République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou

Faculté de Génie Electrique et Informatique

Département Informatique

MEMOIRE DE FIN D'ETUDES

En vue de l'obtention d'un diplôme de Master en Informatique

Spécialité : Ingénierie des Systèmes d'Information

Thème

Implémentation et évaluation des modèles de recommandation basés sur le machine et deep learning

Proposé et dirigé par :

Mr. HAMMACHE

Réalisé par:

BIRI Lydia

GUEMAT Yasmine

Promotion : 2019 / 2020

Chapitre II : Les systèmes de recommandation

Chapitre III : Implémentation et évaluation

Liste des figures

Liste des tableaux

Remerciements

Nous rendons grâce à Dieu qui nous a donné l'aide, la patience, le courage pour accomplir ce travail et nous a maintenu en santé pour mener à bien cette année d'étude.

Nous tenons à remercier notre promoteur, M. HAMMACHE Arezki, pour nous avoir proposé ce travail, pour l'aide qu'il a fourni et les connaissances qu'il a su nous transmettre. Nous le remercions également pour sa disponibilité et la qualité de ses conseils.

Nos remerciements s'adressent aussi aux Mrs les jurés pour l'intérêt qu'ils ont porté à ce travail en acceptant d'être examinateurs.

Nous tenons à saisir cette occasion et adresser nos profonds remerciements et nos profondes reconnaissances aux responsables et au personnel de l'UMMTO.

Nous désirons aussi remercier les professeurs de l'UMMTO, qui nous ont fourni les outils nécessaires à la réussite de nos études universitaires.

Un grand merci à nos mamans et nos papas, pour leur amour, leurs conseils ainsi que leur soutien inconditionnel à la fois moral et économique, ainsi que nos sœurs et frères et toutes nos familles, qui nous ont permis de réaliser les études que nous voulions et par conséquent ce mémoire.

Nous voudrions exprimer nos reconnaissances envers les amis et collègues qui nous ont apporté leur soutien moral et intellectuel tout au long de notre démarche.

Nous remercions toutes les personnes qui nous ont soutenues de près ou de loin dans la réalisation de ce travail.

Dédicace

Merci Allah de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout du rêve.

Je dédie ce modeste travail à ceux qui m'ont encouragé et soutenu, je cite :

Les parents les plus chers au monde, papa et maman, que dieu les garde et les protège.

A mes sœurs et leurs époux, mon frère et sa femme et ma chère nièce Manel

A mes grands-parents, mes oncles et mes chères tantes ainsi que leurs épouses et époux.

Et toutes mes chères cousines et cousins.

Et surtout mes meilleurs amis, qui m'ont toujours encouragé, et à qui je souhaite plus de succès.

À mon binôme Yasmine

A la promotion de département informatique 2019/2020

Et à toute personne ayant participé à l'élaboration de ce travail

Lydia

Dédicace

Je dédie ce travail à tous ceux qui comptent pour moi,

À mes très chers parents,

Aucun hommage ne saurait exprimer mon amour éternel, ma reconnaissance et ma considération pour les sacrifices que vous avez consentis pour mon éducation et mon bien être.

Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitterai jamais assez.

Puisse Dieu, le Très Haut, vous accorder santé, bonheur et longue vie et faire en sorte que jamais je ne vous déçoive.

À mon cher frère « Nassim », et mes chères sœurs « Nassima » et « Fella »,

En témoignage de mon affection fraternelle et de ma profonde tendresse, je vous souhaite une vie pleine de bonheur et de succès et que Dieu, le tout puissant, vous protège et vous garde.

À mes très chères tantes « Tassadit » et « Fatma »

Qui m'ont accompagné par leurs douceurs et leurs soutiens, puisse Dieu leurs prêter longue vie et beaucoup de santé et de bonheur.

*À mon binôme « **Lydia** »,*

À toutes les personnes qui ont participé à l'élaboration de ce travail.

*À tous mes amis(es) de la promotion **2019/2020,** en particulier «**Youcef**».*

À tous ceux que j'ai omis de citer.

Yasmine

Introduction générale

Les systèmes de recommandation sont au cœur des systèmes d'information modernes que nous utilisons quotidiennement. Ils apportent une solution au problème de surcharge d'informations. Dans l'ensemble, ces systèmes ont joué un rôle vital et indispensable dans divers systèmes d'accès à l'information pour stimuler les affaires et faciliter le processus décisionnel.

L'essor du Web et sa popularité ont notamment contribué à la mise en place de tels systèmes comme dans le domaine du e-commerce, citons par exemple le site Web populaire Amazon, de streaming audio ou vidéo comme Netflix, ou par des réseaux sociaux comme Facebook. Les deux tiers des films loués sur Netflix ont été choisis sur la base de recommandation, 38% des news Google sont générées sur la base de recommandations et 35% des achats sur Amazon.com dépendent aussi des recommandations.

Les systèmes de recommandation utilisent principalement deux méthodes de filtrage pour proposer des recommandations personnalisées aux utilisateurs, à savoir le filtrage collaboratif et le filtrage basé sur le contenu. Le premier regroupe des techniques qui visent à opérer une sélection sur les items à présenter aux utilisateurs (filtrage) en se basant sur le comportement et les goûts exprimés de très nombreux autres utilisateurs. Le deuxième consiste à déterminer quels items coïncident le mieux avec les préférences de l'utilisateur. Le filtrage collaboratif est considéré comme la méthode la plus populaire et la plus répandue dans les systèmes de recommandation.

Les systèmes de recommandation basés sur le filtrage collaboratif recommandent à l'utilisateur actif les items que d'autres utilisateurs ayant des préférences similaires ont aimés par le passé. Cette méthode de recommandation repose sur l'idée que, si les utilisateurs sont d'accord sur la qualité de certains items, ils seront probablement d'accord sur d'autres.

L'application des techniques de l'IA dans les systèmes de recommandation est en plein essor. Récemment, il a radicalement changé l'architecture des recommandations et a apporté plus d'opportunités pour améliorer les performances des systèmes de recommandation. Les progrès récents dans ces systèmes basés sur l'apprentissage profond ont retenu l'attention en surmontant les obstacles des modèles conventionnels et en obtenant une qualité de recommandation élevée.

L'apprentissage profond (en anglais deep learning) est un type d'intelligence artificielle dérivé de l'apprentissage automatique, il peut généralement être considéré comme un sous-domaine de ce dernier. C'est un ensemble de méthodes d'apprentissage automatique

qui a permis des avancées importantes en intelligence artificielle dans les dernières années, tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires.

L'apprentissage profond s'appuie sur un réseau de neurones artificiels s'inspirant du cerveau humain. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit.

Notre travail s'inscrit dans le cadre de l'application des techniques de l'apprentissage automatique et profond dans les systèmes de recommandations. Précisément, l'objectif de notre travail est d'implémenter, d'évaluer et de comparer deux méthodes de l'état de l'art de machine learning, k voisins les plus proches (KNN) et le classement personnalisé bayésien (BPR), avec un modèle basé sur le deep learning, filtrage collaboratif neuronal (NCF), dans le cadre des systèmes de recommandation.

Pour atteindre cet objectif, nous avons organisé notre mémoire en trois chapitres :

Dans le premier chapitre nous allons faire un état de l'art qui donne une vision générale sur l'apprentissage automatique et profond.

Le deuxième chapitre est consacré aux systèmes de recommandation, en expliquant l'ensemble des classifications en présentant leurs objectifs et les différents types de recommandation, avec quelques travaux de l'état de l'art basé sur le machine et deep learning.

Nous exposons les résultats de nos expérimentations dans le troisième chapitre, en présentant les différents outils utilisés pour la réalisation, Et enfin une conclusion générale qui résume notre travail.


Chapitre I

I.1 Introduction

L’intelligence artificielle (IA) est la discipline scientifique qui consiste à créer des programmes informatiques qui effectuent des opérations comparables à celles de l’esprit humain, telles que l’apprentissage ou le raisonnement logique. Le machine learning va plus loin, c’est cette branche de l’IA qui permet aux machines d’apprendre par elles-mêmes, sans dépendre de commandes. Il fait référence au développement, l'analyse et l'implémentation des méthodes qui permettent à une machine d'évoluer et de remplir des tâches impossibles pour un être humain grâce à un processus d’apprentissage. L’apprentissage profond (en anglais deep learning) est un type de machine learning, mais plus complexe. C’est un ensemble de méthodes d’apprentissage automatique tentant de modéliser avec un haut niveau d’abstraction des données grâce à des architectures articulées de différentes transformations non linéaires.

Dans ce chapitre nous allons présenter d’abord les notions de base de l’apprentissage automatique, définir l’apprentissage profond. Ensuite nous allons présenter les réseaux de neurones et ses principales architectures. Nous terminons ce chapitre par la présentation de quelques plateformes de développement dédiées au deep learning.

I.2 Notions de base

I.2.1 Apprentissage automatique

L’apprentissage automatique concerne le développement d’algorithmes permettant de rendre une machine capable d’accomplir des tâches complexes sans avoir été explicitement programmée dans ce but. Son objectif est d’extraire et exploiter automatiquement l'information présente dans un jeu de données [1].

La plupart des méthodes d’apprentissage automatique fonctionnent bien en raison de représentations et de fonctions de saisie conçus par l’homme. Il devient simplement une optimisation des poids pour mieux faire une prédiction finale.

I.2.1.1 Types d'apprentissage

L’apprentissage automatique est subdivisé selon la nature des données en entrée puis selon la nature des données en sortie.

A. L’apprentissage automatique selon la nature des données en entrée

- **Apprentissage supervisé**

C’est une forme d’apprentissage automatique qui crée des modèles d’intelligence artificielle en se fondant sur des données d’apprentissage « étiquetées ».

- **Apprentissage non supervisé**

C’est une technique d’apprentissage automatique qui cherche une structure des données non étiquetées.

- **Apprentissage par renforcement**

L’algorithme apprend un comportement étant donné une observation. L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d’apprentissage.

B. L’apprentissage automatique selon le type de résultat

1. La classification

La classification est un type d’apprentissage supervisé, qui identifié la catégorie à laquelle un nouveau élément appartient sur la base d’un data set d’entrainement de données contenant des observations dont la catégorie est connue [2].

Il existe plusieurs types d'algorithmes de classification :

- **Arbre de décision :** classe l'utilisation d'une structure d'arbre avec des règles if-then, exécutant l'entrée via une série de décisions jusqu'à ce qu'elle atteigne une condition de terminaison. Capable de modéliser des processus de décision complexes et très intuitif, mais peut facilement surdimensionné les données.
- **Forêt aléatoire :** un ensemble d'arbres de décision, avec sélection automatique de l'arbre le plus performant. Fournit la force de l'algorithme d'arbre de décision sans problème de surapprentissage.
- **Naïve Bayes classifier :** un classificateur basé sur les probabilités. Calcule la probabilité que chaque point de données existe dans chacune des catégories cibles. Simple à mettre en œuvre et précis pour un large éventail de problèmes, mais sensible à l'ensemble des catégories sélectionnées.
- **K-voisin les plus proche :** classe chaque point de données en analysant ses voisins les plus proches parmi les exemples d'apprentissage. Simple à mettre en œuvre et à

comprendre, efficace pour de nombreux problèmes, en particulier ceux à faible dimensionnalité. Fournit une précision inférieure par rapport aux algorithmes supervisés, et nécessite beaucoup de calculs.

2. La régression

La régression est une technique d'apprentissage supervisé, elle s'agit de la prédiction des valeurs numérique continues pour un ensemble de données à partir d'une base d'apprentissage, elle prédit une variable de résultat continue (y) en fonction de la valeur d'une ou de plusieurs variables prédictives (x). Les entrées sont appelées valeurs indépendantes la sortie est appelée la valeur dépendante. Il existe des poids appelés coefficients, qui déterminent dans quelle mesure chaque valeur d'entrée contribue au résultat, ou son importance [3].

Il existe plusieurs types de régression :

- **Régression linéaire :** convient aux valeurs dépendantes qui peuvent être ajustées avec une ligne droite (fonction linéaire).
- **Régression polynomiale :** convient aux variables dépendantes qui peuvent être ajustées par une courbe ou une série de courbes.
- **Régression logistique :** adaptée aux variables dépendantes qui sont binaires et donc pas normalement distribuées.
- **Régression pas à pas** : une technique automatisée qui peut gérer la dimensionnalité élevée des variables indépendantes.
- **Régression de crête** : une technique de régression qui aide à la multicolinéarité, des variables indépendantes qui sont fortement corrélées. Il ajoute un biais aux estimations de régression, pénalisant les coefficients à l'aide d'un paramètre de rétrécissement.
- **Régression Lasso** : comme la régression Ridge, réduit les coefficients pour résoudre la multicolinéarité, mais elle réduit également les valeurs absolues, ce qui signifie que certains des coefficients peuvent devenir nuls. Cela effectue la « sélection des fonctionnalités », supprimant certaines variables de l'équation.
- **Régression ElasticNet :** combine les régressions Ridge et Lasso et est formée à la régularisation, utilisée pour éviter le overfitting (surapprentissage), L1 et L2, échangeant entre les deux techniques.

I.2.2 L’apprentissage profond (Deep Learning)

I.2.2.1 Historique

L’idée du Deep Learning n’est pas une idée récente, mais elle date en réalité des années 1980, plus particulièrement suite aux travaux de réseaux de neurones multicouches et aux travaux de certains pionniers du Machine Learning et du Deep Learning comme le français Yann Le Cun. En collaboration avec deux autres informaticiens, Kunihiki Fukushima et Geoffrey Hinton, ils mettent au point un type d’algorithme particulier appelé Convolutional neural network [4].

Bien que cette approche donne des résultats, ses progrès et son évolution sont limités par les progrès technologiques en matière de micro-processeurs, de puissance de calculs, et du manque d’accessibilités à des données afin de pouvoir entraîner les réseaux neurones. Cependant certains chercheurs ont continué à travailler sur ce modèle pendant environs deux décennies et avec l’aide des évolutions en matière de technologies, mais surtout avec la disponibilité toujours plus grande de données, ont pu améliorer cette technique.

En 2007 le Stanford Vision Lab, avec Fei-Fei Li à sa tête, développent un agrégateur d’images où sont consignés et étiquetés quelques millions, de photos ImageNet. En 2010, ImageNet regroupe 15millions d’images toutes catégorisées en fonction de leurs caractéristiques propres (véhicules, animaux, etc.) [4].

En 2012 le Deep Learning est remis au au goût du jour avec un succès retentissant au ImageNet Large Scal Visual Recognition Challenge (ILSVRC) qui est un concours annuel de reconnaissance d’image fondée par l’université de Stanford, dans le cadre de son laboratoire STANFORD VISION LAB. Plusieurs équipes de chercheurs en informatique s’affrontent dans ce concours tous les ans afin de décerner la victoire au programme ayant eu le plus faible taux d’échec. Et alors que les algorithmes d’apprentissage profond sont absents de la compétition, en 2012 c’est bel et bien un algorithme de Deep Learning qui va remporter l’édition 2012 à la surprise générale [4].

I.2.2.2 Définition

L’apprentissage profond est un ensemble d’algorithmes d’apprentissage automatique qui tentent d’apprendre à plusieurs niveaux, correspondant différents niveaux d’abstraction. Il a la capacité d’extraire des caractéristiques à partir des données brutes grâce aux multiples couches de traitement composé de multiples transformations linéaires et non linéaires et apprendre sur ces caractéristiques petit à petit à travers chaque couche avec une

intervention humaine [5]. La figure.I.1 nous montre un schéma illustratif de deep learning avec les multiples couches.

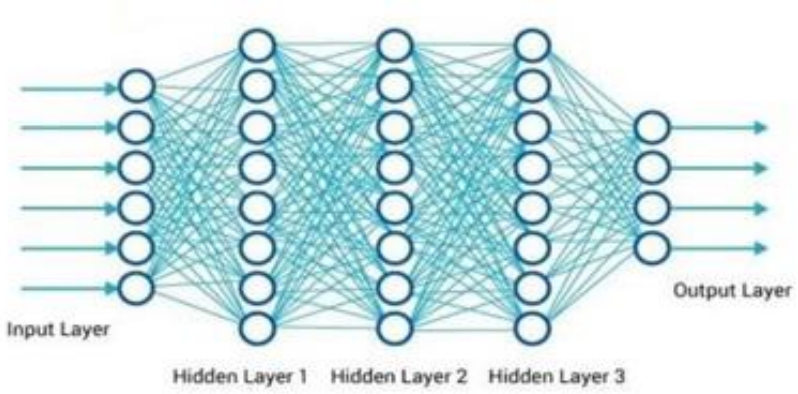


Figure I.1 Schéma illustratif de deep learning avec plusieurs couches [6].

I.2.2.3 Avantages de deep learning

Le deep learning est une véritable révolution dans l'IA et va bien au-delà de l'apprentissage automatique. Il est capable de résoudre une grande variété des problèmes de l'IA, comme :

- Améliorer le développement traditionnel dans des tâches de l'IA.
- Exploiter une grande quantité de données telle que les big dita.
- S'adapter à plusieurs types de problème.
- Extraire les caractéristiques de façon automatique.

La figure.I.2 ci-dessous explique la différence entre le machine et le deep learning. Machine learning dépend d'un ensemble de données généré par l'homme qui apprend au logiciel comment définir les données. Par contre le deep learning, lorsqu'un réseau de neurone traite une entrée, il crée ses propres couches en fonction de l'entrée et de la sortie des données.

Pour récapituler la différence entre l'apprentissage automatique et l'apprentissage profond:

- L'apprentissage automatique utilise des algorithmes pour analyser les données, apprendre de ces données et prendre des décisions éclairées en fonction de ce qu'il a appris
- L'apprentissage profond structure les algorithmes en couches pour créer un «réseau neuronal artificiel» capable d'apprendre et de prendre des décisions intelligentes par lui-même

- L'apprentissage profond est un sous-domaine de l'apprentissage automatique. Bien que les deux appartiennent à la grande catégorie de l'intelligence artificielle, l'apprentissage profond est ce qui alimente l'intelligence artificielle la plus humaine.

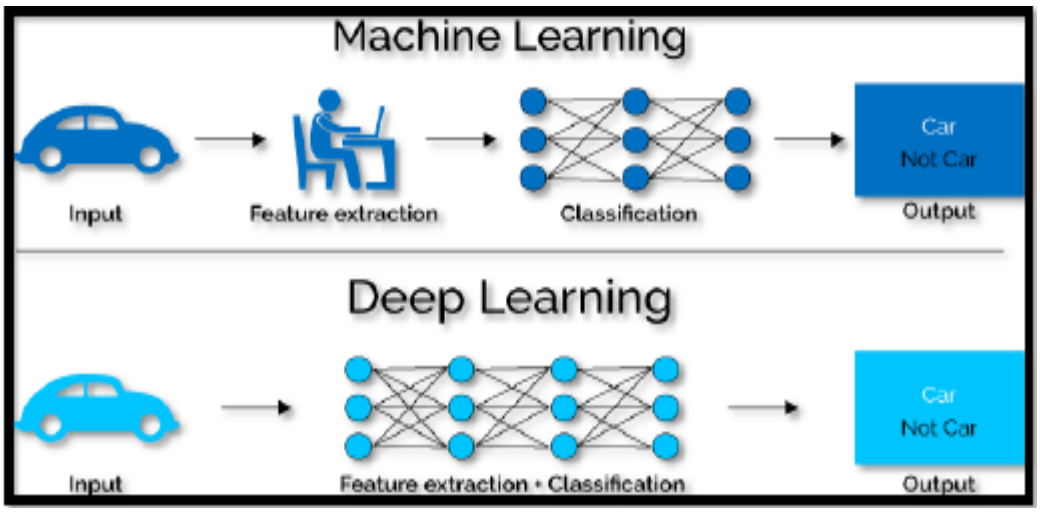


Figure I.2 Machine learning ET le Deep learning [7].

