* telle que la fonction d'erreurs définis comme suit:

$$L(U,V)=\sum_{i=1}^{M}\left\{-\sum_{j=1}^{N}(R_{ij})\log P_{l_i}(g(U_i^TV_j))\right\}+\frac{\lambda}{2}(\|U\|_F^2+\|V\|_F^2)$$

$$=\sum_{i=1}^{M}\left\{-\sum_{j=1}^{N}I_{ij}\left(\frac{\exp(R_{ij})}{\sum_{k=1}^{N}I_{ik}\exp(R_{ik})}\right)\log\left(\frac{\exp(g(U_i^TV_j))}{\sum_{k=1}^{N}I_{ik}\exp(g(U_i^TV_k))}\right)\right\}+\frac{\lambda}{2}(\|U\|_F^2+\|V\|_F^2)$$

Cette fonction reflète l'incertitude entre les listes d'apprentissage (en entrée) et les listes résultantes (en sortie) provenant du modèle d'ordonnancement. La sortie du modèle de recommandation est une liste de recommandations relative à chaque utilisateur i et est générée par l'ordonnancement des articles de la collection dans l'ordre décroissant en fonction de la valeur U_i^TV .

Etant donné que la fonction *L(U,V)* n'est pas convexe en *U* et *V* simultanément, Il a été choisie l'utilisation de la méthode descendante du gradient qui consiste à fixer une des deux matrices et minimiser *L* par rapport à l'autre, à partir de laquelle un minimum local peut être obtenu. Les gradients de *L (u, v)* par rapport à *U* et *V* peuvent être calculé comme suit:

$$\frac{\partial L}{\partial U_i}=\sum_{j=1}^{N}I_{ij}\left(\frac{\exp(g(U_i^TV_j))}{\sum_{k=1}^{N}I_{ik}\exp(g(U_i^TV_k))}-\frac{\exp(R_{ij})}{\sum_{k=1}^{N}I_{ik}\exp(R_{ik})}\right)g'(U_i^TV_j)V_j+\lambda U_i$$

$$\frac{\partial L}{\partial V_j}=\sum_{i=1}^{M}I_{ij}\left(\frac{\exp(g(U_i^TV_j))}{\sum_{k=1}^{N}I_{ik}\exp(g(U_i^TV_k))}-\frac{\exp(R_{ij})}{\sum_{k=1}^{N}I_{ik}\exp(R_{ik})}\right)g'(U_i^TV_j)V_j+\lambda V_j$$

Notez que *g '(x)* désigne la dérivée de *g (x)*.

6.5.1 Factorisation matricielle probabiliste (FMP) :

Le cadre FMP a été proposé par Ruslan Salakhutdinov [**Ruslan et al 08**], où la factorisation matricielle est formulée à partir d'inférence probabiliste de la distribution conditionnelle de notes observées. Le cadre ultime est formulé comme suit:

$$U,V = \underset{U,V}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} I_{ij}\left(R_{ij} - g(U_i^T V_j)\right)^2 + \frac{\lambda_u}{2}\|U\|_F^2 + \frac{\lambda_v}{2}\|V\|_F^2$$

Supposons *R* la matrice de notes utilisateur-article, constituée de *M* utilisateurs et de *N* articles, PMF cherche à représenter la matrice utilisateur-article *R* avec deux matrices de dimension inférieure, où *d* est la dimension de *U* et *V* qui représentent les caractéristiques latentes pour les utilisateurs (dans *U*) et des articles (dans *V*). U_i de d-dimensions est utilisé pour désigner le vecteur colonne de caractéristiques de l'utilisateur *i* et V_j de *d*-dimensions pour désigner le vecteur colonne de caractéristiques de l'article *j*.

I_{ij} est une fonction indicatrice qui est égale à 1 lorsque $R_{ij} > 0$, sinon 0.

g (x) est une fonction logistique définis comme suit : $g(x) = \frac{1}{(1+\exp(-x))}$

6.5.2 Top One Probabilité :

Proposée par Zhe Cao dans [**Cao et al 07**], top One Probabilité indique la probabilité d'un article ordonné dans la position supérieure d'une liste ordonnée de données. Cette probabilité est exprimée comme suit:

$$P_{l_i}(R_{ij}) = \frac{\varphi(R_{ij})}{\sum_{k=1}^{M}\varphi(R_{ij})}$$

Où $\varphi(x)$ représente la fonction exponentielle

R_{ij} étant la note de l'article

7. Métriques d'évaluation :

Parmi les mesures qui évaluent la qualité et l'efficacité des systèmes de recherche d'information et de recommandation il existe:

7.1 Précision :

La précision est la proportion des articles pertinents dans la liste des articles retournés par le système, ce qui correspond au pourcentage ou au nombre d'articles suggérés et s'avérant

véritablement pertinent pour l'utilisateur active. Si l'on considère par exemple, une liste des N meilleures recommandations, la précision correspondra à la proportion d'articles véritablement consultés et appréciés par l'utilisateur courant.

$$précision = \frac{\left|\{\text{Re}\, commandation\ \ pertinentes\} \cap \{\text{Re}\, commandation\ émises\}\right|}{\left|\{\text{Re}\, commandation\ émises\}\right|}$$

7.2 Rappel :

Le rappel est la proportion des articles pertinents par rapport à ceux qui sont pertinents au sein du corpus. Il compte le nombre d'articles dans la liste de recommandations N-top ayant déjà été notés par l'utilisateur courant.

$$Rappel = \frac{\left|\{\text{Re}\, commandation\ \ pertinentes\} \cap \{\text{Re}\, commandation\ émises\}\right|}{\left|\{\text{Re}\, commandation\ \ pertinentes\}\right|}$$

7.3 F-Mesure : C'est la mesure qui combine les deux précédentes:

$$F = 2.\frac{précision.rappel}{(précision + rappel)}$$

7.4 Précision moyenne *(average precision) :*

Cette mesure, notée AvgP, est la moyenne des valeurs de la précision, calculées à chaque rang, où un document pertinent est trouvé. Elle est déterminée de la manière suivante :

$$AvgP = \sum_{i=1}^{n} prec@k.rel(k)$$

Avec *rel(k)* valant 1 si le document au rang *k* est pertinent.

8. Conclusion :

L'apprentissage d'une fonction d'ordonnancement est un type de problème d'apprentissage automatique supervisé ou semi-supervisé dans lequel le but est de construire automatiquement un modèle d'ordonnancement à partir des données d'apprentissage.

Beaucoup de méthodes d'apprentissage d'une fonction d'ordonnancement ont été proposées dans la littérature, avec des motivations et des formulations différentes. Ces méthodes apprennent en général leurs fonctions d'ordonnancement en minimisant certaines fonctions d'erreurs.

Nous avons présenté dans ce chapitre, entre autres, l'état de l'art sur les méthodes d'apprentissages d'ordonnancement pour la recherche et le filtrage d'information.