I.2.2.4 Domaines d'application de deep learning

Les modèles d'apprentissage se développent dans le domaine de l'informatique appliquée aux NTIC (reconnaissance visuelle, par exemple d'un panneau de signalisation par un robot ou une voiture autonome, et vocale notamment) à la robotique, la bio-informatique, la reconnaissance ou comparaison de formes, la sécurité, la santé, etc. La pédagogie assistée par l'informatique et plus généralement à l'intelligence artificielle. L'apprentissage profond peut par exemple permettre à un ordinateur de mieux reconnaitre des objets hautement déformables et/ou analyser par exemple les émotions révélées par un visage photographie ou filmé, ou analyser les mouvements et position des doigts d'une main, ce qui peut être utile pour traduire les langages des signes, améliorer le positionnement automatique d'une caméra, etc. Ils sont utilisés pour certaines formes d'aide au diagnostic médical (ex : reconnaissance automatique d'un cancer en imagerie médicale), ou de prospective ou de prédiction (ex : prédiction des propriétés d'un sol filmé par un robot).

I.3 Les réseaux de neurones

I.3.1 Historique

Les réseaux neuronaux ont vu le jour qu'en 1943 grâce à W.MCCulloch et W.Pitts [8], ils ont pu démontrer que le cerveau est équivalent à une machine de Turing, la pensée devient alors purement des mécanismes matériels et logiques. Ils déclarèrent en 1955 « Plus nous apprenons de choses au sujet des organismes, plus nous sommes amenés à conclure qu'ils ne sont pas simplement analogues aux machines, mais qu'en est-il de cette machine ». La démonstration de McCulloch et Pitts a été l'un des acteurs importants de la création de la cybernétique. En 1949, D. Hebb [9] présenta dans son ouvrage « The Organization of Behavior » une règle d'apprentissage. De nombreux modèles de réseaux, aujourd'hui, s'inspirent encore de la règle de Hebb. En 1958, F.Rosenblatt [10] développe le modèle du Perceptron. C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones, une couche de perception et une couche liée à la prise de décision. C'est le premier système artificiel capable d'apprendre par expérience. Dans la même période, le modèle de L'Adaline (ADAptive LINar Element) a été présenté par B.Window [11], chercheur américain à Stanford. Ce modèle sera par la suite le modèle de base des réseaux multicouches. En 1969, M. Minsky et S.Papert publient une critique des propriétés du Perceptron. Cela va avoir une grande incidence sur la recherche dans ce domaine. Elle va fortement diminuer jusqu'en 1972, où T.Kohonen [12] a présenté ses travaux sur les mémoires associatives et propose des applications à la reconnaissance des formes. C'est en 1982 que J.Hopfield présente son étude d'un réseau complètement rebouclé, dont il analyse la dynamique [13].

I.3.2 Définition

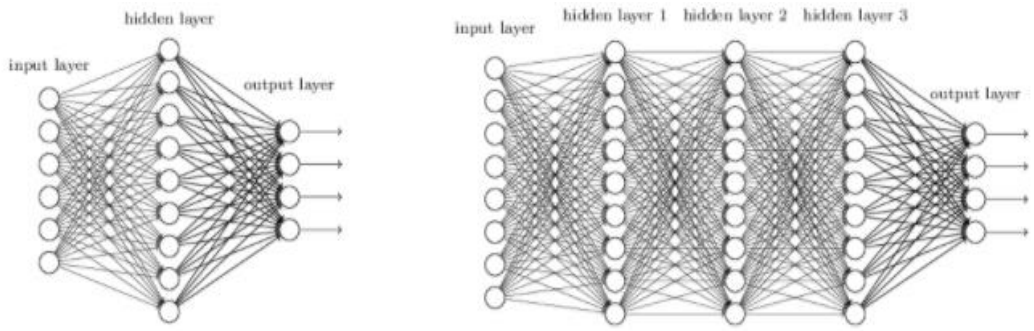
Un réseau neuronal est l'association, en un graphe plus ou moins complexe, d'objets élémentaires, les neurones formels. Les principaux réseaux se distinguent par l'organisation du graphe, c'est-à-dire leur architecture, son niveau de complexité (le nombre de neurones, présence ou non de boucles de rétroaction dans le réseau), par le type des neurones (leurs fonctions de transition ou d'activation) et enfin par l'objectif visé : apprentissage supervisé ou non, optimisation, systèmes dynamiques, etc. [14].

I.3.3 Les réseaux de neurones artificiels et profonds

Les réseaux de neurones artificiels sont un système d'apprentissage supervisé constitué d'un grand nombre d'éléments simples, appelés neurones ou perceptrons. Chaque neurone peut prendre des décisions simples et alimentes ses décisions vers d'autres neurones, organisés en couches interconnectées. Ensemble, le réseau neuronal peut émuler presque toutes les fonctions et répondre à pratiquement toutes les questions, avec suffisamment d'échantillons de formation et de puissance de calcul. Un réseau neuronal « peu profond » ne comporte que trois couches de neurones comme illustré en la figure.I.3.(a))

- Une couche d'entrée qui accepte les variables ou entrées indépendantes du modèle.
- Une couche cachée.
- Une couche de sortie qui génère les prédictions.

Un réseau neuronal profond a une structure similaire, mais il a deux ou plusieurs « couches cachées » de neurones qui traitent les entrées (voir figure.I.3.(b)). Goodfellow, Bengio et Courville [15] ont montré que si les réseaux neuronaux peu profonds sont capables de résoudre des problèmes complexes, les réseaux de neurones profonds sont plus précis et améliorent la précision à mesure que de nouvelles couches de neurones sont ajoutées. Des couches supplémentaires sont utiles jusqu'à une limite de 9 à 10. Aujourd'hui la plupart des modèles et implémentations de réseaux neuronaux utilisent un réseau profond entre 3 à 10 couches de neurones.



(a) (b)

Figure I.3 Réseau de neurone artificiel (a) et profond(b) [16].

I.3.4 Les fonctions d'activation

Une fonction d'activation est une équation mathématique qui détermine la sortie de chaque élément (perceptron ou neurone) dans le réseau neuronal. Comme illustré en figure.I.4**,** il prend l'entrée de chaque neurone et la transforme en sortie, généralement entre

un 0 ou -1 et 1. Les fonctions d'activation classiques utilisées dans les réseaux de neurones incluent la fonction step sigmoid et tanh. De nouvelles fonctions d'activation, destinées à améliorer l'efficacité des réseaux de neurones, incluent ReLu et Swish [17].

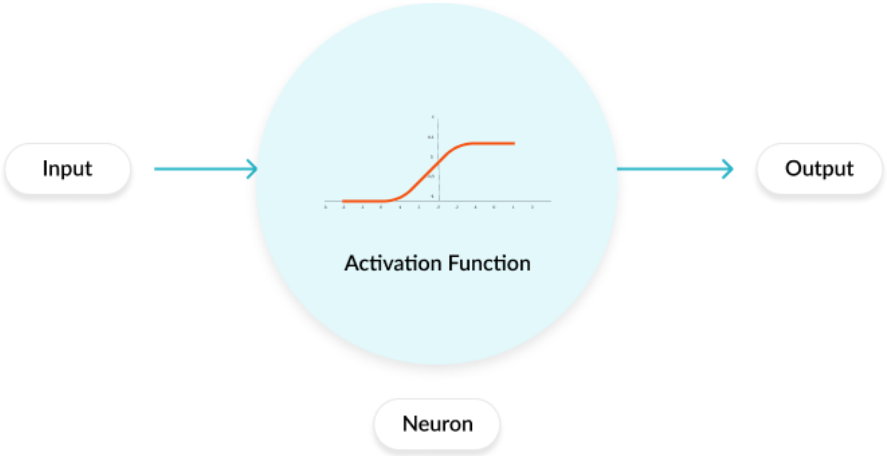


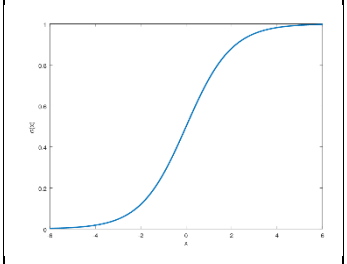
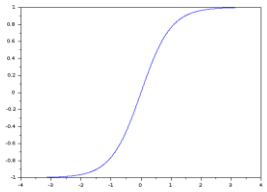
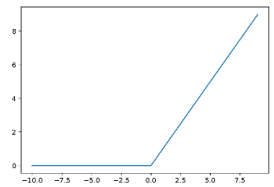
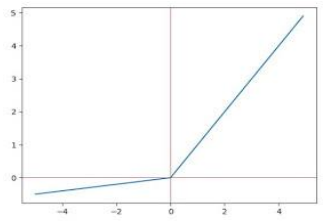
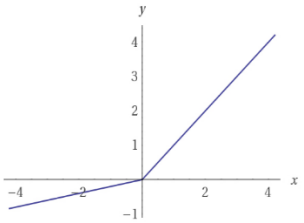
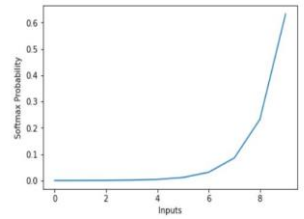
Figure I.4 Structure simplifié d'un neurone artificiel [18].

Dans un réseau neuronal, les entrées, qui sont généralement des valeurs réelles, sont introduites dans les neurones du réseau. Chaque neurone a un poids, et les entrées sont multipliées par le poids et introduites dans la fonction d'activation.

La fonction d'activation est la fonction qui permet de traiter l'information qui arrive à un neurone artificiel en machine learning, comme le fait ceux du cerveau avec les signaux électriques qu'ils reçoivent [19].

La sortie de chaque neurone est l'entrée des neurones dans la couche suivante du réseau, et donc les entrées se succèdent à travers de multiples fonctions d'activation jusqu'à ce que finalement, la couche de sortie génère une prédiction. Les réseaux de neurones reposent sur des fonctions d'activation non linéaires, la dérivée de la fonction d'activation aide le réseau à apprendre à travers le processus de rétropropagation.

Le tableau ci-dessous décrit les principales fonctions d'activation existantes :

Noms	Description	Equation	Graphe
Sigmoïde	Elle génère des valeurs comprises entre zéro et un. Pour des valeurs très élevées ou basses des paramètres d'entrée, le réseau peut être très lent à atteindre une prédiction.	$f(x) = {}^{1}/_{(1+e^{-x})}$	
TanH	Centrée sur zéro, ce qui facilite la modélisation d'entrées fortement négatives, fortement positives ou neutres.	$f(x) = {}^{2}/_{(1+e^{-2x}-1)}$	
ReLu	Très efficace en termes de calcul mais n'est pas en mesure de traiter des entrées qui approchent de zéro ou négatives.	$f(x) = \begin{cases} 0 \; si \; x < 0 \\ x \; si \; x \geq 0 \end{cases}$	
Leaky ReLu	Petite pente positive dans sa zone négative, lui permettant de traiter des valeurs nulles ou négatives.	$f(x) = \begin{cases} 0{,}01x \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$	
Parametric ReLu	Permet d'apprendre la pente négative, en effectuant une rétropropagation pour apprendre la pente la plus efficace pour les valeurs d'entrée zéro et négative.	$f(x) = \begin{cases} \alpha x \; si \; x < 0 \\ x \; si \; x \geq 0 \end{cases}$	
Softmax	Utilisée pour les neurones de sortie. Il normalise les sorties pour chaque classe entre 0 et 1 et renvoie la probabilité que l'entrée appartient à une classe spécifique.	$f(x) = e^{x_i} \Big/ \sum_{i=1}^{k} e^{x_k}$ K : Le nombre de classes	

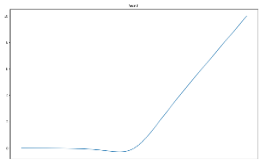
Swish	Nouvelle fonction d'activation découverte par les chercheurs de Google. Il fonctionne mieux que ReLu avec un niveau similaire d'efficacité de calcul.	$f(x)$ $= x.\text{Sigmoïde}(x)$	

Tableau I.1 Les fonctions d'activation.

I.3.5 Les Hyperparamètres de réseau neuronal

Les hyperparamètres déterminent la structure du réseau neuronal, sa formation et le fonctionnement de ses différents éléments. L'optimisation des hyperparamètres est un art : il existe plusieurs façons, allant des essais et erreurs manuels aux méthodes algorithmiques sophistiquées.

I.3.5.1 La différence entre les paramètres et les hyperparamètres

- **Un paramètre de modèle :** Le paramètre est interne aux réseaux de neurones. Il évolue durant l'ensemble du processus d'entrainement, lors de la rétropagation (une méthode pour calculer le gradient de l'erreur pour chaque neurone d'un réseau de neurones), il est utilisé pour faire des prédictions dans un modèle d'apprentissage profond.
- **Un hyperparamètre** est un paramètre externe défini par l'opérateur du réseau neuronal. Par exemple, le nombre d'itérations d'entraînement, le nombre de couches cachées ou la fonction d'activation. Différentes valeurs des hyperparamètres peuvent avoir un impact majeur sur les performances du réseau.

I.3.5.2 Liste des hyperparamètres

Nous avons décrit ci-dessus un des hyperparamètres des réseaux de neurones à savoir « la fonction d'activation ». Nous décrivons brièvement dans le tableau suivant d'autres hyperparamètres que nous avons classifié en deux catégories :

Catégorie	**Nom de l'hyperparamètre**	**Description**
	Nombre de couches cachées	Il s'agit du cœur de perceptron, là où les relations entre les variables vont être mises en exergue. Le nombre a utilisé

Hyperparamètres liés à la structure du réseau neuronal		dépend du problème à traiter, si le jeu de données est simple, deux couches cachées ou moins, sont souvent suffisantes.
	Abandon	Améliore les performances des réseaux de neurones sur les tâches d'apprentissage supervisé en obtenant des résultats de pointe sur de nombreux ensembles de données de référence.
	Fonction d'activation	Une fonction d'activation est une équation mathématique qui détermine la sortie de chaque élément dans le réseau neuronal.
Hyperparamètres liés à l'algorithme d'entraînement	**Taux d'apprentissage**	Facteur multiplicatif appliqué au gradient afin de faire varier le gain du gradient.
	Époque	Fait référence à un cycle à travers l'ensemble de données d'entraînement complet.
	Itérations	Les itérations sont le nombre de lots nécessaires pour réaliser une Epoque
	Taille du lot	Nombre total de données d'entrainement présentés dans un seul lot.
	Algorithme d'optimisation	Une procédure mathématique qui permet à une fonction d'obtenir la valeur maximale ou minimale. Par exemple la descente de gradient
	Élan	Calcule une moyenne glissante pondérée exponentiellement des gradients au fil du temps

Tableau I.2 Liste des hyperparamètres.

I.3.5.3 Les méthodes de réglage des hyperparamètres

Dans une expérience de réseau de neurones, on essaye généralement de nombreuses valeurs possibles des hyperparamètres et on choisit ce qui fonctionne le mieux. Afin d'évaluer le succès de différentes valeurs, on convertit le réseau à l'aide de chaque ensemble d'hyperparamètres et le test par rapport à notre ensemble de validation. Si notre ensemble d'apprentissage est petit, on peut utiliser la validation croisée en divisant l'ensemble d'apprentissage en plusieurs groupes, en entraînant le modèle sur chacun des groupes puis en le validant sur les autres groupes. Voici les méthodes courantes utilisées pour régler les hyperparamètres :

- Réglage manuel de l'hyperparamètre : un opérateur expérimenté peut deviner les valeurs des paramètres qui atteindront une très grande précision. Cela nécessite des essais et des erreurs.
- Recherche dans la grille : cela implique de tester systématiquement plusieurs valeurs de chaque hyperparamètre et de recycler le modèle pour chaque combinaison.
- Recherche aléatoire : une étude de Bergstra et Bengio [20] a montré que l'utilisation de valeurs hyperparamétriques aléatoires est en fait plus efficace que la recherche manuelle ou la recherche sur la grille.
- Optimisation bayésienne : une méthode qui entraîne le modèle avec différentes valeurs des hyperparamètres encore et encore, et essaie d'observer la forme de la fonction générée par différentes valeurs de paramètres. Il étend ensuite cette fonction pour prédire les meilleures valeurs possibles. Cette méthode offre une précision supérieure à la recherche aléatoire.

I.3.6 Les principales architectures des réseaux de neurones

Un réseau de neurones peut prendre des formes différentes selon l'objet de la donnée qu'il traite et selon sa complexité et la méthode de traitement de la donnée.

Les architectures ont leurs forces et faiblesses ce qui les rend plus en adéquation pour une tâche donnée, elles peuvent être combinées pour optimiser les résultats. Le choix de l'architecture s'avère ainsi crucial, déterminer l'objectif à atteindre est le meilleur moyen de choisir l'approche la plus adéquate en fonction de ses avantages et surtout de ses contraintes.

Nous présentons dans ce qui suit les principales architectures des réseaux de neurones.

I.3.6.1 Les réseaux de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs, appelé aussi convnet (pour «Convolutional Network»), ou encore CNN (pour «Convolutional Neural Network»), se sont révélés très efficaces pour les tâches impliquant des données étroitement liées, principalement dans le domaine de la vision par ordinateur. On distingue deux parties dans un CNN, une première partie que l'on appelle la partie convolutive du modèle et la seconde partie, que l'on va appeler la partie classification du modèle qui correspond à un modèle MLP (Multi Layers Perceptron) [21].

C'est un réseau de neurones multicouches et plus précisément c'est un réseau profond composé de quatre types de couches : la couche de convolution, la couche de pooling, la couche de correction ReLU et la couche fully connected.

- **Couche de convolution :** c'est la couche la plus importante et le cœur des éléments constitutif du réseau convolutif, et c'est aussi elle qui effectue le plus grand de calculs lourds. Son but est de repérer la présence d'un ensemble de features dans les images reçues en entrée.
- **Couche de mise en commun (pooling) :** est souvent placée entre deux couches de convolution : elle reçoit en entrée plusieurs feature maps, et applique à chacune d'entre elles l'opération de pooling. L'opération de pooling (ou sub-sampling) consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes.
- **La couche de correction ReLU :** remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.
- **La couche fully-connected :** constitue toujours la dernière couche d'un réseau de neurones, convolutifs ou non, elle n'est donc pas caractéristique d'un CNN. Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

La figure suivante montre l'architecture d'un réseau de neurone de type CNN :

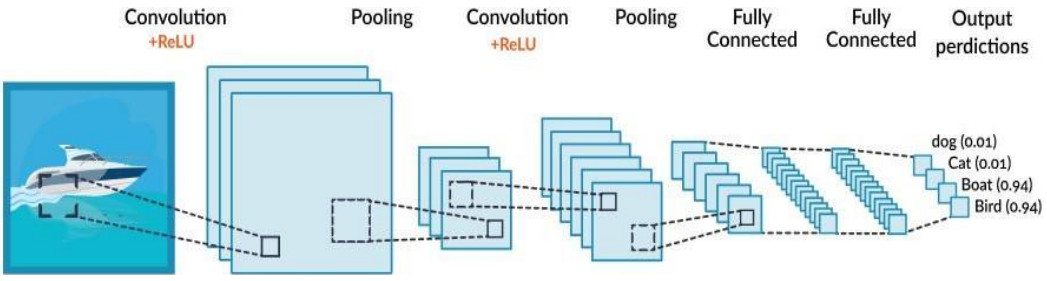


Figure I.5 Fonctionnement d'un réseau de neurones convolutifs [22].

Les réseaux de neurones de type CNN sont utilisés notamment dans :

- La reconnaissance faciale.
- Identifier et classer les objets du quotidien en images.
- Alimenter la vision dans les robots et les véhicules autonomes.
- Reconnaître les scènes et suggérer des légendes pertinentes.

I.3.6.2 Les réseau de neurones récurrents (RNN)

Un réseau neuronal récurrent est destiné à gérer les données d'entrée de nature séquentielle. Par exemple, du texte écrit, de la vidéo, de l'audio ou plusieurs événements qui se produisent l'un après l'autre, comme dans les analyses de réseau ou de sécurité. Un RNN accepte une série d'entrées, se souvient des entrées précédentes et, à chaque nouvelle entrée, ajoute une nouvelle couche de compréhension [23].

La figure I.6 ci-dessus montre un RNN déroulé (ou déplié) dans un réseau complet. Par déroulement, nous entendons simplement que nous écrivons le réseau pour la séquence complète. Par exemple, si la séquence qui nous intéresse est une phrase de 5 mots, le réseau sera déroulé dans un réseau neuronal à 5 couches, une couche pour chaque mot.

Un RNN examine une série d'entrées dans le temps, X0, X1, X2, jusqu'à Xt. Par exemple, cela pourrait être une séquence d'images dans une vidéo ou des mots dans une phrase. Le réseau neuronal a une couche de neurones pour chaque entrée. Lorsque le réseau RNN apprend, il effectue une rétropropagation dans le temps (Backpropagation through time BPTT) une forme de rétropropagation multicouche. BPTT utilise la règle de chaîne pour revenir du dernier pas de temps (Xt), progressivement à chaque étape précédente, à chaque fois en utilisant la descente de gradient pour découvrir les meilleurs poids pour chaque neurone, et également apprendre les poids optimaux qui régissent le transfert d'informations entre une étape de temps à l'autre.

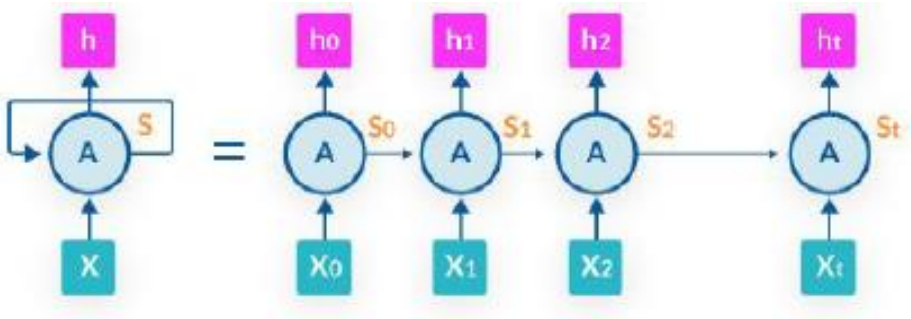


Figure I.6 Réseau de neurones déroulé [24].

Les RNN sont notamment utilisés pour :

- La modélisation linguistique et génération de texte : les modèles fonctionnent en prédisant le caractère suivant, le mot suivant ou la phrase suivante les plus appropriés dans une chaîne de texte.
- La traduction automatique : le texte dans la langue source est saisi par lots et le modèle tente de générer le texte correspondant dans la langue cible.
- La reconnaissance vocale : l'entrée est constituée de signaux acoustiques analysés à partir d'un enregistrement audio ; le modèle produit l'élément phonétique de langue le plus probable pour chaque partie de l'enregistrement.
- La génération de légendes d'image : l'entrée est une image et le modèle identifie les éléments de l'image et génère du texte qui la décrit.
- La détection d'anomalies de séries chronologiques : l'entrée est une série de données séquentielles, comme une série d'événements dans un incident de cyber sécurité potentiel. Le modèle prédit si les données sont une anomalie par rapport au comportement normal.
- Balisage vidéo : l'entrée est une série d'images vidéo et le modèle génère une description textuelle de chaque image de la vidéo.

I.3.6.3 Les modèles Autoencodeur (AE)

Les auto-encodeurs sont des algorithmes d'apprentissage non supervisé à base de réseaux de neurones artificiels, qui permettent de construire une nouvelle représentation d'un jeu de données. Généralement, celle-ci est plus compacte, et présente moins de descripteurs, ce qui permet de réduire la dimensionnalité du jeu de données. L'architecture d'un auto-encodeur est constituée de deux parties : l'encodeur et le décodeur [25].

L'encodeur est constitué par un ensemble de couches de neurones, qui traitent les données afin de construire de nouvelles représentations dites "encodées". À leur tour, les couches de neurones du décodeur, reçoivent ces représentations et les traitent afin d'essayer de reconstruire les données de départ. Les différences entre les données reconstruites et les données initiales permettent de mesurer l'erreur commise par l'auto-encodeur. L'entraînement consiste à modifier les paramètres de l'auto-encodeur afin de réduire l'erreur de reconstruction mesurée sur les différents exemples du jeu de données.

La forme la plus simple d'un auto-encodeur est un réseau de neurones non récurrents qui se propage vers l'avant, très semblable au perceptron multicouches - ayant une couche d'entrée, une couche de sortie ainsi qu'une ou plusieurs couches cachées les reliant -, mais avec toutefois une couche de sortie possédant le même nombre de nœuds que la couche d'entrée, son objectif étant de reconstruire ses entrées.

La figure suivante schématise un auto-encodeur simple, dont l'encodeur (encoder) traite des images (inputs), afin de les représenter comme des points dans un espace à deux dimensions (encoded representation), puis décode cette représentation (decoder), afin de retrouver les données de départ (output).

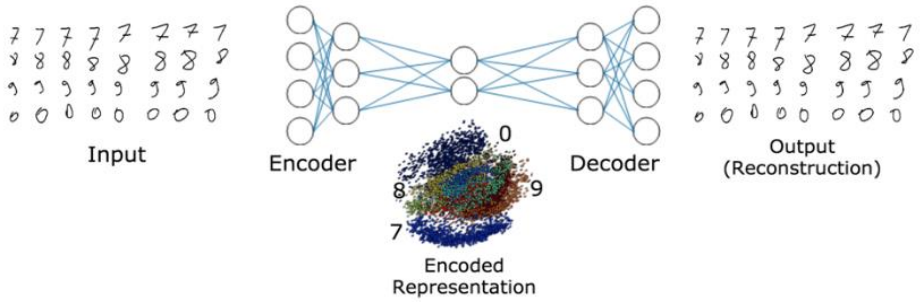


Figure I.7 Autoencodeur [26].

I.3.6.4 Les Mécanismes d'attention (AM)

Les mécanismes d'attention (Attention Models) sont basés sur une intuition sensorielle commune que nous focalisons sur une certaine partie lors du traitement d'une grande quantité d'informations [27].

Un mécanisme d'attention est la capacité d'apprendre à se concentrer sur des parties spécifiques d'une donnée complexe, par exemple une partie d'une image ou un mot dans une phrase. Des mécanismes d'attention peuvent être incorporés dans les architectures de traitement de la langue naturelle et de reconnaissance d'images pour aider un réseau de neurones artificiels à apprendre sur quoi "se concentrer" lorsqu'il fait des prédictions.

I.3.6.5 L'apprentissage par renforcement (RLM)

Reinforcement Learning models ou apprentissage par renforcement est une méthode d'apprentissage basé sur l'expérience. Cette méthode consiste à laisser l'algorithme apprendre de ses propres erreurs, à partir d'expériences successives, ce qu'il

convient de faire de façon à optimiser une récompense quantitative au cours du temps et à trouver la meilleure solution [28].

L'intelligence artificielle se retrouve directement confrontée à des choix. Si elle se trompe, elle est "pénalisée". Au contraire, si elle prend la bonne décision, elle est "récompensée". Afin d'obtenir toujours plus de récompenses.

On distingue deux types d'apprentissage par renforcement **:** positif et négatif. Dans le cas du renforcement positif, un événement qui survient suite à comportement spécifique renforce la fréquence de ce comportement. L'événement a donc un effet " positif " sur le comportement du modèle.

Ce type de renforcement permet de maximiser les performances et de maintenir le changement sur le long terme. En revanche, un excès de renforcement peut conduire à une surcharge d'états et faire diminuer les résultats. Dans le cas du renforcement négatif, le comportement est renforcé parce que les conditions négatives sont empêchées ou évitées. Ceci permet d'augmenter la fréquence du comportement adéquat, mais permet uniquement d'atteindre un résultat minimal.

I.3.7 Avantages et inconvénients

Les réseaux de neurones sont largement utilisés ces dernières années dans beaucoup de domaines car ils présentent plus d'avantages que d'inconvénients. Nous citons ci-dessous ses avantages et inconvénients.

Les avantages :

- Précision souvent élevée.
- Robuste, marche lorsque le jeu d'apprentissage contient des erreurs.
- Permet de résoudre des problèmes complexes par un apprentissage automatique (pas besoin de définir manuellement les règles de l'IA comme dans la méthode classique).
- La sortie peut être une valeur discrète, continue, ou un vecteur de plusieurs valeurs discrètes ou continues.
- S'adapte bien aux problèmes qui nécessitent des niveaux d'abstractions élevée qui sont difficiles à décrire explicitement avec la méthode classique (coder tous les traitements et les conditions).

Inconvénients :

- Apprentissage long (dû au nombre de paramètres souvent élevés).

- Difficile de comprendre le modèle.
- Non évidence du choix des hyperparamètres

I.4 Les Plateformes de développement

Nous vous présentons ci-dessous trois principaux Framework d'apprentissage profond. Nous décrirons chacun séparément, puis nous comparerons les différentes plateformes :

I.4.1 TensorFlow

TensorFlow est une bibliothèque open-source développée par l'équipe Google en 2011. Elle implémente des méthodes d'apprentissage automatique basées sur le principe des réseaux de neurones profonds [29].

À l'origine il s'appelait DistBelief, par la suite, le code source de DistBelief a été modifié et cet outil est devenu une bibliothèque basée application. En 2015, il a été renommé TensorFlow et Google l'a rendu open source. Depuis lors, il a subi plus de 21000 modifications par la communication et est passé en version 1.0 en février 2017.

TensorFlow est une boîte à outils permettant de résoudre des problèmes mathématiques extrêmement complexes avec aisance. Elle permet aux chercheurs de développer des architectures d'apprentissages expérimentaux et de les transformer en logiciels. On peut le concevoir comme un système de programmation dans lequel les calculs sont représentés sous forme de graphiques de dataflow. Les nœuds du graphique représentent les opérations mathématiques, et chaque connexion entre ces nœuds sont des flèches de données multidimensionnelles communiquées entre elles : les tensors.

TensorFlow est Multi-Plateforme. Il fonctionne sur presque tous les processeurs graphiques et les processeurs, y compris les plates-formes mobiles et intégrées, et même les unités de traitement de tenseurs (TPU), qui sont du matériel spécialisé permettant d'effectuer des calculs de tenseurs.

TensorFlow regroupe un grand nombre de modèles et d'algorithmes de Machine Learning et de Deep Learning. Son API front-end de développement d'applications repose sur le langage de programmation Python, tandis que l'exécution de ces applications s'effectue en C++.

I.4.2 Keras

Keras est une bibliothèque open source écrite en python, permet d'interagir avec les algorithmes de réseaux de neurones profonds et de machine Learning [30].

C’est une librairie Python qui encapsule l’accès aux fonctions proposées par plusieurs librairies de machine learning, elle s’intègre avec les outils de Deep Learning (en particulier TensorFlow) à bas niveau, ce qui nous permet d’implémenter toutes les fonctionnalités que nous souhaiterions utiliser. De fait, Keras n’implémente pas nativement les méthodes. Elle sert d’interface avec TensorFlow simplement en proposant des fonctions et procédures relativement simples à mettre en œuvre. Elle propose une API simple et cohérente, minimisant le nombre d’actions requises pour les cas les plus communs, et les messages d’erreur sont suffisamment explicites pour permettre une résolution simple des problèmes.

En novembre 2017, nous comptabilisions plus de 200 000 utilisateurs de Keras, ce qui en fait l’outil le plus utilisé dans le monde industriel et dans la recherche, après TensorFlow lui-même.

I.4.3 PyTorch

PyTorch est une bibliothèque logicielle Python open source d'apprentissage automatique qui s'appuie sur Torch développée par Facebook, il permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond. Ces calculs sont optimisés et effectués soit par le processeur (CPU) soit, lorsque c'est possible, par un processeur graphique (GPU) supportant CUDA[1]. Il est issu des équipes de recherche de Facebook [31].

PyTorch est dérivé d'un logiciel antérieur, Torch, qui s'utilisait avec le langage Lua, mais il est indépendant de Lua et se programme en Python. Il y a une certaine ressemblance entre PyTorch et Torch, mais pour des raisons pratiques vous pouvez considérer que ce sont des projets différents.

Les développeurs PyTorch offrent aussi LibTorch qui permet d'implémenter des extensions à PyTorch à l'aide de C++ et d'implémenter des applications d'apprentissage machine en C++ pur. Les modèles Python écrits avec PyTorch peuvent être convertis et utilisés en C++ avec TorchScript, ils permettent de:

- manipuler des tenseurs (tableaux multidimensionnels), de les échanger facilement avec Numpy[2] et d'effectuer des calculs efficaces sur CPU ou GPU (par exemple, des produites de matrices ou des convolutions);

[1] Est une plateforme de calcul parallèle et un modèle de programmation développé par NVIDIA pour des calculs généraux utilisant le GPU.

[2] Est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

- calculer des gradients pour appliquer facilement des algorithmes d'optimisation par descente de gradient. PyTorch utilise la bibliothèque autograd.

Ci-dessous un tableau qui décompose les fonctionnalités de TensorFlow, Keras et PyTorch.

plateformes / Critères	Keras	PyTorch	TensorFlow
Niveau API	Haute	Faible	Haut et bas
Architecture	Simple, concis, lisible	Complexe, moins lisible	Pas facile à utiliser
Ensembles des données	Ensembles de données plus petits	Grands ensembles de données, hautes performances	Grands ensembles de données, hautes performances
Débogage	Réseau simple, le débogage n'est donc pas souvent nécessaire	Bonnes capacités de débogage	Difficile de procéder au débogage
A-t-il formé des modèles?	Oui	Oui	Oui
Popularité	Le plus populaire	Troisième plus populaire	Deuxième plus populaire
Vitesse	Performances lentes et faibles	Rapide et performant	Rapide et performant
Écrit en	Python	Lua	C++, CUDA, Python

Tableau I.3 Les fonctionnalités des plateformes.

I.5 Conclusion

Le Machine learning (apprentissage automatique) et le deep learning (apprentissage profond) sont les deux concepts les plus importants qui rendent l'intelligence artificielle possible.

Le machine learning qui apprend sur la base d'un grand ensemble de données structurées. Le deep learning n'a pas besoin de données structurées, mais des systèmes qui fonctionnent à partir de plusieurs couches de réseaux neuronaux sans être guidés par l'homme.

Nous avons vu dans ce chapitre qu'est-ce que le deep learning et ses domaines d'application, avec les notions importantes qui sont en relation avec (définition, Architectures, etc.), les réseaux de neurones artificiels et réseaux de neurones profonds, avec quelques plateformes.

Parmi les domaines d'application du deep learning, on trouve les systèmes de recommandation. Nous décrivons ce domaine dans le prochain chapitre, car il s'agit du contexte de notre travail.

Chapitre II

II.1 Introduction

Souvent on se demande Quel film regarder ? Quel article acheter ? Que visiter lorsque l'on est en voyage ? On se retrouve devant un volume très grand de propositions, ce qui rend la tâche du choix très difficile. Par exemple, Netflix disposait en 2007 de plus de 17,000 films dans sa base de données et ce nombre n'a cessé de se croitre au fil des années. La liste des possibilités qui s'offrent à nous est donc en général de très grande taille, l'évaluation de ces possibilités pour trouver ce qui nous convient le plus est une tâche difficile et peut consommer beaucoup de notre temps. Les systèmes de recommandation ont été utilisés afin de faire face au problème de surcharge et de richesse d'informations disponibles notamment à travers le Web ou les e-services.

Les systèmes de recommandation visent à proposer à un utilisateur actif une ou plusieurs recommandations d'items susceptibles de l'intéresser. Ces recommandations peuvent concerner un article à lire, un livre à commander, un film à regarder, un restaurant à choisir, etc.

Dans ce chapitre, nous commençons par définir ce qu'un système de recommandation et ses objectifs. Ensuite, nous présentons les trois approches de filtrage qui permettent la recommandation. Nous enchainons par la présentation des différents types de recommandation. En fin, nous présentons quelques travaux de l'état de l'art des systèmes de recommandation qui sont basés sur les techniques de l'IA.

II.2 Historique

« Information Lens System » [32] peut être considérée comme le premier système de recommandation. À l'époque, l'approche la plus commune pour le problème de partage d'informations dans l'environnement de messagerie électronique était la liste de distributions basée sur les groupes d'intérêt. La première définition pour le filtrage a été donnée aussi par Malone : " Même si le terme a une connotation littérale de laisser les choses dehors (filtrage négatif : enlèvement), nous l'utilisons ici dans un sens plus général qui consiste à sélectionner les choses à partir d'un ensemble plus large de possibilités (filtrage positif : sélection) "[33].

La littérature académique a introduit le terme de filtrage collaboratif par le système « Tapestry », qui a été développé en 1992 par le centre de recherche de "Xerox" aux Etats Unis, il s'agit d'un système de recommandation intégré a une application de mail électronique qui a permis aux utilisateurs de créer des requêtes permanentes, basées sur les

annotations (les tags) des utilisateurs. Quelques années plus tard, un certain nombre de systèmes académiques de recommandation ont vu le jour en 1994 et en 1995, tels que le système de recommandation d'articles d'actualités et de films développé par "GroupLens[3]" [34] et le système de recommandation de musique "Ringo". Ces deux systèmes sont également basés sur le filtrage collaboratif, des livres [34], des vidéos, des films, des pages Web, des articles de nouvelles Usenet et des liens Internet. Par la suite, avec l'essor de l'Internet et des applications Web, il y a eu un engouement pour les systèmes de recommandation qui se sont développés dans différents domaines d'applications. Nous pouvons en citer :

- Les systèmes de recommandation de films, tels que : Movielens et Eachmovie.
- Les systèmes de recommandation de livres (Bookcrossing5).
- Les systèmes de recommandation de musique (LastFM6).
- Les systèmes de recommandation d'articles d'actualités.
- Les systèmes de recommandation de blagues (Jester7).
- Les systèmes de recommandations introduits sur des sites e-commerce (Amazon).
- Les systèmes de recommandation de restaurants.
- Les systèmes de recommandation intégrés aux Extranets documentaires (l'Extranet documentaire du Crédit Agricole).
- Les systèmes de recommandations intégrés aux moteurs de recherche (le moteur de recherche d'AOL).
- Les systèmes de recommandations implémentés sur des sites de recrutement (Job-Finder).
- Les systèmes de recommandations de citations bibliographiques

II.3 Définition

Les systèmes de filtrage ou les systèmes de recommandation peuvent être définis de plusieurs façons, vue la diversité des classifications proposées pour ces systèmes, mais il existe une définition générale de Robin Burke [35] qui les définit comme suit : "Des systèmes capables de fournir des recommandations personnalisées permettant de guider l'utilisateur vers des ressources intéressantes et utiles au sein d'un espace de données important".

[3] est un laboratoire de recherche sur l' interaction homme-ordinateur du Département d'informatique et d'ingénierie de l' Université du Minnesota, Twin Cities, spécialisé dans les systèmes de recommandation et les communautés en ligne.

Il est défini aussi comme : un filtre de flux entrant d'information de façon personnalisée pour chaque acteur [36]. Autrement dit, dans un but 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 termes.

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

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

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

Pour réaliser le filtrage, le système de recommandation (SR) utilise les profils représentant des préférences relativement stables des utilisateurs pour calculer des recommandations. Ce calcul se fait par la prédiction des scores qu'un utilisateur est susceptible d'attribuer aux items. Le SR adapte ce profil au cours du temps en exploitant au mieux le retour de pertinence que les utilisateurs fournissent sur les items reçus. Par exemple, dans la figure II.1, la fonction de décision du système traite le flux entrant d'item pour suggérer à l'utilisateur, en consultant son profil, les items qu'il préfère. A son tour, l'utilisateur peut fournir des évaluations c'est-à-dire évaluer fréquemment les recommandations, pour que le système comprenne mieux ses besoins en information, et lui fournisse par conséquent de meilleures nouvelles recommandations.