Chapitre V

Proposition et évaluation

1. Introduction :

Un système de recommandation doit être en mesure de proposer les articles qui sont susceptibles d'être préférés par l'utilisateur. Dans la plupart des systèmes, le degré de préférence est représenté par un score de prédiction. Soit une base de données d'évaluations, attribuées par des utilisateurs sur un ensemble d'articles. Traditionnellement, les algorithmes du filtrage collaboratif sont basés sur la prédiction de notes que le système saurait attribués aux articles non notés, afin qu'ils puissent être ordonnés par des notes prévues. Ainsi une liste d'articles à recommander est produite à l'utilisateur.

Dans cette recherche, nous nous sommes intéressés à améliorer la qualité de la recommandation en nous basant sur la prédiction d'ordre, selon le goût des utilisateurs. Pour se faire, nous avons appliqué les méthodes d'apprentissage d'ordonnancement au filtrage collaboratif, en proposant l'adaptation d'une méthode d'ordonnancement de recherche d'information pour la tâche de prédiction d'ordre au filtrage collaboratif. Ce qui consiste à ordonner correctement les articles plutôt que de prédire correctement leurs notes.

Cette proposition combine la méthode de factorisation matricielle (MF) (consiste à représenter les utilisateurs et les articles par des facteurs latents) avec une des meilleures méthodes d'ordonnancement de l'approche listewise qui présente de bonnes performances par rapport aux autres (pointwise et pairwise). Une liste ordonnée d'articles est obtenue en réduisant au minimum une fonction d'erreur qui représente l'incertitude entre les listes d'apprentissage en entrée et en sortie.

2. Différents cadres de filtrage collaboratif :

Les différents cadres développés pour le filtrage collaboratif comprennent des approches pures ou hybrides qui peuvent être séquentielles ou non-séquentielles **[Jean 08]**.

Le terme pur indique que la recommandation se fonde uniquement sur les jugements des utilisateurs. Le contenu des articles n'est pas utilisé, ce qui contraste avec les premiers systèmes de filtrage d'information décrits dans les chapitres précédents. La seule information disponible sur un article est donc son identifiant. Pour notre exemple, c'est simplement le nom du film. Ce cadre a été largement étudié notamment dans [**Marlin 04**].

Des approches utilisant en même temps les jugements utilisateurs et des descripteurs des articles sont appelées approches de filtrages collaboratifs hybrides car elles mélangent les filtrages collaboratifs et ceux basés sur le contenu. Elles ont aussi été étudiées et peuvent améliorer les prédictions dans des applications [**Basilico et al 04**].

L'aspect non séquentiel du filtrage collaboratif signifie que le système ne prend pas en compte l'ordre dans lequel les notes sont saisies. L'aspect chronologique est ignoré et n'influence pas le processus de recommandation. Pour notre exemple, cela signifie que toutes les notes de la même valeur ont la même signification. Ceci semble naturel mais en réalité, les notes ne sont pas statiques, elles peuvent évoluer et même devenir obsolètes. Un film noté plus récemment aura plus de sens qu'un film noté il y a quelques années. Pour certaines applications, cet aspect chronologique est important et doit être pris en compte. C'est le cas par exemple d'un système de recommandation basé sur l'historique des pages visitées durant la navigation d'un utilisateur [**Shani et al 05**].

Dans ce mémoire, nous nous plaçons dans un cadre de filtrage collaboratif pur, non–séquentiel, où les jugements sont des notes. Nous ne considérons pas l'aspect séquentiel qui rajouterait la difficulté supplémentaire du traitement de l'ordre chronologique des jugements. Nous considérons le cas où les jugements sont exprimés sous forme de notes. Cela correspond par exemple au cas où le système de filtrage collaboratif demande à chaque utilisateur de fournir des jugements sur des articles de son choix, exprimés sous la forme de note (réel positif borné). C'est le cas de la majorité des systèmes, par exemple pour la recommandation de films, où les utilisateurs peuvent donner des notes de 1 à 5, stockées en machine sous forme de matrice de notes. Ce type de jugements présente le double avantage d'être facile à collecter et à traiter du point de vue du système.

3. Méthodes utilisées :

3.1 ListMLE :

C'est l'une des méthodes de l'approche listwise, qui emploie le modèle de Plackett-Luce paramétré et l'estimateur de maximum de vraisemblance et formalise l'apprentissage d'ordonnancement comme problème de minimisation de la fonction d'erreur. Cette dernière est définie de la manière suivante :

$$L(f(X),Y) = -\sum_{i=1}^{n} \log \prod_{j=1}^{m} \frac{\exp(f(X_{Y(j)})}{\sum_{k=j}^{n} \exp(f(X_{Y(k)})}$$

3.1.1 Modèle de Plackett-Luce (PL) :

Le modèle PL définit une distribution de probabilité sur des permutations d'objets, référés comme probabilité de permutation. Supposons qu'il y a un ensemble d'objets $O = \{O_1, O_2, \cdots, O_n\}$. Soit π une permutation (Liste ordonnée) des objets et $\pi^{-1(j)}$ dénote l'objet dans la $j^{ième}$ position dans la permutation π. Supposant qu'il n'existe pas des scores non

négatifs attribués aux objets. Soit S = {s_1, s_2, · · · ,s_n} les scores des objets. Le modèle PL définit la probabilité de permutation π basée sur les scores S comme suit :

$$P_S(\pi) = \prod_{i=1}^{n} \frac{S_{\pi^{-1}(i)}}{\sum_{j=i}^{n} S_{\pi^{-1}(j)}}$$

Les probabilités des permutations forment naturellement une distribution de probabilité.

Exemple :

Supposons qu'il y a trois objets A, B, C dont les scores sont s_A, s_B, s_C avec ($s_A > s_B > s_C$), la probabilité de la permutation ABC est :

$$P_S(ABC) = (\frac{S_A}{S_A + S_B + S_C})(\frac{S_B}{S_B + S_C})(\frac{S_C}{S_C})$$

Et celle de la permutation CBA est :

$$P_S(CBA) = (\frac{S_C}{S_A + S_B + S_C})(\frac{S_B}{S_A + S_B})(\frac{S_A}{S_A})$$

Le modèle de PL possède quelques propriétés. Parmi elles, la probabilité de permutation dans l'ordre décroissant est plus grande par contre celle de l'ordre croissant est plus petite.

3.1.2 Fonction d'erreur de vraisemblance (Likelihood Loss) :

La fonction d'erreur de vraisemblance est définit de la manière suivante :

$$R(f(X), Y) = -\log P(Y \mid X; f)$$

Où

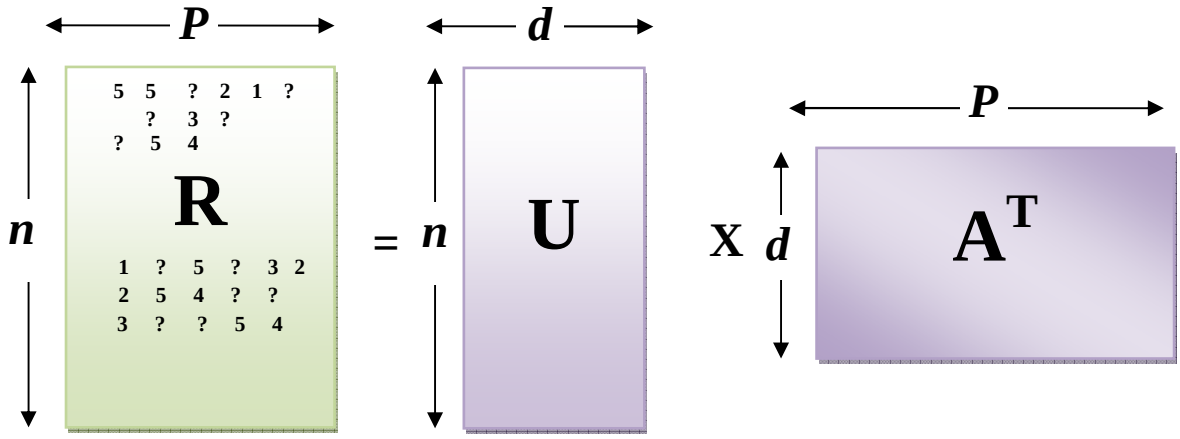
$$P(Y \mid X; f) = \prod_{i=1}^{n} \frac{\exp(f(X_{Y(i)})}{\sum_{k=i}^{n} \exp(f(X_{Y(k)})}$$