Les trois parties suivantes constituent un système de recommandation :

- Les producteurs : Ce sont ceux qui vont permettre de faire les recommandations, ils "fourniront" les données (items).
- Le module de calcul : Il s'agit de l'algorithme en lui-même. En entrée il y a toutes les données que le système a besoin et la demande, en sortie les différentes recommandations.
- Le consommateur (utilisateur): C'est celui qui demande la recommandation.

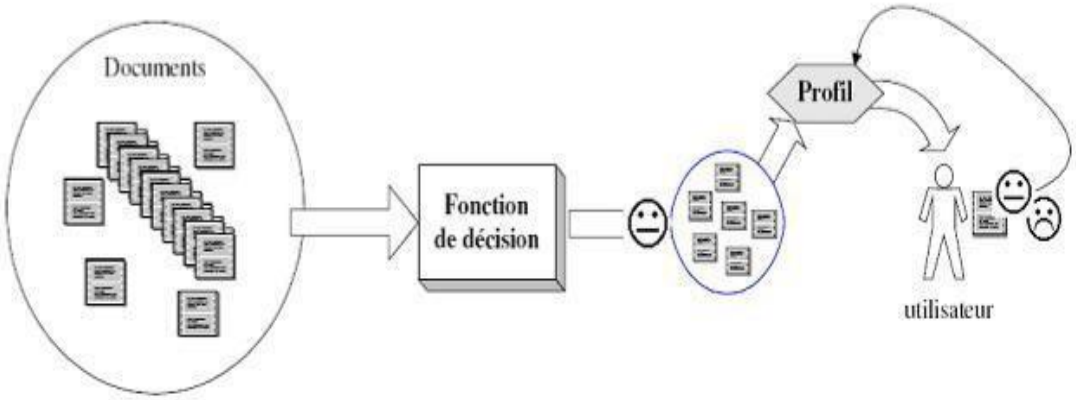


Figure II.1 Schéma général d'un système de filtrage d'information (document) [37].

II.4 Objectifs des systèmes de recommandation

Un système de recommandation a pour objectif de fournir à un utilisateur des ressources pertinentes en fonction de ses préférences. Ce dernier voit ainsi réduire son temps de recherche mais reçoit également des suggestions de la part du système auxquelles il n'aurait pas spontanément prêté attention. L'essor du Web et sa popularité ont notamment contribué à mise en place de tels systèmes comme dans le domaine du e-commerce. Citons par exemple les sites Web populaires Amazon dans le e-commerce mais également Cite-Seer[4], outil référençant des articles de recherche. Les systèmes de recommandation peuvent être vus initialement comme une réponse donnée aux utilisateurs ayant des difficultés à prendre une décision dans le cadre d'utilisation d'un système de recherche d'information "classique".

La recherche d'information, est fondée sur un principe d'indexation des données afin de répondre aux requêtes d'utilisateurs. Plus spécifiquement, la recherche documentaire, sous discipline de la recherche d'information, consiste à interroger une base

[4] Est un moteur de recherche et une librairie numérique pour les articles scientifiques et académiques avec une attention portée sur l'informatique.

documentaire par le bais de requêtes écrites en langues naturelles ou bien sous forme de mots clefs (nommées requêtes ad hoc). Un exemple bien connu d'outils utilisant les fondements de cette discipline sont les moteurs de recherche sur le Web comme Google ou Yahoo! où l'utilisateur formule son besoin en information via une requête (ensemble de mots clés). Ces mots clés sont alors comparés à l'ensemble des indexes des documents présents dans la base de données du moteur de recherche. Dès lors, l'utilisateur se voit retourner un ensemble de résultats plus ou moins pertinents par rapport à sa requête initiale.

Le problème inhérent à ce type de méthode (recherche d'information) résultats est la surcharge d'informations que l'utilisateur doit filtrer. Ainsi, la plupart du temps, un utilisateur se contente de consulter la première page de résultats d'un moteur de recherche en ignorant les autres, faisant l'impasse sur des contenus potentiellement pertinents. Le lien avec la recommandation est ici trivial, l'un des objectifs des systèmes de recommandation étant de faire le tri dans cette masse d'informations pour l'utilisateur de manière transparente.

La construction de profils d'utilisateurs constitue une première avancée dans le domaine de la recherche documentaire en termes de recommandation. En effet, comme le propose Google notamment, l'utilisateur se voit maintenant pondérer les résultats obtenus suite à sa requête en fonction des précédentes recherches qu'il a effectuées et des pages qu'il a consultées. Il s'agit d'un système d'identification de l'utilisateur permettant alors d'analyser ses actions. Cette approche peut être qualifiée d'analyse de traces et constitue l'un des fondements du filtrage d'informations sur lequel s'appuient les systèmes de recommandations.

C'est ainsi que l'on différencie les systèmes de recherche d'informations, pour lesquels la demande de l'utilisateur est de l'orienter et le guider vers des choix appropriés est explicites, et les systèmes de recommandation où la participation de l'utilisateur du système est non, induite notamment par des moteurs de traces.

II.5 Les classifications des systèmes de recommandation

Plusieurs approches et méthodes de classification des systèmes de recommandation ont été proposées. Parfois plusieurs terminologies sont utilisées pour désigner une même méthode ou approche. C'est pour cela que certains chercheurs se sont intéressés à la classification de ces méthodes et proposent une terminologie unifiée comme le montre la figure II.2 Nous citons ici quelques classifications :

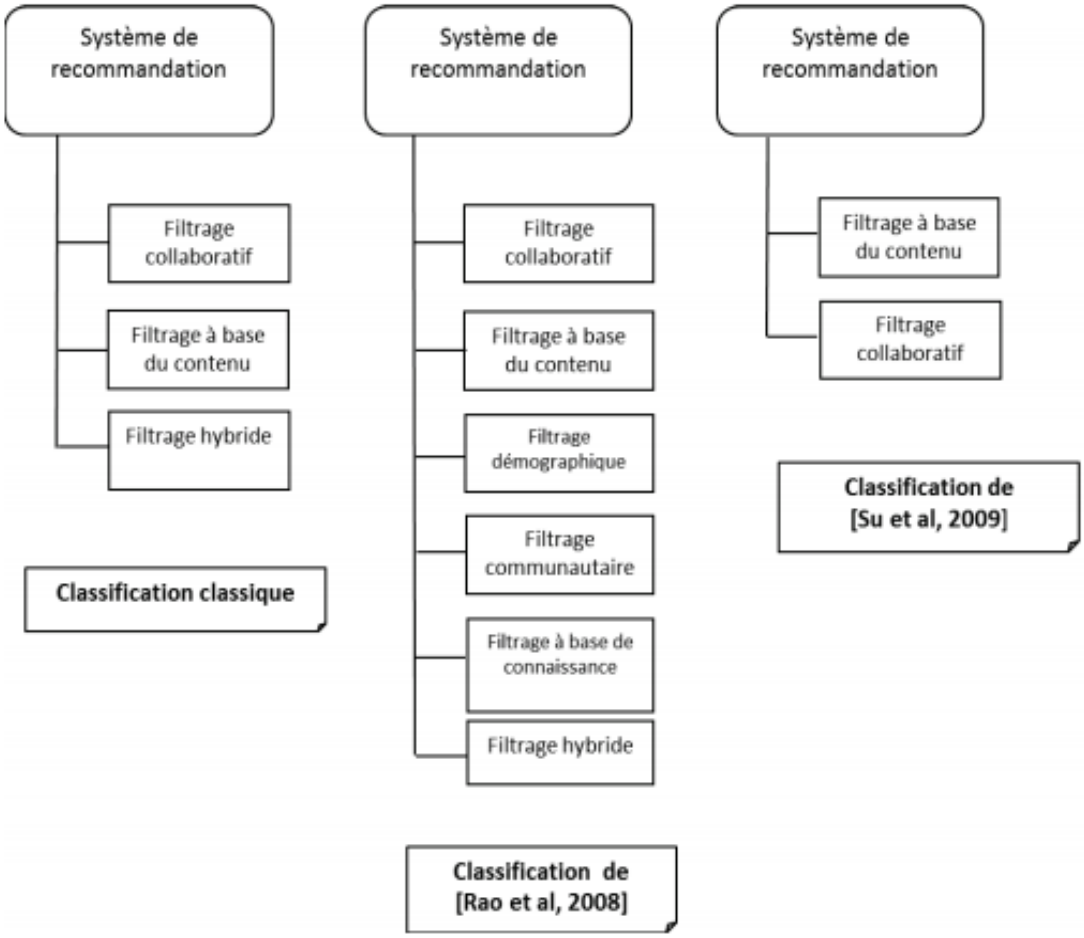


Figure II.2 Principales classifications des systèmes de recommandations [38].

II.5.1 La classification classique

Cette classification de [39], a fait un survol très intéressant pour rassembler les différents points de vue. D'ailleurs, de plus en plus de travaux se basent sur cette dernière.

Elle a identifié trois types de filtrage : le filtrage collaboratif, le filtrage basé sur le contenu et le filtrage hybride.

II.5.2 La classification de [40]

C'est une classification utilisée dans les systèmes collaboratifs. Les auteurs proposent une sous-classification qui comprend les techniques hybrides les classer dans les méthodes de collaboration hybrides. Su et al [40] classent le filtrage collaboratif en trois catégories :

- Approches de filtrage collaboratif à base de mémoire : pour les k plus proches voisins.

- Approches de filtrage collaboratif basé sur un modèle englobant une variété de techniques telles que : le clustering, les réseaux bayésiens, la factorisation de matrices, les processus de décision de Markov.
- Filtrage collaboratif hybride qui combine une technique de recommandation de filtrage collaboratif avec une ou plusieurs autres méthodes.

II.5.3 La classification de [41]

C'est une classification en fonction de la source d'information utilisée. Rao et Talwar ont répertorié un vaste ensemble de systèmes de recommandation pour différents domaines applicatifs, dans des contextes académiques et industriels.

Ils peuvent être classés de manière générale en six catégories en fonction de l'information utilisée (filtrage collaboratif, filtrage à base de contenu, filtrage hybride, filtrage démographique, filtrage communautaire, filtrage à base de connaissance).

II.6 Les types de recommandation

Après avoir donné les différentes classifications des techniques de recommandation nous présentons ci-dessous les techniques les plus importantes.

II.6.1 Recommandation basée sur le filtrage collaboratif

Elle s'appuie sur les appréciations données par un ensemble d'utilisateurs sur un ensemble d'items. Elle est basée sur le partage d'opinions entre les utilisateurs. Ces appréciations sont traduites en valeurs numériques.

Dans une recommandation basée sur le filtrage collaboratif, il faut que les utilisateurs fournissent des évaluations des items qu'ils ont déjà utilisés, sous forme de notes, pour constituer leurs profils. De tels systèmes ne tentent pas d'analyser ou de comprendre le contenu des items à recommander. La méthode consiste à faire des prévisions automatiques sur les intérêts d'un utilisateur en collectant des avis de nombreux utilisateurs. Il est très efficace au cas où le contenu des objets sont complexes, il est compliqué ou impossible de l'analyser, l'utilisateur peut apercevoir divers domaines intéressants, car le principe du filtrage collaboratif ne se fonde absolument pas sur la dimension thématique des profils, et n'est pas soumis à l'effet « entonnoir » (entonnoir est un type de filtrage incapable de recommander des documents qui sont différents de ceux que l'utilisateur a déjà vu et évalué).

On distingue deux types de recommandations basées sur le filtrage collaboratif qui consiste à comparer les utilisateurs entre eux et à retrouver ceux ayant des goûts en commun, les notes d'un utilisateur étant ensuite prédites selon son voisinage (utilisateurs similaires). La recommandation se référant aux items consiste à rapprocher les items appréciés par les mêmes personnes et à prédire les notes des utilisateurs en fonction des items les plus proches de ceux qu'ils ont déjà notés.

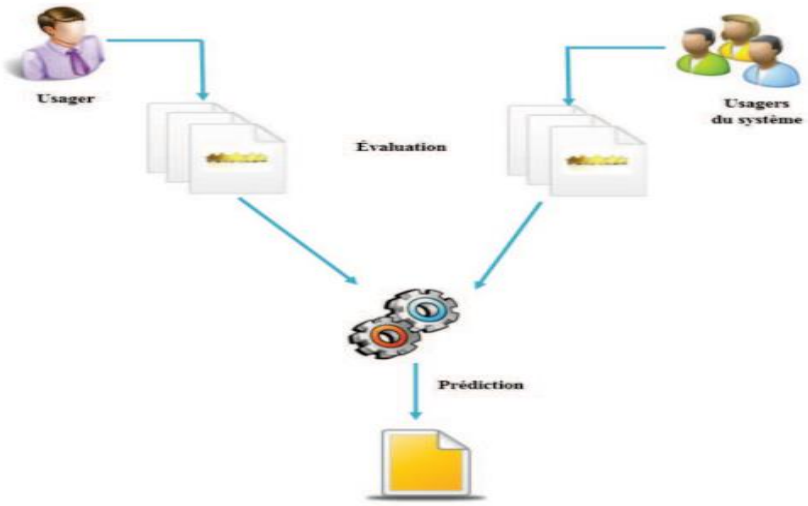


Figure II.3 Recommandation basée sur le filtrage collaboratif [43].

La figure II.4 représente un tableau de films avec sur un axe les utilisateurs d'un même système (ex : un groupe d'amis sur MovieLens) et sur un autre les films. Chaque cellule de la matrice contient l'avis donné par un utilisateur pour un film, la cellule vide signifie qu'il n'a pas d'avis particulier sur ce film. Par exemple, afin de prédire si « Illyes » apprécierait le film « Harry Potter » et probablement lui recommander ce film, on compare

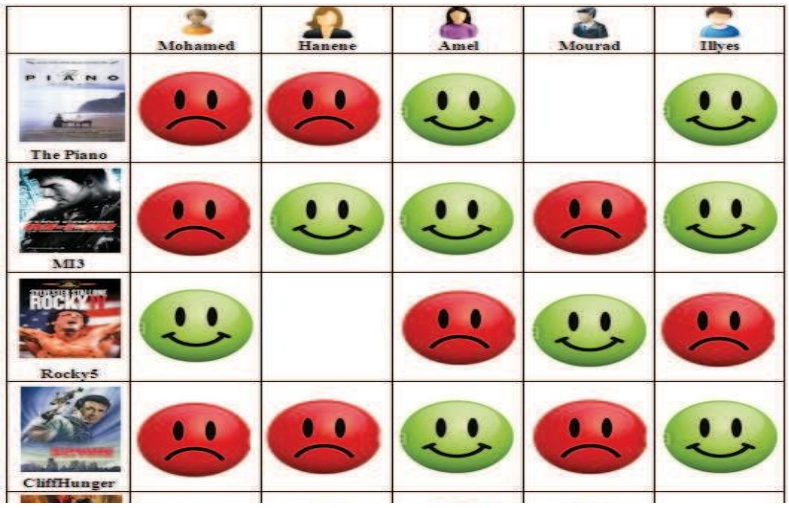


Figure II.4 Exemple de recommandation basé sur le filtrage collaboratif [43].

les votes de « Illyes » à ceux des autres utilisateurs choisis. On peut alors voir que « Illyes » et « Amel » ont des votes identiques, et que Amel n'a pas aimé le film «Harry Potter », on pourrait alors prédire que « Illyes » n'aimera pas aussi ce film et de ne lui pas faire cette suggestion.

Le filtrage collaboratif est très utilisé vu ses avantages, par les systèmes: Amazon, Netflix, MovieLens, Jester, Citeseer, Tapestry, Phoaks, etc. Parmi les avantages du filtrage collaboratif les jugements de valeur des utilisateurs intègrent non seulement la dimension thématique mais aussi d'autres facteurs relatifs à la qualité des items tels que la diversité, la nouveauté, l'adéquation du public visé, etc.

Un des problèmes des systèmes collaboratifs est que ses performances dépendent beaucoup de la distribution des évaluations (notes) données par utilisateurs. Dans le cas où il y a plusieurs items qui ont été utilisés et évalués par très peu d'utilisateurs, ces items seraient recommandés très rarement, même si ces utilisateurs ont donné des notes très hautes pour ces items. Ce problème est connu comme le problème de parcimonie (sparsity problem). De la même façon, si dans le système il existe des utilisateurs qui ont des goûts très différents en comparaison avec les autres, le système ne peut pas trouver des similarités entre utilisateurs et donc ne peut pas donner des bonnes recommandations.

L'exploitation des données disponibles dans un système de filtrage peut se faire de plusieurs manières. Ces méthodes sont classées en deux familles principales : les algorithmes basés sur la mémoire et les algorithmes basés sur le modèle.

II.6.1.1 Le filtrage collaboratif basé sur la mémoire

Le filtrage collaboratif basé sur la mémoire utilise une matrice des votes contenant des préférences des utilisateurs pour prédire des sujets additionnels ou des produits auxquels un nouvel utilisateur peut être intéressé. Son objectif est de prédire l'utilité des ressources (items) pour un utilisateur particulier (l'utilisateur actif) basé sur les votes des utilisateurs.

Ainsi, dans le filtrage basé sur la mémoire, les notes des utilisateurs stockées par le système sont directement utilisées pour prédire les notes pour de nouveaux items. Cela peut être fait de deux manières connues sous le terme de recommandations basées sur les utilisateurs ou recommandations basées sur les items.

a. Le filtrage collaboratif basé sur la mémoire (utilisateurs)

Les systèmes basés sur le voisinage utilisateur, évaluent l'intérêt d'un utilisateur pour un item en utilisant les notes de cet item. Ces notes sont données par d'autres utilisateurs, appelés voisins, qui ont des habitudes de notation similaires. Les plus proches voisins sont les utilisateurs les plus similaires dans leur notation.

En se basant sur le profil d'un utilisateur, le système recherche les utilisateurs qui lui sont les plus similaires. Les deux mesures de similarité qui sont très utilisées sont la similarité cosinus et corrélation de Pearson.

• La mesure de cosinus

Dans cette méthode les utilisateurs A et B sont considérés comme deux vecteurs de même origine dans un espace de n dimensions, n est égal au nombre d'items évalués par les deux utilisateurs. Plus deux utilisateurs sont similaires, plus l'angle entre leurs vecteurs respectifs est petit. Empiriquement, la similarité entre ces deux utilisateurs est calculée par la formule du Cosinus suivante :

$$Sim(A,B) = \frac{\sum_{i=1}^{n} v_{Ai} \times v_{Bi}}{\sqrt{\sum_{i=1}^{n} v_{Ai}^2} \times \sqrt{\sum_{j=1}^{n} v_{Bi}^2}} \qquad \text{(II.1)}$$

$\boldsymbol{n}$: Nombre d'items communs entre A et B votés par *v*

$\boldsymbol{v_{Ai}}$ **:** Vote de A pour l'item i

$\boldsymbol{v_{Bi}}$ **:** Vote de B pour l'item i

• Corrélation de Pearson

La corrélation de Pearson est une méthode issue des statistiques. Elle est aussi très utilisée dans le domaine des systèmes de recommandation pour mesurer la similarité entre deux utilisateurs. La formule suivante, présente le calcul de la similarité entre A et B avec cette mesure :

$$sim(A,B) = \frac{\sum_j (v_{Ai} - \overline{v}_{Ai})(v_{Bi} - \overline{v}_{Bi})}{\sqrt{\sum_j (v_{Ai} - \overline{v}_{Ai})(v_{Bi} - \overline{v}_{Bi})^2}} \qquad \text{(II.2)}$$

$\boldsymbol{i}$ *:* Nombre d'objets ayant voté à la fois par A et B.

$\boldsymbol{v_{Ai}}$ **:** Vote de A pour l'item i.

$\boldsymbol{\overline{v}_{Ai}}$: Moyenne des votes de A.

$\boldsymbol{v_{Bi}}$ **:** Vote de B pour l'item i.

$\boldsymbol{\overline{v}_{Bi}}$: Moyenne des votes de B.