Il est à noter la définition réelle de la distribution de probabilité exponentielle paramétrée sur toutes les permutations obtenues par la fonction d'ordonnancement, ainsi que la fonction d'erreur comme étant le négatif log-vraisemblance de la liste obtenue par les scores réels (ordre souhaité). La distribution de probabilité s'avère être un modèle de Plackett-Luce [**Marden 95**].

3.2 Technique de factorisation Matricielle

Considérons *R* la matrice de notes avec des données manquantes qui contient *n* utilisateurs et *p* articles, chaque utilisateur ayant noté au moins un article. Un exemple d'apprentissage est un triplet *(u, a, r)* où *u* est un indice d'utilisateur dans $U = \{1, \ldots, n\}$, *a* est un indice d'article $A = \{1, \ldots, p\}$ et *r* est une note dans $V = \{1, \ldots, v\}$. Nous proposons d'apprendre pour chaque utilisateur une fonction score qui permet de représenter les préférences de manière simple et intuitive. Une fonction d'utilité $f : U \times A \rightarrow R$ modélise la préférence d'un utilisateur pour un article par un score réel. Si un utilisateur $u \in U$ préfère un article *a* à un article *b*, cette préférence est simplement représentée par l'inégalité $f(u,a) > f(u,b)$. Dans le contexte des systèmes de recommandation, les articles sont présentés à chaque utilisateur par ordre décroissant des scores $f(u,.)$.

La méthode de factorisation matricielle (FM) **[Koren 09]** permet d'apprendre la représentation vectorielle des données en utilisant une factorisation matricielle régularisée. Le principe est de trouver deux matrices $A \in R^{pxd}$, $U \in R^{nxd}$ telle que le produit matriciel UA^T est la matrice de scores $(n \times p)$ associée à la fonction d'utilité *f*, où $(UA^T)_{ua} = f(u, a)$. Remarquons que le paramètre de taille *d*, commun aux matrices *U* et *A*, définit la dimension des espaces de représentation des utilisateurs et des articles. Chaque ligne de *A* peut être vue comme une représentation vectorielle des articles. Nous noterons A_j le vecteur représentant le $j^{ème}$ article. De même, chaque ligne de *U* caractérise le classifieur linéaire pour un utilisateur donné. Nous noterons u_i le vecteur poids pour le $i^{ième}$ utilisateur.



Les matrices des paramètres *U* et *A* peuvent être trouvées en résolvant le problème d'optimisation suivant:

$$\min_{U,A} \sum_{\{i,j\} \in R} \left(R_{ij} - U_i A_j^T\right) + \lambda \left(\|U\|_F^2 + \|A\|_F^2\right)$$

Où $\|U\|_F^2$ dénote la norme de Frobenius, et λ le paramètre de régularisation afin d'éviter le sur-apprentissage. avec R désignant l'indexation des notes non nulles dans la matrice creuse.

4. Solution proposée :

Notre objectif est d'obtenir un ordonnancement correct de tous les articles en préservant les préférences de chaque utilisateur et de recommander les k-top articles. La méthode que l'on propose, consiste à combiner les deux méthodes citées auparavant (ListMLE et FM), où la fonction d'erreur (proposée) basée sur leurs principe est définie comme suit:

$$L(U,A)=\sum_{i=1}^{n}\left\{-\sum_{j=1}^{p} I_{iZ(j)}\log\frac{\exp(U_i A_{Z(j)}^T)}{\sum_{k=j}^{p} I_{iZ(j)}\exp(U_i A_{Z(j)}^T)}\right\}+\frac{\lambda}{2}(\|U\|_F^2+\|A\|_F^2)$$

$$L(U,A)=\sum_{i=1}^{n}\left\{\sum_{j=1}^{p} I_{iZ(j)}\left[\log\left(\sum_{k=j}^{p} I_{iZ(j)}\exp(U_i A_{Z(j)}^T)\right)-U_i A_{Z(j)}^T\right]\right\}+\frac{\lambda}{2}(\|U\|_F^2+\|A\|_F^2)$$

Où :

Z : représente l'ordonnancement réel des articles donné par l'utilisateur u;

$Z(j)$: représente la position des articles selon la préférence de l'utilisateur ;

$I_{iZ(j)}$: égale 1 si $Y_{iZ(j)}>0$ sinon 0

Utilisateurs \ Articles	A_1	A_2	A_3	A_4	A_5	A_6
u=1	3	?	5	?	2	1
u=2	4	3	2	5	1	2
u=3	?	2	5	?	5	3

u =2 : $A_4 > A_1 > A_2 > A_6 > A_3 > A_5$

Z

J	1	2	3	4	5	6
Indice de l'article Selon les préférences de l'utilisateur u =2	4	1	2	6	3	5

La fonction L n'est pas convexe en U et A simultanément, en revanche elle l'est en chacune des matrices séparément. Le problème d'optimisation associé s'écrit simplement:

$$(U^*,A^*)=\underset{U,A}{\operatorname{argmin}} L(U,A)$$

Pour minimiser la fonction d'erreur *L*, nous optons pour une approche itérative en 2 étapes, où chaque étape consiste à fixer une des deux matrices et minimiser *L* par rapport à l'autre grâce à la méthode du gradient conjugué.

Algorithme

Entrée:

- L'ensemble de préférences de chaque utilisateur

Initialiser:

- Initialiser $U^{(1)}$ et $A^{(1)}$ aléatoirement
- I⟵ 1

Repeat

- $U^{(I+1)} \longleftarrow \arg\min_{U^{(I)}} L(U^{(I)}, A^{(I)})$
- $A^{(I+1)} \longleftarrow \arg\min_{A^{(I)}} L(U^{(I+1)}, A^{(I)})$
- I⟵ I+1

Until convergence de *L (U, A);*

Sortie: *U* et *A*

Nous donnons les formules permettant de calculer les gradients de *L* par rapport à *U* et à *A*, nécessaires à l'optimisation :

$$\frac{\partial L}{\partial U_{iq}} = \sum_{j=1}^{p} I_{iZ(j)} \left[\frac{\left(\sum_{k=j}^{p} I_{iZ(k)} \exp(U_i A_{Z(k)}^T) A_{Z(k)q} \right)}{\sum_{k'=j}^{p} I_{iZ(k')} \exp(U_i A_{Z(k')}^T)} - A_{Z(j)q} \right] + \lambda U_{iq}$$

$$\frac{\partial L(U,A)}{\partial A_{jv}} = \sum_{i=1}^{n} \left\{ \sum_{x=1}^{p} I_{iZ(x)} \left[\frac{\sum_{k=j}^{p} I_{iZ(k)} \exp(U_i A_{Z(k)}^T) \delta_{Z(k)x} U_{iv}}{\sum_{k'=j}^{p} I_{iZ(k')} \exp(U_i A_{Z(k')}^T)} - \delta_{Z(x)j} U_{iv} \right] \right\} + \lambda A_{jv}$$

$$\text{avec} \quad \begin{matrix} \delta_{Z(k)x}=1 \;\; si \;\; Z(k)=x & \qquad & \delta_{Z(x)j}=1 \;\; si \;\; Z(x)=j \\ \delta_{Z(k)x}=0 \;\; si \;\; Z(k)\neq x & & \delta_{Z(x)j}=0 \;\; si \;\; Z(x)\neq j \end{matrix}$$

5. Expérimentation :

5.1 Jeu de données:

Nous utilisons pour l'expérimentation, le jeu de données réelles du système de recommandation de films MovieLens[1]. Ce jeu de données est très populaire et utilisé dans beaucoup d'études du domaine de filtrage collaboratif [**Nguyen 06**]. Le site Web MovieLens

1 http://movielens.umn.edu/

est développé par le groupe de recherche GroupLens à l'université de Minnesota, Etats-Unis. On dispose sur ce site de deux jeux de données d'évaluations de films de tailles différentes.

Nous utilisons dans cette expérimentation le jeu contenant 100 000 évaluations de 1 à 5 étoiles, fournies par 943 utilisateurs sur 1682 films pendant une période allant de Septembre 1997 à Avril 1998. Ainsi, il existe 93,7% de données manquantes (matrice creuse). La distribution de ces évaluations est présentée dans la figure ci-dessous :

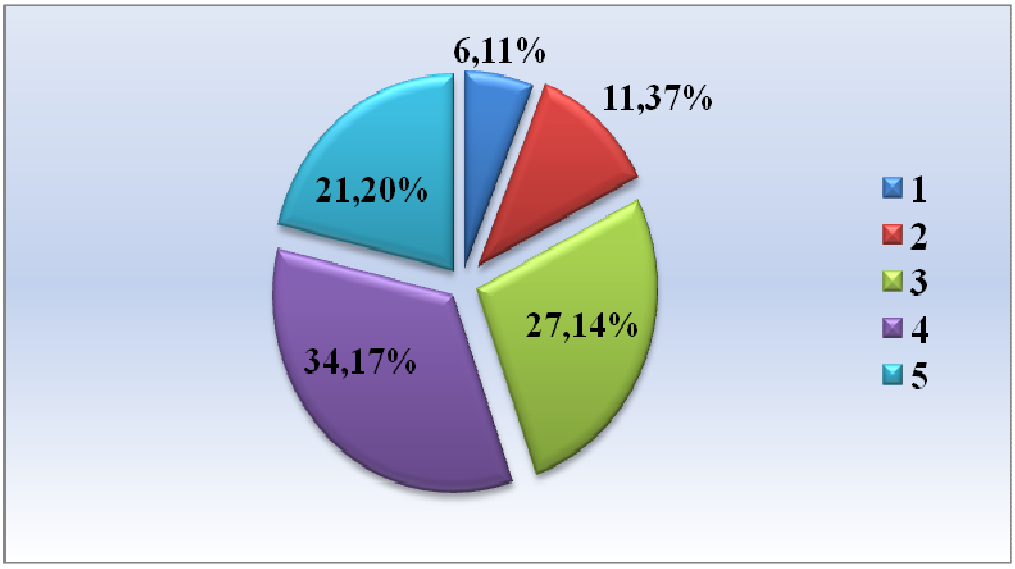


Figure V.1 *Distribution des notes dans la base de données MovieLens*

Nous constatons une grande proportion des évaluations variant entre 3 et 4 étoiles pour un jugement respectif moyen et positif. Par contre une faible proportion pour les évaluations défavorables de 1 et 2 étoiles. Ceci montre la tendance des utilisateurs à n'évaluer que les films qu'ils préfèrent.

5.2 Métrique d'évaluation utilisée :

Le NDCG (Normalized Discounted Cumulative Gain) [**Kalervo et al 00**] est choisi comme métrique d'évaluation, du fait de sa grande sensibilité à la pertinence des k-top films ordonnés. Cette métrique est une version normalisée de la DCG. Elle est calculée en divisant le DCG par le gain cumulatif Idéal IDCG. $$NDCG\ @\ k = \frac{DCG\ @\ k}{IDCG\ @\ k}$$

On peut calculer le score NDCG pour toute permutation d'un ensemble d'articles dont les étiquettes sont connues. NDCG possède un paramètre *k* défini par l'utilisateur et deux fonctions qui se rendent souhaitable dans le cadre d'ordonnancement. Ce paramètre de coupure détermine le nombre d'articles dans la liste d'ordonnancement à considérer.

Le DCG utilise deux fonctions :

– un gain croissant en fonction de la valeur de l'étiquette *D(y),* elle permet de traiter différemment les degrés de pertinence des instances,

– Et un facteur décroissant en fonction du rang *r* de l'instance *g(r)* pour tenir compte des différentes étiquettes.

La mesure est alors définie comme :

$$DCG@k = \sum_{i=1} D(y_i) g(x_i)$$

$$\text{Où } D(y_i) = 2^r - 1 \text{ et } g(x_i) = \frac{1}{Log_2(1+r)}$$

La valeur de NDCG varie de 0 à 1 avec une valeur plus élevée qui indique une meilleure efficacité d'ordonnancement.

5.3 Outil d'évaluation :

Afin d'évaluer notre proposition, nous avons utilisés le langage MATLAB (MATtrix LABoratory) développé par la société MathWorks. MATLAB est un outil conçu dans le but de fournir un environnement de calcul matriciel efficace, interactif et portable. Avec ses fonctions spécialisées, MATLAB peut être aussi considéré comme un langage de programmation adapté pour les problèmes scientifiques et un outil très efficace qui est largement utilisé pour le calcul numérique et la visualisation graphique.

5.4 Résultats et analyse :

Pour notre expérimentation, trois cas se présentent. Pour chaque utilisateur, nous choisissons aléatoirement 10, 20 et 50 films évalués dans le but de l'apprentissage et nous utilisons le reste des films notés dans le profil de l'utilisateur pour le test. Pour chaque cas, nous supprimons les utilisateurs qui ont noté moins de 20, 30 et 60 films respectivement afin de nous assurer que nous pouvons évaluer au moins 10 films notés par utilisateur. Le tableau suivant résume le nombre des évaluations utilisées dans la base d'apprentissage pour les trois cas :

	Cas 1	**Cas 2**	**Cas 3**
Nb user X Nb d'article	943 X 10	744 X 20	497 X 50

Tableau V.1 *Nombre des évaluations utilisé dans la base d'apprentissage*

L'exécution de notre algorithme a été reproduite une dizaine de fois, où une moyenne a été considérée pour chaque cas.