Une fois que toutes les similarités de l'utilisateur cible A par rapport aux autres utilisateurs sont calculées et que les n utilisateurs les plus similaires qui constituent le voisinage de cet utilisateur cible sont définis, la prédiction de la valeur d'un item j évaluée par l'utilisateur A ($P_{A,j}$) est calculée à l'aide de la somme pondérée des estimations des voisins les plus proches qui ont déjà estimé l'item j :

$$P_{A,j} = \frac{\sum_{i=1}^{n} Sim(A,i) \times v_{i,j}}{\sum_{i=1}^{n} Sim(A,i)} \quad \text{(II.3)}$$

n : Nombre d'utilisateurs présents dans le voisinage de A, ayant déjà voté sur l'item j.
$v_{i,j}$: Vote de l'utilisateur i pour l'objet j.

b. Le filtrage collaboratif basé sur la mémoire (items)

Alors que les méthodes basées sur le voisinage utilisateur s'appuient sur l'avis d'utilisateurs partageant les mêmes idées pour prédire une note, les approches basées sur les items prédisent la note d'un utilisateur pour un item en se basant sur les notes pour des items similaires. Dans de telles approches, deux items sont similaires si plusieurs utilisateurs du système les ont notés d'une manière similaire.

Les choix possibles pour calculer la similarité Sim (i, j) entre les items « i » et « j » sont aussi la corrélation Pearson et la mesure de cosinus. La similarité entre deux items est calculée par la formule de :

- **La mesure de cosinus**

$$sim(i,j) = \frac{(v_{A,i} - \bar{v_A})}{\sqrt{\sum_{A=1}^{m} (v_{A,i} - \bar{v_A})^2}} \times \frac{(v_{Aj} - \bar{v}_A)}{\sqrt{\sum_{A=1}^{m} (v_{Aj} - \bar{v}_A)^2}} \quad \text{(II.4)}$$

m : nombre d'utilisateurs qui ont votés pour les deux items
$\boldsymbol{v_{Ai}}$ **:** Vote de A pour l'item *i*
$\boldsymbol{v_{Aj}}$ **:** Vote de A pour l'item *j*
$\bar{v}_{Ai}$: Moyenne des votes de A

- **Corrélation de Pearson**

$$sim(A,B) = \frac{\sum_{A=1}^{m} (v_{A,i} - \bar{v}_A) \times (v_{Bj} - \bar{v}_A)}{\sum_{A=1}^{m} (v_{A,i} - \bar{v}_A)^2 \times \sum_{A=1}^{m} (v_{Bj} - \bar{v}_A)^2} \quad \text{(II.5)}$$

m : Nombre d'utilisateur présents en communs entre A et B votés par *v*

$\boldsymbol{v_{Aj}}$ **:** Vote de A pour l'item *j*

$\boldsymbol{v_{Bj}}$ **:** Vote de B pour l'item j

$\boldsymbol{\bar{v}_{Ai}}$: Moyenne des votes de A

$\boldsymbol{\bar{v}_{Bi}}$: Moyenne des votes de B

Une fois que la similarité parmi les items ait été calculée, la prochaine étape est de prévoir pour l'utilisateur cible A, une valeur pour l'item actif i. Une manière commune est de capturer comment l'utilisateur a évalué les items similaires. La valeur prévue est basée sur la somme pondérée des estimations de l'utilisateur ainsi que les déviations des estimations moyennes et peut être calculée à l'aide de la formule suivante :

$$P_{A,i} = \bar{v}_i + \frac{\sum_{j=1}^{m} Sim(A,B) \times (v_{A,j} - \bar{v}_j)}{\sum_{j=1}^{m} |Sim(i,j)|} \quad \text{(II.6)}$$

m : nombre d'items présents dans le voisinage de l'item *i*, ayant déjà été voté par l'utilisateur A.

$v_{A,j}$: vote de l'utilisateur A pour l'objet j.

$\bar{v}_j$: Moyenne des votes pour l'item j.

$|Sim(i,j)|$: Similarité moyenne.

II.6.1.2 Le filtrage collaboratif basé sur le modèle

Ce type d'algorithme est comme son nom l'indique est basé sur des modèles, supposés réduire la complexité. Ces modèles utilisent la base de données des évaluations des utilisateurs pour estimer ou apprendre un modèle qui est alors utilisé pour les prédictions. Ils peuvent être basés sur des classificateurs permettant de créer des classes pour réduire la complexité.

II.6.2 Recommandation basée sur le contenu

Les systèmes de recommandation basés sur le contenu s'appuient sur des évaluations effectuées par un utilisateur sur un ensemble de documents ou items. L'objectif est alors de comprendre les motivations l'ayant conduit à juger comme pertinent ou non un item donné. Il peut être considéré comme un système de recherche d'information exploitant le profil d'utilisateur.

L'avantage des systèmes de recommandation basés sur le contenu est qu'ils permettent d'associer des items à un profil utilisateur. Notamment, en utilisant des

techniques d'intelligence artificielle. L'utilisateur est indépendant des autres ce qui lui permet d'avoir des recommandations même s'il est le seul utilisateur du système.

Cependant, ce type de systèmes présente certaines limitations :

- L'effet « entonnoir » : les besoins de l'utilisateur sont de plus en plus spécifiques, ce qui l'empêche d'avoir une diversité de sujets. Même pire, 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.
- Problème de démarrage à froid : Un nouvel utilisateur du système éprouve des difficultés à exprimer son profil en spécifiant des thèmes qui l'intéressent

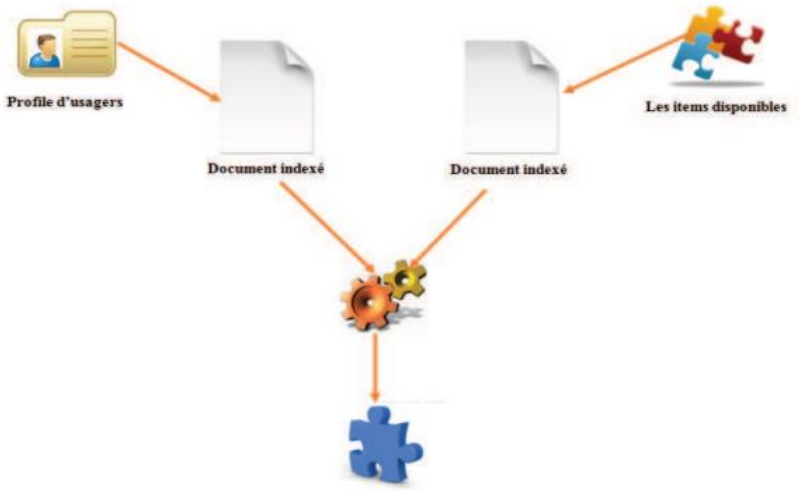


Figure II.5 Recommandation basée sur le contenu [44].

On distingue deux types de recommandation basé sur le contenu : recommandation basé sur les mots clefs et recommandation basé sur la sémantique :

II.6.2.1 Recommandation basée sur les mots clefs

La technique de recommandation basée sur le contenu peut être appliquée à la recommandation de pages Web, de films, d'articles actualités, de restaurants, etc. Si nous prenons l'exemple d'un système de recommandation d'articles scientifiques basé sur le contenu, lorsqu'un utilisateur a tendance à consulter souvent des articles portant sur le domaine de la génétique, le système lui proposera des recommandations liées à la génétique. En effet, ces articles disposent de mots-clés communs tels que : "ADN", "gène" ou "protéine".

II.6.2.2 Recommandation basée sur la sémantique

La sémantique a été introduite par plusieurs méthodes dans le processus de recommandation. Ces méthodes sont abordées en tenant compte de plusieurs critères :

- Le type de source de connaissance impliquée (lexique, ontologie…).
- Les techniques adoptées pour l'annotation ou la représentation d'items.
- Le type de contenu inclus dans le profil utilisateur.
- La stratégie de correspondance entre items et profil.

Les systèmes de recommandation basés sur la sémantique évoluent au rythme des méthodes et outils proposés dans le domaine du Web sémantique.

II.6.3 Le filtrage hybride

Un système de recommandation est dit hybride quand il combine deux ou plusieurs approches de recommandation différentes. La recommandation basée sur le contenu et la recommandation collaborative ont souvent été considérées comme complémentaires.

En général, l'hybridation, afin de traiter les insuffisances de chaque technique utilisée seule et profiter de leurs points forts, s'effectue en deux phases :

1. Appliquer séparément le filtrage collaboratif et autres techniques de filtrage pour générer des recommandations candidates.
2. Combiner ces ensembles de recommandations préliminaires selon ces méthodes : la pondération, la mixtion, la cascade, la commutation, Combinaison et augmentation de caractéristiques et Méta niveau. Afin de produire les recommandations finales pour les utilisateurs.

Plus généralement, les systèmes hybrides gèrent des profils d'utilisateurs orientés contenu, et la comparaison entre ces profils donne lieu à la formation de communautés d'utilisateurs permettant le filtrage collaboratif.

II.6.4 Recommandation démographique

C'est une recommandation simple proposant des items par rapport au profil démographique de l'utilisateur. Elle consiste à regrouper les utilisateurs en plusieurs classes par rapport aux informations démographiques telles que l'âge, la profession, la localisation, la langue, le pays, le sexe comme sur la figure II.6 ou l'item : revue de cuisine a été recommandée par une femme aux utilisateurs de même sexe.

Le principe de cette approche est que les utilisateurs ayant évolué dans un environnement similaire partagent des goûts communs que ceux ayant évolué dans des environnements différents.

De nombreux sites utilisent cette solution simple à proposer une offre de contenu « personnalisé ». Par exemple, les utilisateurs sont redirigés vers un site Web particulier en fonction de leur langue ou de pays.

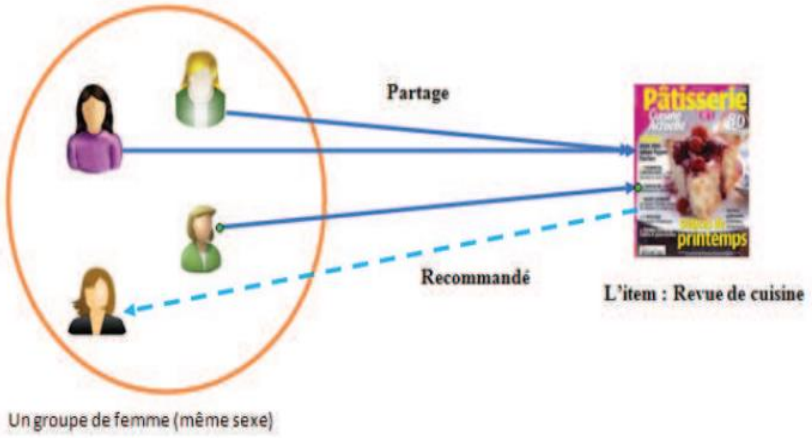


Figure II.6 Recommandation démographique [45].

II.6.5 Recommandations à base de connaissances

La recommandation basée sur la connaissance consiste à collecter le maximum d'informations sur un utilisateur pour pouvoir ensuite lui recommander des items. Une analogie avec la vie réelle serait par exemple une recommandation faite par un ami qui nous connaît bien et se serait basé sur des informations précises nous concernant, plutôt que sur nos préférences. En d'autres termes les systèmes de recommandation basés sur la connaissance (knowledge-based) se basent sur la connaissance explicite de l'utilisateur. Par exemple, pour un système de recommandation pour la vente d'appartements ou de voitures, l'utilisateur peut spécifier explicitement ses préférences (ex. « le prix maximum de la voiture est x », « la consommation moyenne ne doit pas dépasser y »).

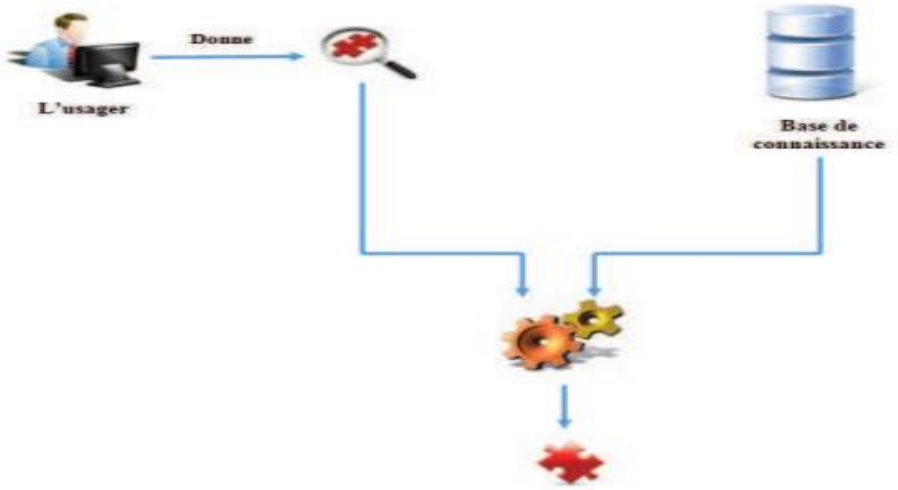


Figure II.7 Recommandation à base de connaissance [46].

Il existe deux sous types de recommandation à base de connaissance :

II.6.5.1 Le raisonnement à base des cas

Le raisonnement à base des cas tire parti de la régularité du monde réel afin de résoudre des problèmes en recherchant la solution d'un cas semblable rencontré et résolu dans le passé. Il s'appuie sur des expériences décrites dans des formats complètement structurés tels que des objets ou des enregistrements de base de données [42].

Cette approche est utilisée dans les systèmes de recommandation, ils estiment combien les besoins ou les préférences (description de problème) de l'utilisateur correspondent aux recommandations possibles (solutions du problème).

Il a été avancé que le raisonnement basé sur des cas est non seulement une méthode puissante pour le raisonnement informatique, mais également un comportement omniprésent dans la résolution de problèmes humains quotidiens ; ou, plus radicalement, que tout raisonnement est basé sur des cas passés vécus personnellement.

II.6.5.2 Le raisonnement à base de contraintes

Une recommandation à base de contraintes est un autre type de systèmes à base de connaissances. Elle exploite des bases de connaissances prédéfinies qui contiennent des règles explicites sur la façon de relier les exigences des clients avec des fonctionnalités d'item. Par exemple, un utilisateur peut être intéressé à acheter des produits avec un certain ensemble de caractéristiques et dans une gamme de prix spécifique.

II.6.6 Recommandation communautaire

La recommandation communautaire ou comme on l'appelle souvent recommandation sociale, vu que la plupart des réseaux sociaux s'appuient sur cette classification dans leurs recommandations. L'idée de base est donc de dire que si des utilisateurs ont partagé les mêmes intérêts dans le passé, il y a de fortes chances qu'ils partagent aussi les mêmes goûts dans le futur. Par exemple sur la figure.II.8**,** trois utilisateurs ont partagés un item au passé et le quatrième utilisateur a partagé un item de même intérêt, cet item sera recommandé aux trois premiers utilisateurs car il y'a une forte probabilité qu'ils partagent aussi les mêmes goûts sur des items.

Figure II.8 Recommandation communautaire [47].

II.7 Avantages et inconvénients des méthodes des systèmes de recommandation

Le tableau ci-dessus résume les forces et faiblesses des méthodes traditionnelles utilisées par les systèmes de recommandation, en l'occurrence le filtrage collaboratif, le filtrage démographique, le filtrage à base de contenu, et le filtrage à base de données communautaires.

Techniques	Avantages	Inconvénients
Filtrage Démographique	- Il n'exige aucun historique d'estimations.	- Problème deconfidentialité. - Utilisateur avec un gout. - Nouvel Item.
Filtrage À base de données communautaire	- Adaptabilité : la qualité croit avec le nombre d'amis.	- Nouvel utilisateur. - Nouvel item.
Filtrage à base du contenu	- Pas besoin d'une large communauté d'utilisateurs pour pouvoir effectuer des recommandations. - Une liste de commandassions peut être générée même s'il n'y a qu'un seul utilisateur. - La qualité croit avec le temps. - Pas besoin d'information sur les autres utilisateurs. - Prendre en considération les gouts uniques des utilisateurs.	- L'analyse du contenu nécessaire pour faire une recommandation. - Problème de recommandation des images et des vidéos. - Nécessité du profil d'utilisateur.
Filtrage collaboratif	-Ne demande aucune connaissance sur le contenu de l'item ni sa sémantique. -La qualité de la recommandation peut être évaluée -plus le nombre d'utilisateur est grand plus la recommandation est meilleure	-Démarrage à froid - Nouvel Item - Nouvel utilisateur - Problème de confidentialité La complexité : dans les systèmes avec un grand nombre d'item et d'utilisateurs, le calcul croit linéairement.

Tableau II.1 Avantages et inconvénients des méthodes de recommandation.

II.8 Présentation de quelques travaux de l'état de l'art basé sur le machine et deep learning

Nous présentons dans cette partie quelques travaux de l'état de l'art basé sur le machine et deep learning.

II.8.1 Filtrage collaboratif utilisant k-voisins les plus proches (KNN)

Les algorithmes de filtrage collaboratif basés sur les voisins utilisent généralement la totalité de la matrice des notes des utilisateurs pour faire la recommandation. Pour utiliser l'algorithme de K plus proche voisins, le système doit avoir une mesure de similarité pour faire la distinction entre des utilisateurs qui sont près et ceux qui sont éloignés. Cette mesure de similarité peut être issue de la distance euclidienne, la mesure du cosinus, la corrélation de Pearson, etc. Ensuite, lorsque le système est interrogé sur une nouvelle information, il va aller trouver les k utilisateurs qui se trouvent le plus près de l'utilisateur cible. Finalement, un vote majoritaire se fait pour décider quel item sera recommandé (filtrage basé sur les utilisateurs).

- **Calcul de la similarité**

Le calcul de la similarité a pour objectif de déterminer dans quelle mesure deux utilisateurs sont similaires. Il existe plusieurs façons de calculer cette similarité, cependant les méthodes les plus utilisées sont :

- La mesure de cosinus (voir formule II.1).
- Corrélation de Pearson (voir formule II.2).

Une fois que la similarité a été calculée, on calcule la prédiction (voir formule II.3) a fin de prédire la note de l'utilisateur cible pour un item.

II.8.2 Perceptron multicouches

Un perceptron multicouche (multilayer perceptron ou MLP) est un perceptron qui s'associe à des perceptrons supplémentaires, empilés en plusieurs couches, pour résoudre des problèmes complexes.

Dans le cas des systèmes de recommandation, les perceptrons multicouches sont utilisés pour encoder les utilisateurs et items séparément avant de les combiner avec un autre perceptron multicouches. Ils modélisent les interactions entre les utilisateurs et les items, et

sélectionnent les items les plus représentatifs pour chaque utilisateur et les attributs d'item les plus importants pour chaque utilisateur.

Pour modéliser l'interaction utilisateur-item, on utilise une représentation multicouche, où la sortie d'une couche sert d'entrée à la couche suivante. La première couche d'entrée a deux vecteurs d'entrée V_u et V_i qui représentent l'utilisateur *u* et l'item *i*. Après la couche d'entrée, il y a une couche d'intégration. Cette couche est entièrement connectée, qui projette la représentation séparée sur un vecteur dense. Ces couches d'intégration sont ensuite introduites dans une architecture neuronale multicouche pour mapper les vecteurs latents à des scores de prédiction. Nous pouvons également personnaliser chaque couche cachée pour découvrir de nouvelles structures latentes à partir des interactions utilisateur-item. La dernière couche donne le score prévu [48].

II.8.3 Système de recommandation en utilisant le classement personnalisé bayésien (BPR)

Lorsque les utilisateurs achètent en ligne, ils ne parcourent généralement que les premières pages des sites Web. Par ailleurs, de plus en plus de personnes utilisent des tablettes et des mobiles pour faire leurs achats en ligne, il est donc indispensable de faire un classement personnalisé.

Lorsqu'un utilisateur clique sur un item pour voir ses détails, cela signifie généralement que certaines caractéristiques de l'item sont attrayantes. Ainsi, on peut supposer que les utilisateurs préfèrent ces items aux autres et les utiliser pour les classer après qu'un utilisateur a parcouru le site Web pendant un certain temps. La tâche principale du classement personnalisé est de fournir à un utilisateur une liste classée d'items [49].