Nous avons adopté le protocole d'évaluation appelé «généralisation faible» dans l'estimation de CoFiRank. Ce protocole permet de mesurer la performance et la qualité de prédiction pour des utilisateurs déjà présents dans la base d'apprentissage.

Afin de mettre en relief notre proposition, nous procédons à une comparaison avec l'une des méthodes de la prédiction d'ordre (CoFiRank-NDCG), ainsi que celle de prédiction de notes (Factorisation Matricielle). Dans les trois cas de l'expérimentation, nous avons choisis k= 10 (NDCG@10) afin de permettre une comparaison direct des résultats et d =5 (taille du paramètre commun aux matrices U et A) du fait des résultats probants obtenus.

Le coefficient de régularisation λ influe sur la convergence de la fonction d'erreur ainsi que sur le contrôle de sur-apprentissage. La figure V.2 illustre la relation entre λ et l'erreur. Aussi elle montre le sur-apprentissage lorsque λ est inférieur ou égale à 0,01. Notons qu'aucun effet de sur-apprentissage avec λ égal ou supérieur à 0,1.

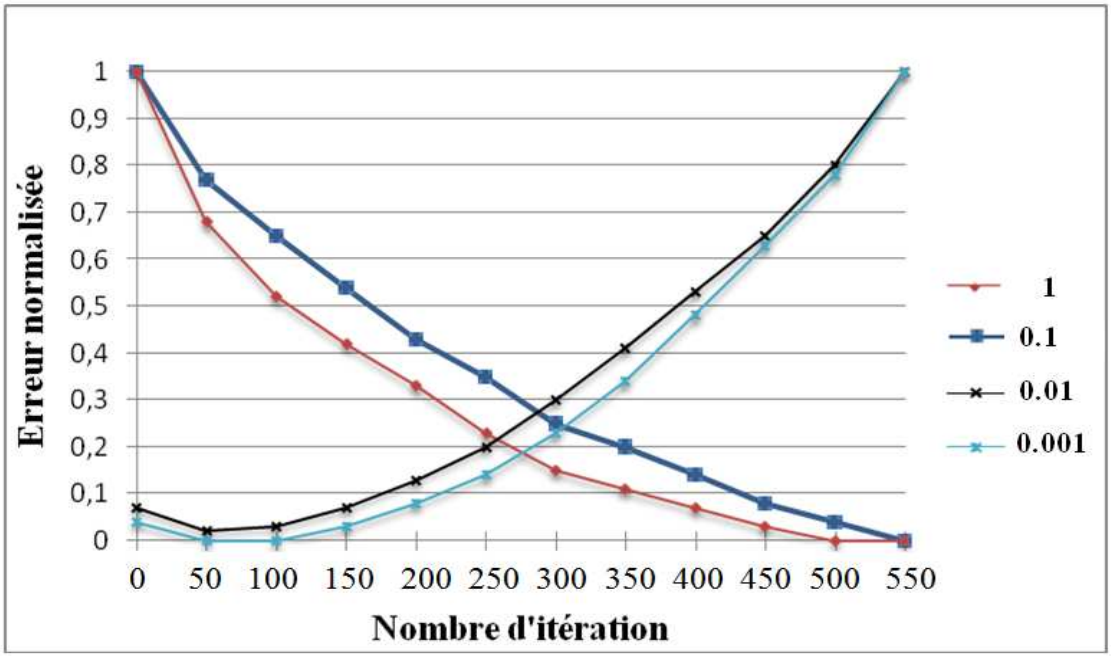


Figure V.2 *L'impact du coefficient de régularisation sur la convergence de l'erreur dans le processus d'apprentissage*

Les résultats expérimentaux sont consignés dans le tableau suivant :

	Cas d'apprentissage		
	10	**20**	**50**
CoFiRank-NDCG	**0.6400 ±0.0061**	0.6307 ±0.0062	0.6076 ±0.0077
Proposition	0.6381 ±0.0073	**0.6340 ±0.0055**	**0.6210 ±0.0030**
FM	0.6025±0.0053	0.6210 +0.0024	0.6165 ±0.0015

Tableau V.2 : *Comparaison entre CoFiRank-NDCG, FM et proposition par la métrique NDCG*

L'examen du tableau ci-dessus, montre :

- Un rapprochement de nos résultats avec ceux de CoFiRank-NDCG où l'on remarque une intersection des résultats dans les intervalles respectifs [0.6454, 0.6339], [0.6369, 0.6245] pour les deux premiers cas et une amélioration pour le troisième.

- Une amélioration des résultats de notre proposition par rapport à la méthode de factorisation matricielle.

L'optimisation de notre fonction d'erreur a amené à la minimiser. Cette minimisation d'erreur conduirait à un bon établissement d'ordonnancement. La figure V.3 ci-après montre que la relation entre le NDCG@10 et l'erreur est inversement proportionnelle au cours des itérations de l'optimisation. On remarque, que l'efficacité de la performance d'ordonnancement devienne à la fois optimale et convergente lors de l'optimisation de la fonction d'erreur. En outre, il indique que notre proposition est efficace lorsque le NDCG@10 devient optimal après 180 itérations.

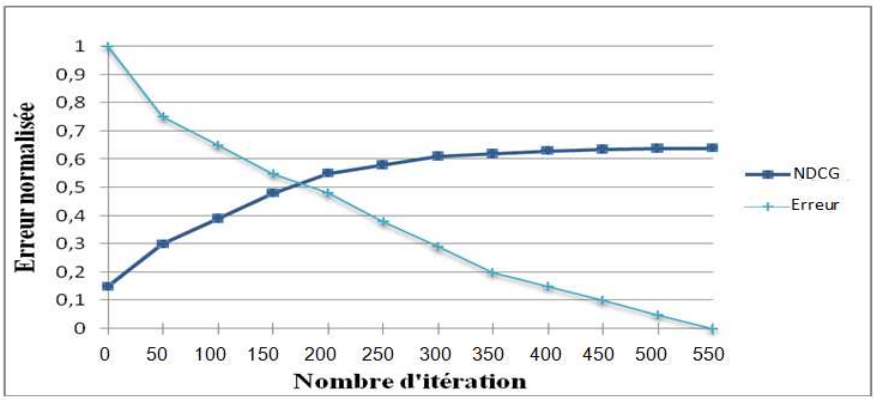


Figure V.3 *L'efficacité de la proposition pour atteindre NDCG@10 optimale en minimisant l'erreur*

6. Conclusion :

Il ressort qu'à travers cette recherche, l'ordonnancement collaboratif reste l'une des meilleures façons pour faire des recommandations. Inspiré par les travaux récents dans la communauté Learning to rank, notre proposition utilise une des méthodes de l'approche listwise qui donne des résultats expérimentaux satisfaisants. Dans notre cas cette méthode est utilisée comme une technique pour ordonner les articles pour chaque utilisateur, dans laquelle les utilisateurs et les articles sont représentés comme des caractéristiques latentes utilisant la factorisation matricielle (MF). La contribution de notre proposition fournit une amélioration des performances par rapport à la factorisation matricielle, ainsi qu'elle présente un rapprochement des résultats voire une amélioration par rapport à CoFiRank-NDCG.