Soit « U » l'ensemble de tous les utilisateurs et « I » l'ensemble de tous les items. La figure II.9 ci-dessous montre comment les données implicites sont traitées dans le cas de recommandations d'items.

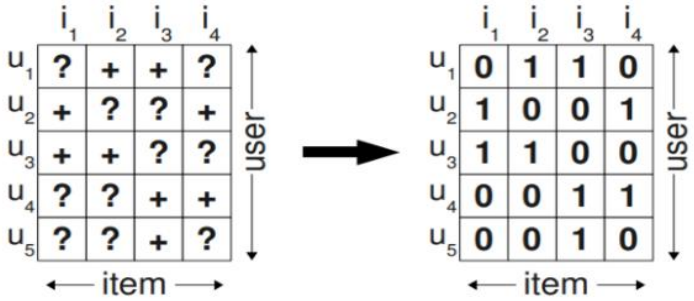


Figure II.9 les interactions existantes entre l'utilisateur et l'item [49].

L'approche habituelle consiste à prédire un score X_{ui} personnalisé pour un item qui reflète la préférence de l'utilisateur pour l'item. Après cela, les items seront classés en fonction de ce score. Ici, comme vous pouvez le voir dans la figure ci-dessus, toutes les interactions existantes entre l'utilisateur et l'item sont étiquetées comme une classe positive (1) et le reste des interactions est étiqueté comme une classe négative (0).

Dans l'approche BPR [49], au lieu de prendre un item, les paires d'items sont considérées comme des données d'entraînement. L'optimisation serait effectuée en fonction du classement de ces paires utilisateur-item au lieu de noter uniquement l'interaction utilisateur-item. L'ensemble de données qui serait considéré est formulé comme suit :

$$(u, i, j) \in D_s$$

La sémantique de $(u, i, j) \in D_s$ est que l'utilisateur u est supposé préférer i à j.

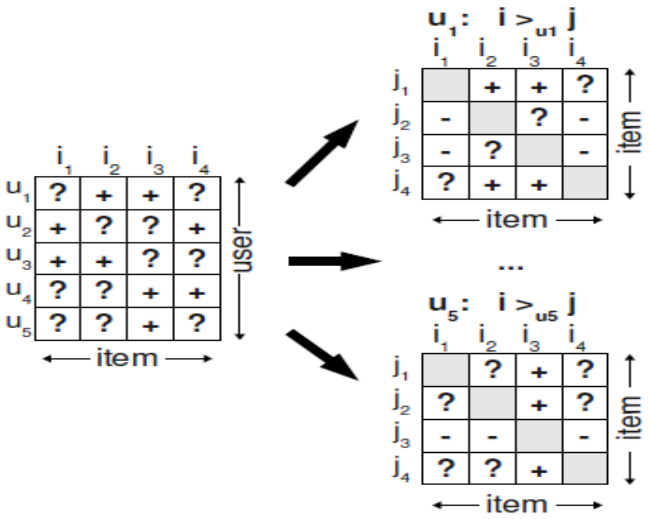


Figure II.10 les préférences par paire spécifiques à l'utilisateur entre une paire d'items [49].

Dans la figure II.10 ci-dessus, l'utilisateur u_1 a vu l'item i_2 mais pas l'item i_1, de sorte que l'algorithme suppose que cet utilisateur préfère l'item i_2 à i_1 ($i_2 > i_1$) et donne un signe positif. Aucune inférence ne peut être faite sur une préférence pour les items qui ont tous deux été vus par un utilisateur et qui sont affichés comme « ? ». Il en va de même pour deux items qu'un utilisateur n'a pas encore vus (par exemple, l'item i_1 et i_4 pour l'utilisateur u_1). Au contraire, vous pouvez observer un signe négatif pour (i_1, j_2) car l'utilisateur préfère item 2 (i_2) à item 1 (i_1).

Les auteurs [49] proposent que BPR se compose du critère d'optimisation BPR-OPT et de l'algorithme LearnBPR pour l'optimisation. Plutôt que de prédire une note précise

pour chaque item, nous n'avons qu'à prédire les préférences relatives de l'utilisateur pour toutes les paires (*i*, *j*).

BPR-OPT :

$$\sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda\|\Theta\|^2 \quad \text{(II.7)}$$

Ou σ est la Fonction logistique sigmoïde :

$$\sigma(x) := \frac{1}{1+e^{-x}} \quad \text{(II.8)}$$

$\hat{x}_{uij}(\Theta)$ dans l'équation ci-dessus est une fonction à valeur réelle qui représente la relation entre l'utilisateur u, l'item i et l'item j et est généralement calculée en utilisant le modèle de factorisation matricielle.

Θ représente le vecteur de paramètres d'une classe de modèle arbitraire (par exemple, factorisation matricielle).

$\lambda\ \Theta$ sont des paramètres de régularisation spécifiques au modèle

Nous pouvons formuler la probabilité individuelle qu'un utilisateur préfère l'item i à l'item j comme suit :

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta)) \quad \text{(II.9)}$$

Ici,> u est la structure de préférence souhaitée mais latente pour l'utilisateur u. Il est également important de noter que p (> u | θ) est une fonction de vraisemblance spécifique à l'utilisateur.

Fonction de vraisemblance : est une fonction des paramètres d'un modèle statistique calculée à partir de données observées.

$$\text{P}\ (\Theta| > \text{u})\ \alpha\ \text{p}\ (> \text{u}\ |\Theta)\ \text{p}\ (\Theta) \quad \text{(II.10)}$$

Nous devons donc maximiser le nombre de prédictions correctes de toutes les paires (i,j) , ce qui équivaut à maximiser l' AUC qui est l'aire sous la courbe ROC.

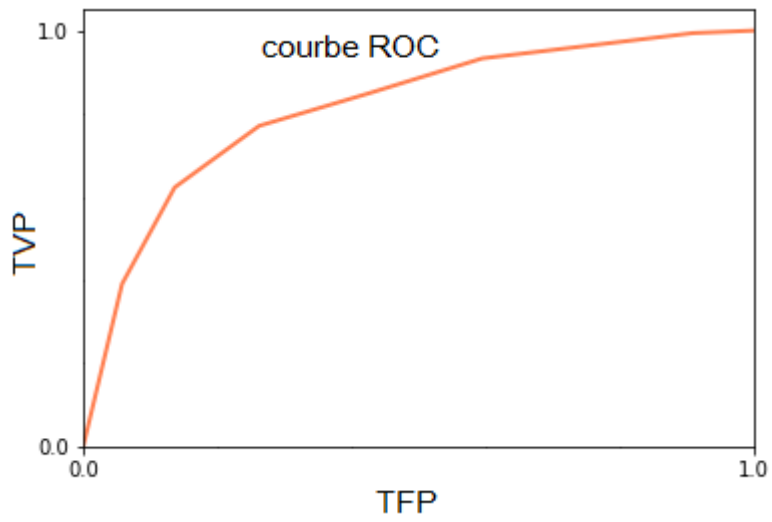


Figure II.11 la courbe ROC [50].

Une courbe ROC **(**receiver operating characteristic**)** est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification. Cette courbe trace le taux de vrais positifs en fonction du taux de faux positifs :

Le taux de vrais positifs (TVP) est l'équivalent du rappel. Il est donc défini comme suit :

$$TVP = \frac{VP}{VP+FN} \tag{II.11}$$

Le taux de faux positifs (TFP) est défini comme suit :

$$TFP = \frac{FP}{FP+VN} \tag{II.12}$$

Analogies à l'optimisation AUC (Area Under The Curve)

L'équation ci-dessus est la formule de l'AUC, où I est l'ensemble de tous les items, et I_u^+ est l'ensemble des items sur lesquels l'utilisateur a cliqué :

$$\text{AUC}= \frac{1}{|U|}\sum_{u\in U}\frac{1}{|I_u^+||I/I_u^+|}\sum_{i\in I_u^+}\sum_{j\in|I/I_u^+|}\delta(\widehat{x_{uiJ}}>0) \tag{II.13}$$

L'AUC, c'est l'aire sous la courbe ROC qui est tracée avec un taux vrai positif comme un axe y et un taux faux positif comme axe x pour différents seuils du modèle qui est considéré. En d'autres termes, pour un ensemble de recommandations d'items, nous pouvons tracer la courbe ROC en calculant le pourcentage de bonnes recommandations d'articles et le pourcentage de mauvaises recommandations pour chaque seuil.

II.8.4 Le filtrage collaboratif neuronal (NCF*)*

La factorisation matricielle en anglais factorization matrix *(*MF*)* pour les systèmes de recommandation est une des méthodes utilisées par les services en ligne comme Netflix afin d'accélérer la recherche de recommandations de contenu pour les utilisateurs

NCF abréviation de Neural Collaborative Filtering, a tenté d'atteindre les objectifs suivants :

- NCF essaie d'exprimer et de généraliser la factorisation matricielle (MF) dans son cadre.
- NCF essaie d'apprendre les interactions utilisateur-item via un perceptron multicouche.

Le filtrage collaboratif neuronal utilise MF pour identifier la relation entre les items et les utilisateurs. Les évaluations des utilisateurs sur les items à l'entrée aident à prédire comment les utilisateurs évalueraient les items afin que les utilisateurs puissent obtenir la recommandation basée sur la prédiction.

Le filtrage collaboratif neuronal (NCF) vise à concevoir une meilleure fonction d'interaction dédiée pour modéliser l'interaction des fonctionnalités latentes entre les utilisateurs et les items en :

1. Modélisant l'interaction des fonctionnalités utilisateur-item via l'architecture de réseau neuronal en utilisant un Perceptron multicouche (MLP). Il s'agit d'une mise à niveau par rapport à MF car MLP peut (théoriquement) apprendre n'importe quelle fonction continue et présente un niveau élevé de non-linéarités (en raison de plusieurs couches).
2. Généralisant et exprimant la MF comme un cas particulier de NCF. MF connaît un grand succès dans le domaine des recommandations, cela donnera plus de crédibilité à NCF.

II.8.5 Le filtrage collaboratif basé sur un auto-encodeur (ACF)

Les auto-encodeurs sont une technique d'apprentissage non supervisée dans laquelle nous exploitons les réseaux de neurones pour la tâche d'apprentissage de la représentation en tentant de reconstruire ses données d'entrée dans la couche de sortie. Ils se composent d'une couche d'entrée, d'une couche cachée et d'une couche de sortie. Les données d'entrée sont transmises à la couche d'entrée. La couche d'entrée et la couche cachée construisent un

encodeur. La couche cachée et la couche de sortie construisent un décodeur. Les données de sortie sortent de la couche de sortie [51].

Il existe deux façons générales d'appliquer l'auto-encodeur au système de recommandation :

- Utiliser l'auto-encodeur pour apprendre des représentations d'entités de dimension inférieure à la couche de goulot d'étranglement
- Remplir les blancs de la matrice d'interaction directement dans la couche de reconstruction. Presque toutes les variantes d'autoencodeur peuvent être appliquées à la tâche de recommandation.

Il existe de nombreuses variantes de codeurs automatiques actuellement utilisés dans les systèmes de recommandation. Les quatre plus courants sont :

a) Autoencodeur de débruitage (Le Denoising Autoencoder DAE)

Corrompt les entrées avant de les mapper dans la représentation masquée, puis reconstruit l'entrée d'origine à partir de sa version corrompue. L'idée est de forcer la couche cachée à acquérir des fonctionnalités plus robustes et d'empêcher le réseau d'apprendre simplement la fonction d'identité.

b) Auto-encodeurs à réduction de bruit empilés (Stacked Denoising Autoencoder SDAE)

Empile plusieurs auto-encodeurs de débruitage les uns sur les autres pour obtenir des représentations de plus haut niveau des entrées. La formation est généralement optimisée avec des algorithmes gourmands, allant couche par couche. Les inconvénients apparents ici sont le coût de calcul élevé de la formation et le manque d'évolutivité pour les fonctionnalités de grande dimension.

c) Autoencodeurs de réduction du bruit marginalisés (Marginalized Denoising Autoencoder MDAE)

Évite le coût de calcul élevé du SDAE en marginalisant la corruption des fonctionnalités stochastiques. Ainsi, il a une vitesse d'entraînement rapide, une mise en œuvre simple et une évolutivité vers des données de grande dimension.

d) Les auto-encodeurs variationnels (Variational Autoencoder VAE)

Est un modèle de variable latente non supervisé qui apprend une représentation approfondie à partir de données de grande dimension. L'idée est d'encoder l'entrée comme une distribution de probabilité plutôt qu'une estimation ponctuelle comme dans l'auto-encodeur vanille. Ensuite, VAE utilise un décodeur pour reconstruire l'entrée d'origine en utilisant des échantillons de cette distribution de probabilité.

AutoRec :

AutoRec L'un des premiers modèles qui considèrent le problème du filtrage collaboratif du point de vue de l'auto-encodeur.

AutoRec prend en entrée les vecteurs $r^{(u)}$ de l'utilisateur ou les vecteurs d'item $r^{(i)}$ et vise à les reconstruire dans la couche de sortie, il a deux variantes : AutoRec basé sur les items (I-AutoRec) et AutoRec basé sur l'utilisateur (U-AutoRec).

NCF est une extension d'AutoRec et possède les deux avantages suivants :

- Il déploie les techniques de débruitage, ce qui rend NCF plus robuste ;
- Il incorpore les informations secondaires telles que les profils d'utilisateurs et les descriptions d'articles pour atténuer la rareté et les influences de démarrage à froid.

II.9 Conclusion

Les systèmes de recommandation sont une nouvelle technique puissante et deviennent rapidement un outil crucial notamment pour le commerce électronique sur le Web.

Dans ce chapitre, nous avons présenté les systèmes de recommandation qui sont devenus omniprésents ces dernières années dans de nombreux domaines. Ces systèmes sont conçus pour aider les utilisateurs à trouver des ressources qui les intéressent et qui sont adaptées à leurs préférences, parmi le nombre important des choix qui s'offrent à eux.

Nous avons d'abord défini la notion des systèmes de recommandation. Ensuite, Nous avons présenté ses composants principaux qui sont les utilisateurs et les items. Basé sur les préférences d'utilisateur et l'exploitation du profil d'utilisateur, un système de recommandation peut proposer des items aux utilisateurs.

Plus précisément, dans ce chapitre nous avons passé en revue les techniques (approches) de recommandations existantes : le filtrage collaboratif, le filtrage à base de contenu, le filtrage contextuel, le filtrage hybride. Nous avons aussi discuté d'un ensemble

de travaux de l'état de l'art qui exploitent les techniques de machine et deep learning dans les systèmes de recommandation.

Dans le prochain chapitre nous décrivons l'implémentation, l'évaluation et la comparaison de trois algorithmes de recommandation. Deux de ces algorithmes sont basés sur le machine learning qui sont les modèles k-voisins les plus proches (KNN) et le classement personnalisé bayésien (BPR) ; et un modèle basé sur le deep learning qui est le modèle de filtrage collaboratif neuronal NCF.

Chapitre III

III.1 Introduction

Nous avons vu dans le premier chapitre l'apprentissage profond et les réseaux de neurones, dans le second chapitre les systèmes de recommandation et quelques travaux de l'état de l'art des systèmes de recommandation basés sur le machine et deep learning.

L'objectif de ce dernier chapitre est de présenter l'implémentation, l'évaluation et la comparaison de trois algorithmes de l'état de l'art. Les expérimentations ont été réalisées sur une collection de test « MovieLens », plusieurs mesures d'évaluation ont été utilisées pour évaluer les différents algorithmes.

Ce chapitre est structuré comme suit, en premier lieu nous présentons le jeu de données utilisé pour tester les algorithmes implémentés, ensuite nous présentons les différentes métriques d'évaluation utilisées pour évaluer les algorithmes, une description succincte des algorithmes est présentée. Á la fin de ce chapitre nous présentons les résultats obtenus avec les métriques pour les différents algorithmes et nous discutons ces résultats.

III.2 Collection et mesures d'évaluations

III.2.1 Collection MovieLens

MovieLens, c'est l'un des plus populaires ensemble de données qui a été créé en 1997 par GroupLens Research, un laboratoire de recherche sur les interactions homme-machine du Département d'Informatique et d'Ingénierie de l'Université du Minnesota, afin de recueillir des données de recherche sur des recommandations personnalisées. Il fonde ses recommandations sur les informations fournies par les utilisateurs du site Web. Les classements dans MovieLens peuvent survenir à tout moment, en fait, cela peut arriver des années plus tard après avoir regardé un film. Les utilisateurs saisissaient souvent de nombreuses évaluations à la fois dans l'espoir d'obtenir des recommandations plus personnalisées ou simplement pour leur satisfaction [52].

Dans notre travail nous avons utilisé un ensemble de données MovieLens 100K, 100 000 évaluations (note de 1 à 5) de 1000 utilisateurs sur 1700 films. Chaque utilisateur a évalué au moins 20 films. Les données ont été collectées via le site Web MovieLens pendant la période du 19 septembre 1997 au 22 avril 1998. Ces données ont été nettoyées (les utilisateurs avec moins de 20 évaluations ou sont données démographiques complètes, leurs informations ont été supprimées de cet ensemble de données).

III.2.2 Les mesures d'évaluation utilisées

Les mesures d'évaluation mesurent la capacité de l'algorithme de recommandation à prédire correctement les évaluations connues des utilisateurs. Elles utilisent des évaluations correctes d'un ensemble de donnée de test et les comparent avec des prédictions proposées par l'algorithme de recommandation en se basant sur un ensemble de donnée d'apprentissage. La majorité de ces métriques proviennent du domaine de la recherche d'information.

Nous présentons ci-dessous les métriques que nous avons utilisées**.**

III.2.2.2 Précision et rappel

La précision et le rappel sont deux métriques qui, ensemble, sont utilisés pour évaluer les performances des systèmes de classification ou de recherche d'informations. La précision est définie comme la fraction d'instances pertinentes parmi toutes les instances récupérées. Le rappel, parfois appelé «sensibilité», est la fraction des instances récupérées parmi toutes les instances pertinentes.

Précision (P) : est définie comme le nombre de vrais positifs (T_p) sur le nombre de vrais positifs plus le nombre de faux positifs (F_p).

$$P = \frac{T_p}{T_p + F_p} \quad \text{(III.1)}$$

Rappel (R) : est défini comme le nombre de vrais positifs (T_p) sur le nombre de vrais positifs plus le nombre de faux négatifs (F_n).

$$R = \frac{T_p}{T_p + F_n} \quad \text{(III.2)}$$

III.2.2.1 Le gain cumulé actualisé normalisé NDCG

Pour comprendre NDCG, nous devons comprendre ses prédécesseurs : **Gain Cumulatif (CG)** et **Gain Cumulatif actualisé (DCG) [53].**

a) Le gain cumulé CG :

Chaque recommandation est associée à un score de pertinence. Le gain cumulé est la somme de tous les scores de pertinence d'un ensemble de recommandations.

$$CG = \sum_{i=1}^{n} rel_p \quad \text{(III.3)}$$

rel_p : Représente la liste des items pertinents dans la collection jusqu'à la position p.

b) Le gain cumulé actualisé DCG

Le gain cumulé actualisé (DCG) est souvent utilisé pour mesurer l'efficacité des algorithmes. À l'aide d'une échelle de pertinence graduée des items dans un ensemble de résultats, DCG mesure l'utilité ou le gain d'un item en fonction de sa position dans la liste de résultats. Le gain est accumulé du haut de la liste de résultats vers le bas.

$$\mathbf{DCG} = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i}-1}{\log_2(i+1)} \quad \text{(III.4)}$$

IDCG est le gain cumulé actualisé (DCG) idéal avec ordre.

c) NDCG :

La comparaison des performances d'un algorithme à l'autre ne peut pas être obtenue de manière cohérente en utilisant uniquement DCG, de sorte que le gain cumulé à chaque position pour une valeur choisie de P doit être normalisé. Cela se fait en triant tous les items pertinents du corpus par leur pertinence relative, en produisant le maximum de DCG possible par position P, également appelé Ideal DCG (IDCG) à travers cette position. Pour calculer NDCG, nous devons d'abord calculer :

- DCG de l'ordre recommandé
- DCG de l'ordre idéal (iDCG).