Conclusion et perspectives :

Les systèmes de recommandation ont été conçus comme des outils puissants pour aider les utilisateurs à trouver et évaluer les articles d'intérêt. Ces systèmes utilisent une variété d'algorithmes pour soutenir les utilisateurs à identifier les articles qui correspondent le mieux à leurs goûts ou à leurs besoins. La plupart de ces algorithmes sont basés sur la prédiction de notes, qui consiste essentiellement à prédire les notes le plus précisément possible.

Ces dernières années les systèmes de recommandation ont connu des progrès significatifs et de nombreuses techniques ont été proposées pour améliorer la qualité de recommandation. Parmi ces techniques la prédiction d'ordre qui consiste à ordonner correctement les articles selon le goût des utilisateurs. Généralement, ces systèmes ne s'intéressent qu'aux premiers articles de la liste, afin de faire des recommandations personnalisées, qui consistent à résoudre le problème d'ordonnancement relatif à chaque utilisateur. Tout en signalant que cette liste est le résultat direct d'un algorithme d'ordonnancement, ou peut être calculé à partir des résultats d'un algorithme de prédiction de note par le tri des articles en fonction de leur note prédit. Nous pensons qu'ordonner correctement les articles est plus important que prédire correctement leurs notes.

Dans ce contexte et dans un premier temps, nous avons présenté un état de l'art regroupant trois chapitres, à savoir : les systèmes de filtrage d'information avec leurs types, les systèmes de filtrage collaboratif en incluant les différents algorithmes de calcul de prédiction de notes, ainsi que les méthodes d'apprentissage d'ordonnancement pour la recherche et le filtrage d'information. Dans notre contribution, nous avons proposé une approche qui exploite l'apprentissage d'une fonction d'ordonnancement dans le filtrage collaboratif, consistant à combiner la méthode d'ordonnancement ListMLE avec la technique de factorisation matricielle. Le principe de notre proposition consiste à minimiser une fonction d'erreur afin d'obtenir une liste d'articles qui préserve le mieux possible la préférence de chaque utilisateur.

L'expérimentation à été menée sur le jeu de données réel MovieLens, pour lequel nous avons utilisé un protocole expérimental permettant d'évaluer les performances en apprentissage hors ligne «généralisation faible» pour évaluer la prédiction d'ordre.

Nous souhaitons pour les futurs travaux, d'étendre l'évaluation de notre proposition en utilisant le protocole expérimental d'apprentissage en ligne «généralisation forte». De plus nous projetons d'utiliser des benchmarks à grande échelle tels que Netflix, EachMovie et de diversifier la comparaison de notre proposition avec d'autres méthodes qui sont basées sur la prédiction d'ordre.

Nous signalons que la présente proposition, comme la plupart des algorithmes de recommandation de filtrage collaboratif, pourrait être considérée comme une approche de recommandation variationelle, où les mesures d'évaluation, telles que, la précision moyenne MAP et NDCG, ne sont pas directement liées à notre modèle. Par contre les récentes recherches dans le domaine d'apprentissage d'une fonction d'ordonnancement pourraient être plus exploitable pour améliorer la performance de recommandation par optimisation directe des mesures d'évaluation.

Bibliographie:

[Adomavicius et al 05] Adomavicius G. and Tuzhilin A. *«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, 2005.

[Amini 07] Amini Massih-Reza (2007) *«Apprentissage de Fonctions de Classification et d'Ordonnancement avec des Données Partiellement Étiquetées»* Habilitation à diriger des recherches, Université Pierre Et Marie Curie - Paris 6

[Basilico et al 04] Basilico, J. and Hofmann, T. (2004). *«A joint framework for collaborative and content filtering»*. In Sanderson, M., Järvelin, K., Allan, J., and Bruza, P., editors, SIGIR, pages 550–551. ACM.

[Belkin 92] Belkin N.J., Croft W.B.*«Information filtering and information retrieval: two sides of the same coin»* Communications of the ACM, vol. 35, n°12, p. 29-38, 1992.

[Bell et al 07] Bell R. M. and Koren Y. *«Improved neighbourhood based collaborative filtering. In Proceedings of the KDDCup»* 2007.

[Berrut 03] Berrut C., Denos N. «*Filtrage collaboratif. In Assistance intelligente à la recherche d'informations*» p. 241-269, Hermes - Lavoisier, 2003.

[Berry 04] Berry M., Shahnaz F., Pauca P., Plemmons R., *« Document Clustering using Nonnegative Matrix Factorization »*, IPM Journal, 2004.

[Breese 98] Breese J. S., Heckerman D., and Kadie C. *«Empirical Analysis of Predictive Algorithms for Collaborative Filtering»*. pages 43–52. Morgan Kaufmann, 1998.

[Burges et al 08] Burges C., Li P. and Wu Q. *«Mcrank: Learning to rank using multiple classification and gradient boosting»*. In NIPS '07: Advances in Neural Information Processing Systems 20, pages 897–904, Cambridge, 2008

[Cao et al 07] Cao Z., Qin, T., Liu, T.-Y., Tsai, M.-F. and Li, H., 2007. « Learning to rank: From pairwise approach to listwise approach» Technical Report, MSR-TR-2007-40, Microsoft Research.

[Chen 98] Chen L. and Sycara K., *« WebMate: A personal agent for browsing and searching »*, Proceedings of the 2nd international conference on autonomous agents and multi agent systems, Minneapolis, 1998, p. 10-13.

[Cho et al 02] Cho Y. H., Kim J. K., and Kim S. H. *«A Personalized Recommender System Based onWeb Usage Mining and Decision Tree Induction»*. Expert Systems with Applications, 23(3) :329 – 342, 2002.

[Clémençon et al 05] Clémençon S., Lugosi G., Vayatis N., (2005) *«Ranking and Scoring Using Empirical Risk Minimization»* , AUER P., MEIR R., Eds., COLT, vol. 3559 de Lecture Notes in Computer Science, Springer, , p. 1-15.

[Cohen et al 97] Cohen, W. W., Schapire, R. E., and Singer, Y. (1997). *«Learning to order things»* In Jordan, M. I., Kearns, M. J., and Solla, S. A., editors, NIPS. The MIT Press.

[Collins 97] Collins, M. (1997). *«The EM algorithm»*. In fulfillment of Written Preliminary Exam II requirement

[Cortes and Mohri 04] Cortes, C. and Mohri, M. (2004). *«AUC optimization vs. error rate minimization»*. In Thrun, S., Saul, L., and Sch¨olkopf, B., editors, Advances in Neural Information Processing Systems *16*. MIT Press, Cambridge, MA.

[Cossock et al 08] Cossock D. and Zhang T. (2008) *«Statistical analysis of bayes optimal subset ranking»* Information Theory, 54:5140–5154.

[Crammer et al 02] Crammer, K., Singer, Y (2002) «*PRanking with ranking»*. Advances in Neural Information Processing Systems, 14: 641-647.

[Croft 93] Croft W.B., *« Knowledge-based and Statistical approaches to Text Retrieval »*, IEEE EXPERT, vol. 8, n° 2, p. 8-12, avril 1993.