NDCG est alors le rapport entre DCG d'ordre recommandé et DCG d'ordre idéal.

$$\mathbf{NDCG} = \frac{DCG_p}{IDCG_p} \quad \text{(III.5)}$$

Le principal avantage du NDCG est qu'il prend en compte les valeurs de pertinence notées. Lorsqu'elles sont disponibles dans l'ensemble de données, le NDCG est un bon ajustement.

III.3 : Les méthodes comparées

Nous avons comparé une méthode de recommandation basée sur deep learning avec deux méthodes de recommandation basées sur les techniques de machine learning. Puisque la méthode des réseaux de neurones évalue les interactions utilisateur-item, nous avons mis en concurrence la méthode avec des méthodes de filtrage collaboratif utilisateur-item plutôt qu'avec des modèles item-item.

III.3.1 : Filtrage collaboratif utilisant k-voisins les plus proches (KNN)

Il s'agit de l'une des techniques de filtrage collaboratif de voisinage populaires basées sur l'utilisateur [54]. Nous avons adapté cette méthode pour apprendre à partir des données d'interactions utilisateur-item implicites.

III.3.2 Système de recommandation en utilisant le classement personnalisé bayésien (BPR)

Ce modèle effectue l'optimisation en utilisant une technique de perte de classement par paires pour évaluer les interactions utilisateur-item à partir des commentaires implicites en utilisant l'algorithme LearnBPR pour estimer si un utilisateur préfère un item à un autre. [67].

III.3.3 Le filtrage collaboratif neuronal (NCF)

NCF est un nouveau modèle de factorisation de matrice neuronale, qui regroupe la factorisation matricielle généralisée (GMF) et le perceptron multicouche (MLP) pour unifier les forces de linéarité de MF et de non-linéarité de MLP pour modéliser les structures latentes utilisateur-item [56].

III.4 Configuration des systèmes

Dans ce qui suit nous allons présenter l'environnement du développement et les outils que nous avons utilisé pour réaliser notre travail.

Pour l'environnement matériel, nous avons utilisé deux machines qui ont les caractéristiques suivantes :

	Ordinateur portable 1	Ordinateur portable 2
Processeur	Hp500 Intel (R) Core(TM) i3-5005U CPU @ 2.00 GHz 2.00 GHz.	Asus Intel (R) Core(TM) m3-7Y30 CPU @ 1.00 GHz 1.61GHz
Mémoire installée	4,00 Go	4,00 Go
Type de système	système d'exploitation 64 bits, processeur x64	système d'exploitation 64 bits, processeur x64
Systèmes d'exploitation	Windows 8.1 professionnel.	Windows 10 professionnel.

Tableau III.1 Caractéristiques du matériel utilisé.

Pour l'environnement de développement nous avons utilisé :

Anaconda

Anaconda est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions se paquetages sont gérées par le système de gestion de paquets conda. La distribution Anaconda est utilisée par plus de 6 millions d'utilisateur et comprendre plus de 250 paquets populaires en science des données adaptés pour Windows, Linux et MacOs [57].

Jupyter

Jupyter est une application web utilisée pour programmer, initialement développés pour les langages de programmation Julia, Python et R (d'où le nom Jupyter, et supporte près de 40 langages.

Jupyter est une évolution du projet IPython. Jupyter permet de réaliser des calepins ou notebooks qui sont utilisés en science des données pour explorer et analyser des données. La cellule est l'élément de base d'un notebook jupyter. Elle peut contenir du texte formaté au format markdown ou du code informatique qui pourra être exécuté.

Pour les langages de programmation que nous avons utilisés, nous avons :

Python

Python est un langage de programmation assez généraliste, c'est-à-dire qu'il est à peu près possible de tout faire avec : des sites et applications web, des applications mobiles, des scripts personnels, des applications de bureau, de l'analyse de données et même des jeux vidéo. Grâce à ses bibliothèques et packages de data science, il est devenu le langage le plus populaire pour les algorithmes de machine learning, data science et le big data [58]. Nous avons utilisé python 3.7 sur Windows 10.

III.5 Configuration des modèles comparés

Un bon classifieur en machine Learning, est un classifieur qui généralise, autrement dit, il a la capacité de faire des prédictions non seulement sur les données utilisées pour le construire, mais surtout sur de nouvelles données. Pour obtenir ces deux types de données,

la procédure consiste à effectuer un fractionnement en deux sous-ensembles sur le jeu de données :

- Ensemble de données d'apprentissage : communément appelé **train set**, utilisé pour entraîner le classifieur, l'ensemble des observations (enregistrements) est appelé **x_train** et le vecteur de classes (target) correspondantes aux observations (enregistrements) est décrit par **y_train**.
- Ensemble de données test : Communément appelé **test set**, l'ensemble des observations (enregistrements) est appelé **x_test** et le vecteur de classes correspondantes aux observations est appelé **y_test**. Ce sont les données qui n'ont pas servi à l'entraînement, utilisées comme une entrée du modèle pour tester sa performance en comparant les prédictions données par le classifieur **(y_pred)** et les valeurs attendues **(y_test)**.

Le module "division de jeu de données" s'occupe de ce fractionnement en ensemble de test et ensemble d'apprentissage, dans notre implémentation **80%** du jeu de données est consacré pour l'apprentissage quant aux **20%** restantes elles forment l'ensemble de test, pour ce faire nous avons utilisé la fonction **train_test_split de Sklearn**.

Nous avons implémenté trois modèles de l'état de l'art, à savoir KNN, BPR et NCF. Nous spécifions ci-dessous les hyperparamètres de chacun d'eux.

III.5.1 L'algorithme KNN

Nous avons implémenté l'algorithme KNN à l'aide du modèle KNeighborsClassifier présent dans la bibliothèque sklearn, accompagné d'un ensemble de paramètres à spécifier afin d'obtenir le modèle le plus optimal possible. Ces paramètres sont les suivants :

Hyperparamètres	Définition	Plage des valeurs	Valeurs optimales
n_neighbors	Le nombre de voisins pris par le modèle c'est le paramètre (k) de l'algorithme.	[1-300] avec un pas de 50	2
metric	La métrique de similarité à utiliser.	Cosinus	

Tableau III.2 Les hyperparamètres de l'algorithme KNN.

III.5.2 L'algorithme BPR

Cet algorithme a été implémenté à l'aide du Framework Cornac. Le BPR a des paramètres importants que nous avons pris en compte. Ces paramètres sont les suivants :

Hyperparamètres	Définition	Plage des valeurs	Valeurs optimales
k (n_factors)	contrôle la dimension de l'espace latent (c'est-à-dire la taille des vecteurs)	[1-300] avec un pas de 50	100
max_inter (epochs)	nombre d'itérations que l'algorithme fonctionnera sur l'ensemble de données d'apprentissage.	[1-350] avec un pas de 50	200
learning_rate	détermine la taille du pas à chaque itération	0,01	
lambda_rate	contrôle la régularisation dans la fonction objective.	0,001	

Tableau III.3 Les hyperparamètres de l'algorithme BPR.

III.5.3 L'algorithme NCF

Cet algorithme a été implémenté à l'aide du Framework TensorFlow. NCF a beaucoup de paramètres, les plus importants sont les suivants :

Hyperparamètres	Définition	Plage des valeurs	Valeurs optimales
n_factors	Contrôle la dimension de l'espace latent.	[1-350] avec un pas de 50	100
layer_sizes	Taille de la couche d'entrée et des couches cachées de MLP, le type d'entrée est une liste.	[[16, 8,4]-[64, 32,16]]	[16, 8, 4]
n_epochs	Nombre d'entrainement du réseau de neurones.	[1-300] avec un pas de 50	200
model_type	Nous pouvons former un modèle unique « MLP », « GMF » où un modèle combiné « NCF » en changeant le type de modèle.	model_type="NeuMF" Fonctions d'activation : Sigmoïde et ReLu	

Tableau III.4 Les hyperparamètres de l'algorithme NCF.

III.6 Evaluation des algorithmes

Dans ce qui suit nous allons tester les algorithmes avec leurs hyperparamètres à fin de déterminer les valeurs optimales qui obtiennent les meilleurs résultats en terme de précision, de rappel et de NDCG.

III.6.1 : Impact des hyperparamètres

Sur les graphes illustrés dans les figures (III.1), (III.2), (III.3) la courbe orange se réfère à la précision, la grise pour le rappel quant à la courbe bleue pour NDCG.

III.6.1.1 : Impact des hyperparamètres du modèle KNN

En premier lieu, nous examinons les variations des valeurs des trois métriques obtenues par le modèle KNN et cela en fonction de l'hyperparamètre k_neighbors.
La figure (III.1) présente ces variations. Nous notons que nous avons fixé l'autre hyperparamètre du modèle KNN à savoir la mesure de similarité à cosinus.

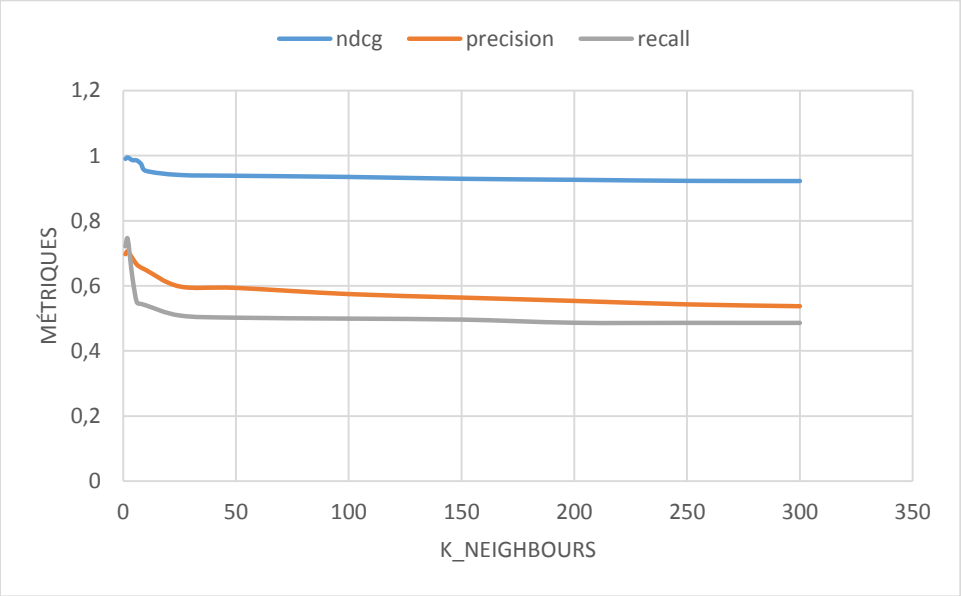


Figure III.1 Impact de l'hyperparamètre K_neighbors (KNN).

A partir de la figure (III.1) nous remarquons que le modèle KNN donne les meilleurs résultats avec les différentes métriques lorsque la valeur de k_neighbors est égale à 2. Les performances du modèle baissent jusqu'à un seuil et se stabilisent à partir du k=25.

III.6.1.2 : Impact des hyperparamètres du modèle BPR

Nous évaluons les performances de l'algorithme BPR en variant les deux paramètres : epochs et n_factors. Nous avons fixé les deux autres hyperparamètres learning_rate à 0,01 et lambda_rate à 0,001.

Le graphe suivant présente les variations des trois métriques d'évaluation en fonction de l'hyperparamètre epochs.

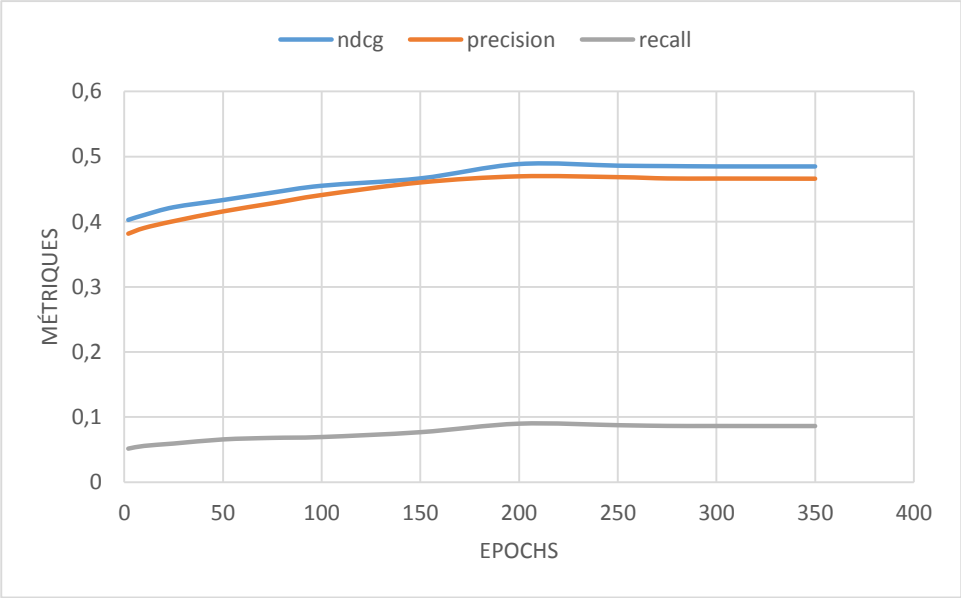


Figure III.2 Impact de l'hyperparamètre epochs (BPR).

Nous observons dans la figure (III.2) que les meilleurs résultats obtenus par l'algorithme BPR avec l'hyperparamètre epochs est lorsqu'elle est égale à 200. Les performances du modèle se stabilisent à partir de cette valeur.

Quant au graphe suivant présente les variations des trois métriques en fonction de l'hyperparamètre n_factors. Nous avons fixé l'hyperparamètre epochs à 200.

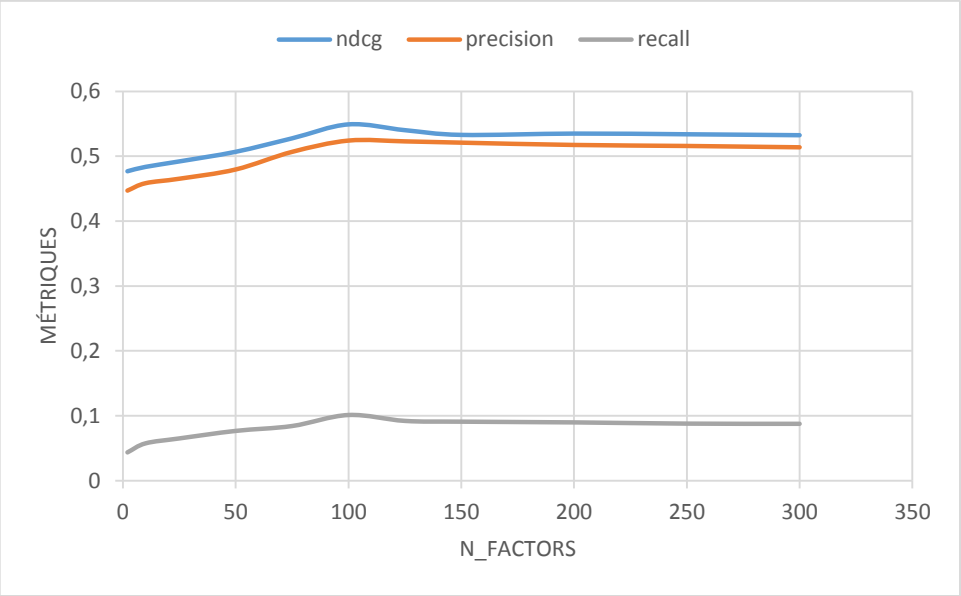


Figure III.3 Impact de l'hyperparamètre n_factors (BPR).

Nous remarquons dans la figure (III.3) que l'algorithme BPR obtient les meilleurs résultats quand l'hyperparamètre n_factors a une valeur égale à 100. Les performances du modèle se stabilisent à partir de cette valeur.

III.6.1.3 Impact des hyperparamètres du modèle NCF

Nous évaluons les performances de l'algorithme NCF en variant les deux hyperparamètres : epochs et n_factors.

Le premier graphe présente la variation des trois métriques d'évaluation en fonction de l'hyperparamètre epochs.

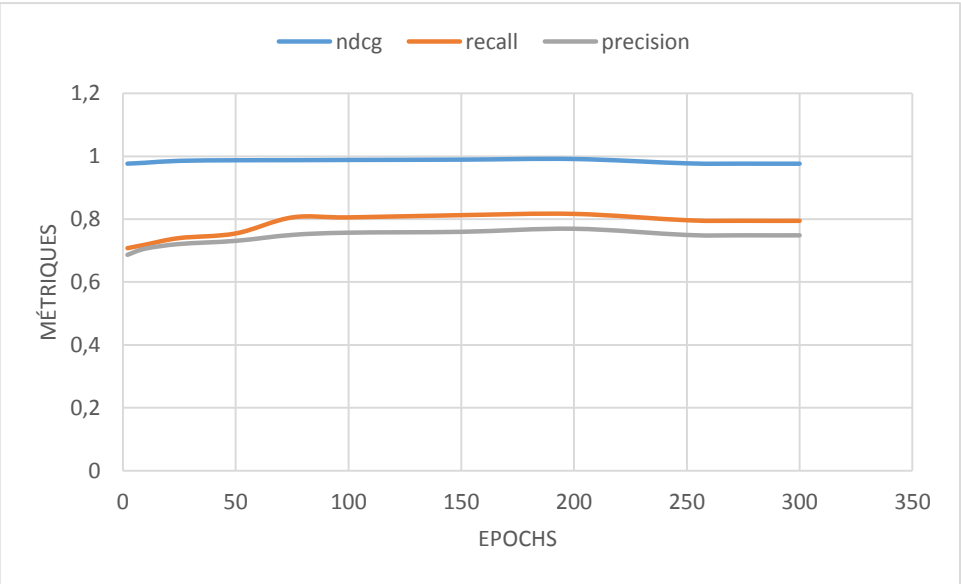


Figure III.4 Impact de l'hyperparamètre epochs (NCF).

Nous constatons à partir de la figure (III.4) que le modèle NCF donne les meilleurs résultats avec les différentes métriques lorsque la valeur epochs est égale à 200. Les performances du modèle se stabilisent à partir de cette valeur.

Le deuxième graphe présente la variation des trois métriques d'évaluation en fonction de l'hyperparamètre n_factors.

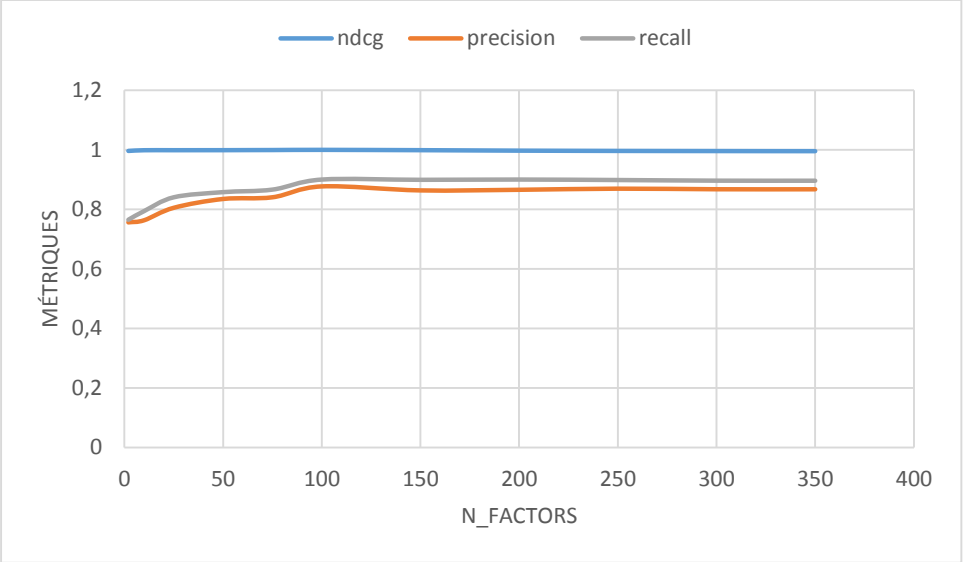


Figure III.5 Impact de l'hyperparamètre n_factors (NCF).