[Damien 11] Damien Poirier, (février 2011) *« Des Textes Communautaires à La Recommandation »*, Thèse doctorat, université D'orléans.

[Dammak 09] Faïza Dammak, Hager Kammoun, (2009) *«Apprentissage de Fonctions d'Ordonnancement d'Alternatives avec Approche Actif»*.

[Das et al 07] Das, A., Datar, M., Garg, A., and Rajaram, S. (2007). «Google news personalization : scalable online collaborative filtering». InWilliamson, C. L., Zurko, M. E., Patel-Schneider, P. F., and Shenoy, P. J., editors, WWW, pages 271–280. ACM.

[DeCoste 06] DeCoste D. (2006) *«Collaborative Prediction Using Ensembles of Maximum Margin Matrix Factorizations »*, International Conference on Machine Learning.

[Dempster et al 77] Dempster A. P., Laird N. M., et Rubin D. B. *«Maximum likelihood from incomplete data via the em algorithm»*. Journal of the Royal Statistical Society. Series B (Methodological), 39(1) :1–38, 1977.

[Devarajan 08] Devarajan, K. (2008). *«Nonnegative matrix factorization : An analytical and interpretive tool in computational biology»* PLoS Computational Biology, 4(7) :e1000029+.

[Dhillon et Sra 06] Inderjit S. Dhillon et Suvrit Sra. *«Generalized nonnegative matrix approximations with bregman divergences»* NIPS, 2006.

[**Esslimani 10**] Esslimani Ilham, (décembre 2010) *«Vers une approche comportementale de recommandation : apport de l'analyse des usages dans un processus de personnalisation»* Thèse de doctorat, université Nancy 2.

[François 09] François Maillet, (décembre 2009) *«Algorithmes de recommandation musicale»*

[Freund et al 03] Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. (2003) *«An efficient boosting algorithm for combining preferences»* Journal of Machine Learning Research, 4: 933-969.

[Gallardo 05] Gallardo Lopez L., *«Accès à l'information par un système de filtrage collaboratif contrôlé»*, Rapport de thèse de doctorat, université de Joseph Fourier, 2005.

[Goldberg et al 92] Goldberg D., Nichols D., Oki B. M., and Terry D. *«Using Collaborative Filtering toWeave an Information Tapestry»*. Commun. ACM, 35(12) :61–70, 1992.

[Goldberg et al 01] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). *«Eigentaste : A constant time collaborative filtering algorithm»*. Information Retrieval, 4(2): 133–151.

[Herbrich et al 00] Herbrich, R., Graepel, T., and Obermayer, K. (2000). *«LargeMargin Ranking Boundaries for Ordinal Regression»*. In Smola, A., Bartlett, P., B., S., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 115–132.

[Herlocker et al 99] Herlocker J. L., Konstan J. A., Borchers A., and Riedl J. (1999). *«An algorithmic framework for performing collaborative filtering»* In SIGIR, pages 230–237.

[Herlocker et al 04] Herlocker J. L., Konstan J. A., Terveen L. G. and Riedl J. T. (January 2004), *«Evaluating Collaborative Filtering Recommender Systems»* Inf. Syst., 22 :5–53,

[Hofmann 04] Hofmann, T. (2004). *«Latent semantic models for collaborative filtering»* ACM Transactions on Information Systems, 22(1) : 89 – 115.

[Huyen 08] Huyen-Trang VU (octobre 2008) *«Apprentissage d'ordonnancements pour la constitution de corpus d'évaluation et pour l'agrégation de listes en recherche d'information»* ;

[Jannach et al 11] Jannach D., Zanker M., Felfernig A., and Friedrich G. (2011) *«Recommender Systems An Introduction»* Cambridge University Press.

[Jean et al 06] Jean-François Pessiot, Vinh Truong, Nicolas Usunier, Massih-Reza Amini et Patrick Gallinari, *«Factorisation en matrices non négatives pour le filtrage collaboratif»* CORIA 2006.

[Jean el al 07] Jean-François Pessiot, Vinh Truong, Nicolas Usunier, Massih-Reza Amini, Patrick Gallinari, *«Filtrage Collaboratif avec un Algorithme d'Ordonnancement»*, CORIA 2007, pp. 165–180, Saint- ´ Etienne, 28–30 mars 2007.

[Jean 08] Jean-François Pessiot, (2008) *«Apprentissage automatique pour l'extraction de caractéristiques : Application au partitionnement de documents, au résumé automatique et au filtrage collaboratif»*, Thèse doctorat. Université Pierre et Marie Curie

[Kalervo et al 00] Kalervo Järvelin et Jaana Kekäläinen. (2000) *« Ir evaluation methods for retrieving highly relevant documents»*. In SIGIR '00 : Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, pages 41–48, New York, NY, USA, ACM.

[Kien 06] Kien D. N., *«Moteurs de composition pour le système d'information sémantique et adaptatif, mémoire de fin d'études master en informatique, Institut de la francophonie pour l'informatique»*, 13 septembre 2006.

[Koren 09] Koren, Y., Bell, R., and Volinsky, C., (2009) *«Matrix factorization techniques for recommender systems»*. IEEE Computer, 42, 8, 30-37.

[Lee et Seung 99] Lee D. D. et Seung H. S. *«Learning the parts of objects by non-negative matrix factorization»*. Nature, 401(6755) :788–791, October 1999.

[Lee and Seung 00] Lee, D. D. and Seung, H. S. (2000). *«Algorithms for nonnegative matrix factorization»*. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, NIPS, pages 556–562.

[Léa 10] Léa Laporte, Nomao *«RI géolocalisée et contextualisation»*

[Linden 03] G. Linden, B. Smith, and J. York. *«Amazon.com Recommendations : Item-to-Item Collaborative Filtering»*. IEEE Internet Computing, 7 :76–80, January 2003.

[Liu et al 08] Liu N., and Yang, Q., (2008) *«EigenRank: a ranking-oriented approach to collaborative filtering»*. In SIGIR ′08, 83-90.

[Liu et al 09] Liu N., Zhao, M., and Yang, Q., (2009). *«Probabilistic latent preference analysis for collaborative filtering»*. In CIKM ′09, 759-766.

[Luce 59] Luce, R.D. *«Individual Choice Behavior»*. Wiley, New York (1959)

[Lui 09] Tie-Yan Liu, *« Learning to rank for Information Retrieval »*, Foundations and Trends in Information Retrieval, vol. 3, n°3, 2009, p. 225-331.

[MacQueen 67] MacQueen J. B., *«Some Methods for classification and Analysis of Multivariate Observations»*, Proc. of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1967, 1:281-297.

[Mallows 75] Mallows, C.L. *«Non-null ranking models»*. Biometrika **44**, 114–130 (1975)

[Maltz 95] MALTZ D., EHRLICH K., *« Pointing the way: active collaborative filtering »*, Proceedings of CHI'95, p. 7-11, mai 1995.

[Marden 95] Marden, J.,(1995) *«Analyzing and Modeling Rank Data»*. CRC Press,

[Marlin 03] Marlin, B. (2003). *«Modeling user rating profiles for collaborative filtering»*. In Thrun, S., Saul, L. K., and Schölkopf, B., editors, NIPS. MIT Press.

[Marlin 04a] Marlin, B. (2004). *«Collaborative filtering : A machine learning perspective»*

[Marlin 04b] Marlin B., *«Modeling User Rating Profiles for Collaborative Filtering»*, Neural Information Processing Systems, 2004b.

[Montaner et al. 03] Montaner M., López B., De La Rosa J. L. *«A Taxonomy of Recommender Agents on the Internet»*. Artificial Intelligence Review, vol. 19, p. 285-330, Kluwer Publishers, 2003.

[Nakamura 98] Nakamura A., Abe N., (*1998) «Collaborative Filteringusing weighted majority prediction algorithms», Proceedings of ICML98, pages395-403.*

[Nguyen 06] Nguyen A. T., (2006) *«COCoFil2 : Un nouveau système de filtrage collaboratif basé sur le modèle des espaces de communautés»*, Thèse doctorat, université Joseph Fourier -Grenoble I.

[Nguyen 09] Nguyen Tuong Vinh Truong, (2009) *«Apprentissage de Fonctions d'Ordonnancement avec peu d'Exemples Étiquetés: une Application au Routage d'Information, au Résumé de Textes et au Filtrage Collaboratif»*, Thèse doctorat. Université Pierre et Marie Curie

[Nicolas 06] Nicolas Usunier, (2006) *«Apprentissage de fonctions d'ordonnancement : une étude théorique de la réduction à la classification et deux applications à la Recherche d'Information»,* Thèse doctorat. Université Paris 6

[Nouali 97] Nouali O., Benmezùine S., Djaid M., Sidi-Boumediène S., (1997) *«Filtrage de l'Information»* Laboratoire Intelligence Artificielle, CE.R.I.S.T

[Pauca et al 04] Pauca, Shahnaz, Berry, and Plemmons (2004). *«Text mining using non-negative matrix factorizations»*. In SDM.

[Plackett 75] Plackett, R.L. (1975) *«The analysis of permutations»*. Applied Statistics 24(2), 193–202.

[Polcicová 04] Polcicová, G. (2004). *«Topographic Organization of User Preference Patterns in Collaborative Filtering»*. PhD thesis, Slovak University of Technology in Brastilava.

[Rennie et al 05] Rennie J. D. M., Srebro N., *« Fast maximum margin matrix factorization for collaborative prediction »*, ICML '05 : Proceedings of the 22nd international conference on Machine learning, ACM Press, New York, NY, USA, p. 713-719, 2005.

[Resnick et al 94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. *«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, New York, NY, USA, 1994. ACM.

[Ruslan et al 08] Salakhutdinov, R., and Mnih, A., 2008. *«Probabilistic matrix factorization»*. In NIPS '08, 20.

[Salakhutdinov et al 07] Salakhutdinov, R., Mnih, A., and Hinton, G. E. (2007). *«Restricted boltzmann machines for collaborative filtering»*. In Ghahramani, Z., editor,ICML, volume 227 of ACM International Conference Proceeding Series, pages 791–798. ACM.

[Sarwar et al 00] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. T. (2000). *«Application of dimensionality reduction in recommender systems-a case study »*. In In ACM WebKDD Workshop.

[Sarwar et al 01] Sarwar B., Karypis G., Konstan J., Riedl J., *«Item-Based Collaborative Filtering Recommendation Algorithms»*, Proc. 10th Int'l WWW Conf., 2001.

[Settles 09] Settles, B. (2009). *«Active learning literature survey»*. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

[Schafer et al 07] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. *«The AdaptiveWeb. Chapter Collaborative Filtering Recommender Systems»*, pages 291–324. Springer-Verlag, Berlin, Heidelberg, 2007.

[Shahnaz et al 06] Shahnaz, F., Berry, M. W., Pauca, V. P., and Plemmons, R. J. (2006). *«Document clustering using nonnegative matrix factorization»*. Information Processing and Management, 42(2) :373–386.

[Shani et al 05] Shani, G., Heckerman, D., and Brafman, R. I. (2005). *«An mdpbased recommender system»*. Journal of Machine Learning Research, 6 :1265–1295.

[Shardanand 95] Shardanand U., Maes P., *«Social Information Filtering: Algorithms for Automating 'Word of Mouth'», Proc. Conf. Human Factors in Computing Systems, 1995*.

[Shi 10] Y. Shi, M. Larson, and A. Hanjalic, (2010) *«List-wise learning to rank with matrix factorization for collaborative filtering»*, In Proceedings of the fourth ACM conference on Recommender systems, pp. 269–272.

[Sieg 07] Sieg A., Mobasher B., Burke R., *«Learning Ontology-Based User Profiles: A Semantic Approach to Personalized Web Search»*, IEEE Intelligent Informatics Bulletin, Nov. 2007, Vol.8 No.1.

[Srebro et al 03] Srebro, N. and Jaakkola, T. (2003). *«Weighted low-rank approximations»*. In Fawcett, T. and Mishra, N., editors, ICML, pages 720–727. AAAI Press.

[Su et al 09] Su X. and Khoshgoftaar T. M. *«A Survey of Collaborative Filtering techniques»*. Adv. in Artif. Intell., page 42, January 2009.

[Tamine et al 08] Tamine.L.L., Calabretto.S. (2008). *«Recherche d'information contextuelle et web»*. In Recherche d'information: état des lieux et perspectives. Chap. 7., pp 201-230. Edition Hermes Science Lavoisier.

[Tebri 04] Hamid TEBRI, (2004) *«Formalisation et spécification d'un système de filtrage incrémental d'information»*.

[Thomas 11] Thomas Piton, (Octobre 2011) *« Une Méthodologie de Recommandations Produits Fondée sur l'Actionnabilité et l'Intérêt Économique des Clients Application à la Gestion de la Relation Client du groupe VM Matériaux»*, Thèse doctorat, université de nantes

[Tintarev et al 07] Tintarev N.and J. Masthoff. *« A Survey of Explanations in Recommender Systems»*. Data Engineering Workshops, 22nd International Conference on, 0 :801–810, 2007.

[Vapnik 00] Vapnik, N. V. (2000). *«The Nature of Statistical Learning Theory »*,Springer-Verlag.

[Wall et al 03] Michael E.Wall, Andreas Rechtsteiner, et Luis M. Rocha. *«Singular Value Decomposition and Principal Component Analysis»*, chapter 5, pages 91–109. Kluwel, Norwell, MA, Mar 2003.

[Weimer et al 07] Weimer, M, Karatzoglou, A., Le, Q., and Smola, A., (2007) *«CoFirank maximum margin matrix factorization for collaborative ranking»*. In NIPS 07, 20, 1593-1600.

[Xia et al 08] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. *«Listwise approach to learning to rank: theory and algorithm»*. In ICML '08: Proceedings of the 25th international conference on Machine learning, pages 1192–1199, New York, NY, USA, 2008.

[Xu 03] Xu W., Liu X., Gong Y., « *Document clustering based on non-negative matrix factorization* », SIGIR '03, 2003, p. 267–273.

[Zemirli 05] Zemirli N., Lechani T. L., Boughanem M. *«Accès personnalisé à l'information : proposition d'un profil utilisateur multidimensionnel»*, 7th ISPS Algérie, Mai 2005.