Nous remarquons sur la figure (III.5), que le modèle NCF obtient les meilleurs résultats avec la métrique n_factors avec une valeur égale à 100. Les performances se stabilisent à partir de cette valeur.

III.6.2 : Impact des facteurs

Dans cette section, nous comparons les performances des trois modèles (KNN, BPR et NCF) en utilisant les métriques de précision, de rappel et de NDCG et cela en variant la valeur de l'hyperparamètre k_neighbors pour KNN et n-factors pour BPR et NCF. Nous notons que nous avons fixé l'hyperparamètre epochs à 200 pour BPR et NCF.

Le premier graphe présente la variation de la précision des trois modèles en fonction de factors.

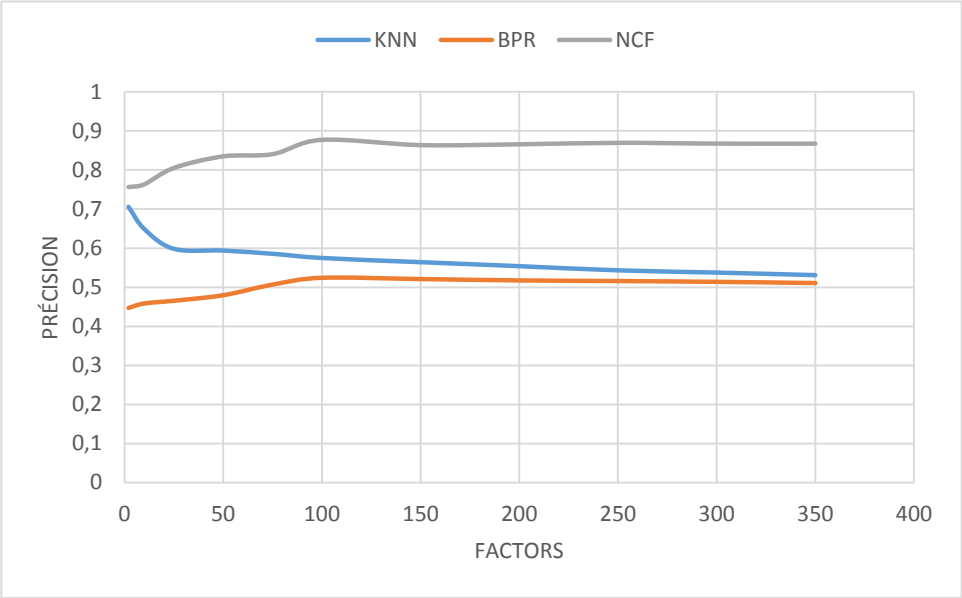


Figure III.6 La variation de la précision des trois modèles en fonctions de factors.

Nous remarquons sur la figure (III.6) que les valeurs de précision obtenues par le modèle BPR varient entre [0.4, 0.5], celles obtenues par KNN varient entre [0.5, 0.7] et les valeurs de NCF varient entre [0.7, 0.9]. Les performances des modèles NCF et BPR se stabilisent à partir de factors=100. Pour KNN les performances se stabilisent à partir du k=25.

Le deuxième graphe présente la variation de rappel des trois modèles en fonction de factors.

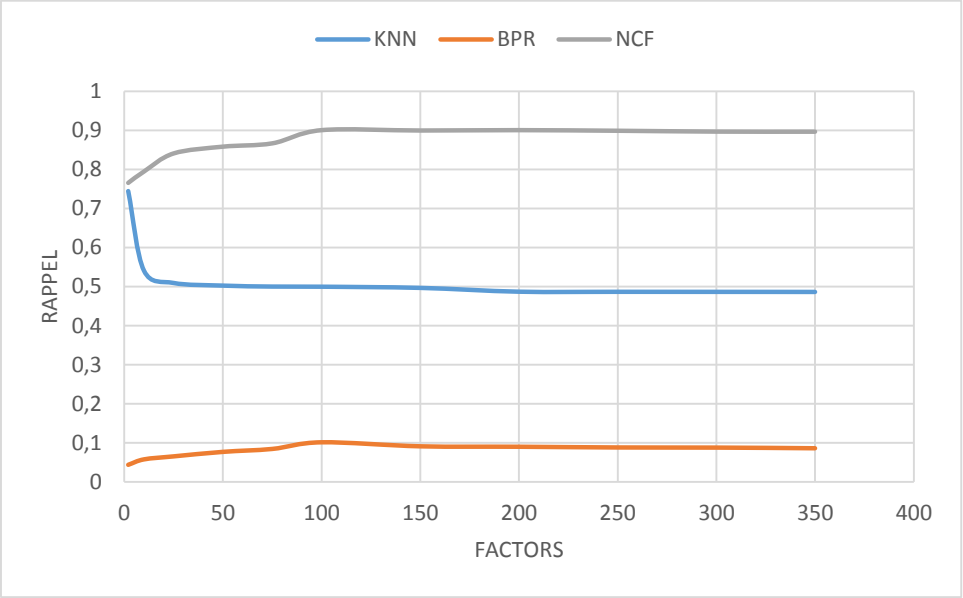


Figure III.7 La variation de rappel des trois modèles en fonctions de factors.

Nous constatons à partir de la figure (III.7) que les valeurs de rappel obtenues par BPR varient aux alentours de 0.1, celles de KNN entre [0.4, 0.7] et celles de NCF entre [0.7, 0.9]. Les performances des modèles NCF et BPR se stabilisent à partir de factors égale à 100, et pour KNN se stabilisent à partir de factors égale à 25.

Le troisième graphe présente la variation de NDCG des trois modèles en fonction de factors.

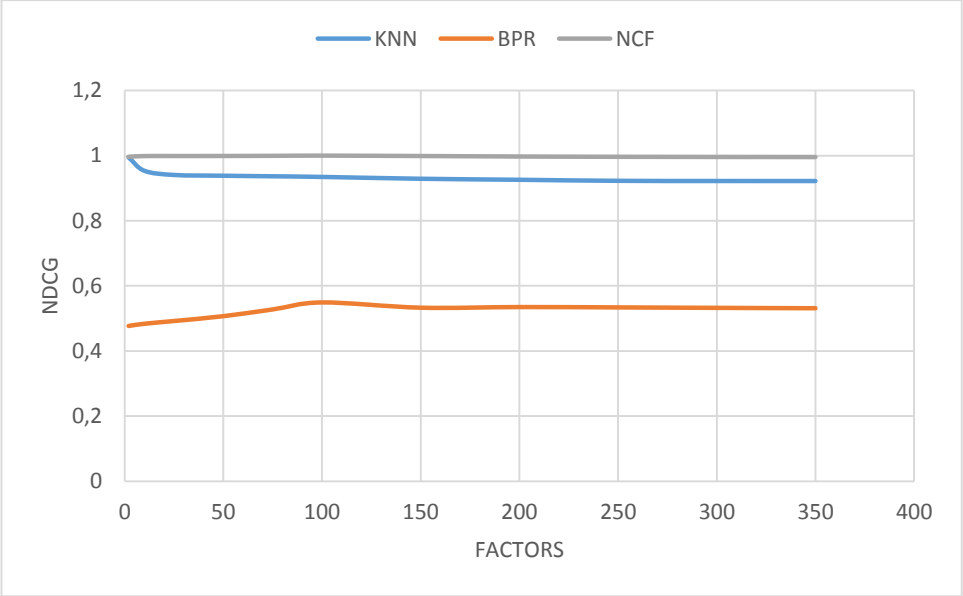


Figure III.8 La variation de NDCG des trois modèles en fonctions de factors.

Nous observons à partir de la figure (III.8) que les valeurs de NDCG obtenues par BPR varient entre [0.4, 0.5], celles de KNN varient aux alentours de 0.9 et celles de NCF entre [0.9, 1]. Les performances de modèle NCF augmentent pour les premières valeurs ensuite se stabilisent. Pour BPR se stabilisent à partir de la valeur 100. Pour KNN se stabilisent à partir de la valeur 25.

Nous constatons la différence entre NFC et les deux autres algorithmes, les valeurs de NCF sont meilleures par rapport à celles de KNN et BPR.

III.7 Comparaison et analyse des algorithmes

Nous présentons dans cette section les résultats des métriques obtenues par les algorithmes implémentés pour notre jeu de données. Nous avons pris comme valeurs des hyperparamètres, celles qui ont donné les meilleurs résultats précisément :

- Pour l'algorithme KNN, k_neighbors=2.
- Pour l'algorithme BPR, n_factors=100 et epochs= 200.
- Pour NCF, n_factors=100 et epochs=200.

Les figures ci-dessous montrent les résultats obtenus :

```
from sklearn.metrics import precision_score, recall_score, ndcg_score
import numpy as np
print("NDCG : ", ndcg_score(np.asarray([y_test]), np.asarray([y_pred]), k=None, sample_weight=None, ignore_ties=False ))
print("Precision :", precision_score(y_test, y_pred, labels=None, pos_label=1,
                    average='macro', sample_weight=None, zero_division='warn'))
print("Recall : ", recall_score(y_test, y_pred, labels = None , pos_label = 1 ,
                  average='macro' , sample_weight = None , zero_division = 'warn' ))
```

```
NDCG :  0.9945533660430288
Precision : 0.7058352212378244
Recall :  0.7442653452832564
```

Figure III.9 résultat de l'évaluation avec KNN.

```
k=10
eval_ndcg = ndcg_at_k(test, all_predictions, col_prediction='prediction', k=k)
eval_precision = precision_at_k(test, all_predictions, col_prediction='prediction', k=k)
eval_recall = recall_at_k(test, all_predictions, col_prediction='prediction', k=k)


print("NDCG:\t%f" % eval_ndcg,
      "Precision:\t%f" % eval_precision,
      "Recall:\t%f" % eval_recall, sep='\n')
```

```
NDCG:   0.549117
Precision:      0.524149
Recall: 0.101258
```

Figure III.10 résultat de l'évaluation avec BPR.

```
eval_ndcg = ndcg_at_k(test, all_predictions, col_prediction='prediction', k=TOP_K)
eval_precision = precision_at_k(test, all_predictions, col_prediction='prediction', k=TOP_K)
eval_recall = recall_at_k(test, all_predictions, col_prediction='prediction', k=TOP_K)

print("NDCG:\t%f" % eval_ndcg,
      "Precision:\t%f" % eval_precision,
      "Recall:\t%f" % eval_recall, sep='\n')
```

```
NDCG:   1.000000
Precision:      0.877263
Recall: 0.900243
```

Figure III.11 résultat de l'évaluation avec NCF.

Nous présentons dans le tableau ci-dessous les résultats obtenus avec les différentes métriques.

Métrique / Algorithme	NDCG	Précision	rappel
KNN	0.994553	0.705835	0.744265
BPR	0.549117	0.524149	0.101258
NCF	1.00	0.877263	0.900243

Tableau III.5 Comparaison des performances des trois modèles.

Le tableau ci-dessus montre que les algorithmes ont donné des résultats différents pour le jeu de données MovieLens, nous remarquons que les valeurs des métriques varient : entre [0.7, 0.9] pour KNN. Entre [0.1 ,0.5] pour BPR et pour NCF elles varient dans la plage de [0.8, 1.00], ce dernier modèle est celui qui a affiché les meilleurs résultats.

D'après les résultats obtenus, nous remarquons que l'algorithme basé sur les réseaux de neurones (NCF) est plus performant pour cet ensemble de données que les deux autres algorithmes (BPR et KNN). Nous pouvons conclure que les réseaux de neurones peuvent être bénéfiques pour les systèmes de recommandation.

III.8 Conclusion

Dans ce chapitre, nous avons présenté les résultats de l'implémentation des trois différents modèles de recommandation basés sur les algorithmes de machine et deep learning.

Précisément, l'objectif de notre travail est dans un premier temps d'implémenter et d'évaluer les trois modèles : le filtrage collaboratif neuronal (NCF), le filtrage collaboratif en utilisant k-voisins les plus proches (KNN) et le modèle basé sur le classement personnalisé bayésien (BPR), en utilisant un ensemble de métriques, et ensuite de comparer les résultats d'évaluation obtenus par les trois modèles avec les valeurs optimales de leurs hyperparamètres.

Nous avons constaté que le filtrage collaboratif basé sur les réseaux de neurones est celui qui a affiché les meilleurs résultats avec toutes les métriques utilisées.

Conclusion générale

Le travail présenté dans ce mémoire rentre dans le cadre de l'exploitation des techniques de machine et deep learning dans les systèmes de recommandation. Précisément, nous avons implémenté, évalué et comparé trois algorithmes de filtrage collaboratif, l'un est basé sur les réseaux de neurones (NCF) les deux autres sont basés sur les techniques de machine learning (KNN et BPR). Dans notre cas, on a pu constater que l'algorithme NCF a donné de meilleurs résultats comparé aux autres algorithmes testés.

Nous avons réussi, à travers la série des tests effectués sur les algorithmes, à constater l'impact des hyperparamètres sur les performances des modèles. L'évaluation faite à l'aide des métriques nous a permis de déduire que la technique des réseaux de neurones présente les meilleurs résultats pour notre ensemble de données.

Pour ce faire, nous avons dans un premier temps**,** donné une vision générale sur le machine et deep Learning et introduit les types d'apprentissage, ensuite présenter les réseaux de neurones avec ses principales architectures. Dans un second temps, nous avons défini les systèmes de recommandation et les approches de filtrage qui permettent la recommandation ainsi que les différents types et quelques travaux de l'état de l'art basés sur le machine et deep learning. Et pour finir, nous avons présenté notre travail en introduisant l'implémentation des algorithmes basés sur le machine et deep learning, discuter les résultats d'évaluation et de comparaison, nous avons aussi spécifié le jeu de données, les métriques d'évaluation, l'environnement du développement et les outils utilisés pour la réalisation.

Ce travail n'a pas été exempt d'obstacles. En effet, nous avons été confrontés à différentes difficultés, par exemple au niveau de l'implémentation des algorithmes, cependant c'était une véritable expérience et une chance pour la découverte du domaine du deep learning.

Comme perspectives par rapport à ce travail, il serait intéressant de faire cette étude sur différents jeux de données pour montrer si le choix de l'algorithme dépend du jeu de données, de sa taille et aussi des hyperparamètres du l'algorithme.

Bibliographie

[1] Thierry Denœux Université de Technologie de Compiègne Département Génie Informatique Heudiasyc ,14 septembre 2018

[2] https://blog.varonis.fr/classification-des-donnees-kesako/

[3] Classification avec les réseaux de neurones. Réalisé par : HADJ SAID Nadine, REKOUANE Hichem 2019-2020

[4] :https ://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Allez_plus_loin_ :_le_deep_Learning

[5] : La reconnaissance des expressions faciales Présentée par : DIALLO Nene Adama Dian Juillet 2019

[6]:https://medium.com/diaryofawannapreneur/deep-learning-for-computer-vision-for-the-average-person-861661d8aa61

[7] : Développement d'une Architecture Basée sur l'Apprentissage Profond (Deep Learning) pour la Détection d'Intrusion dans les Réseaux. Mr. MIMOUNE Zakarya.

[8] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics 5: 115-133, 1943.

[9] D. Hebb, The Organization of Behavior, New York: Wiley, 1949.

[10] F. Rosenblatt, "The Perceptron: a probabilistic model for information storage and organization in the brain," Psychological Review 65: 386-408, 1958.

[11] B. Widrow and M. Hoff, "Adaptive switching circuits," 1960 IRE WESCON Convention Record, New York: IRE, pp. 96-104, 1960.

[12] T. Kohonen, cf. chp. 5

[13]:https://www.futura-sciences.com/tech/dossiers/robotique-presentation-historique-reseaux-neuronaux-31/

[14]: https ://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-rn.pdf

[15] https ://hal-amu.archives-ouvertes.fr/hal 01338010/file/Les_reseaux_de_neurones_artificiels.pdf

[16] https://www.quora.com/What-is-the-difference-between-deep-and-shallow-neural-networks

[17] https ://fr.wikipedia.org/wiki/Fonction_d%27activation

[18] : Classification avec les réseaux de neurones. Réalisé par : HADJ SAID Nadine, REKOUANE Hichem 2019-2020

[19] https ://www.racinely.com/post/les-fonctions-d-activation-part
[20] Bergstra, J. et Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research, 13(Feb), 281-305.
[21] https://datasciencetoday.net/index.php/en-us/deep-learning/173-les-reseaux-de-neurones-convolutifs
[22] : Classification avec les réseaux de neurones. Réalisé par : HADJ SAID Nadine, REKOUANE Hichem 2019-2020
[23] Classification avec les réseaux de neurones. Réalisé par : HADJ SAID Nadine, REKOUANE Hichem 2019-2020
[24]: https://missinglink.ai/guides/neural-network-concepts/recurrent-neural-network-glossary-uses-types-basic-structure/
[25] https ://dataanalyticspost.com/Lexique/auto-encodeur/
[26] : https://dataanalyticspost.com/Lexique/auto-encodeur/
[27] https ://tel.archives-ouvertes.fr/tel-00719609/document
[28] https ://www.lebigdata.fr/reinforcement-learning-definition
[29] https ://www.lebigdata.fr/tensorflow-definition-tout-savoir
[30] https ://fr.linkfang.org/wiki/Keras
[31] https://datafranca.org/wiki/PyTorch
[32] Malone, T., Brobst, S., Cohen, S., Grant, K., and Turbak, F. (1987). Intelligent information des systemes de partage. In Communications of the ACM, volume 30, pages 390–402.
[33] Resnick, P. and Varian, H. (1997). Recommender systems. In Communications of the ACM, volume 40, pages 56–58.
[34] Maes, P. and Shardanand, U. (1995). Social information filtering: algorithms for automating “word of mouth”. In the SIGCHI conference on Human factors in computing systems, Denver, Colorado, United States. ACM Press/Addison-Wesley Publishing Co.
[35] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4) :331–370.
[36] A. T. NGUYEN. COCoFil2 : Un nouveau système de filtrage collaboratif basé sur le modèle des espaces de communautés. Thèse. Université joseph Fourier Grenoble I. Novembre 2006.
[37] : https://www.semanticscholar.org/paper/Cartes-de-communaut%C3%A9s-pour-l'adaptation-interactive-Nguyen-Denos/92979cdc814c089a7b13a57a0eeb632801f43a23

[38]: Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6):734–749.
[39]: Su, X., and Khoshgoftaar T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence. Hindawi Publishing Corporation. 1-19.
[40]: Rao, N. and Talwar, V. (2008). Application domain and functional classification of recommender systems a survey. Desidoc journal of library and information technology, 28(3)
[41]: Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4):331–370.
[42] : Système de recommandation des services web sémantiques Réalisé par : Mme KARAOUZENE Meryem Présenté le 14 Juin 2015
[42]: Système de recommandation des services web sémantiques Réalisé par : Mme KARAOUZENE Meryem Présenté le 14 Juin 2015
[43] : Système de Recommandation de Cours à Base d'Ontologie Présentée par Ait Ahmed Nora Idris Khodja Asma 2018
[44] : Utilisation de l'approche pré-filtrage contextuel des systèmes de recommandation sensible au contexte Réalisé par : - BENOSMAN Imene. - CHARIF Nesrine 2015-2016
[45]: Utilisation de l'approche pré-filtrage contextuel des systèmes de recommandation sensible au contexte Réalisé par : - BENOSMAN Imene. - CHARIF Nesrine 2015-2016
[46]: Système de recommandation des services web sémantiques Réalisé par : Mme KARAOUZENE Meryem Présenté le 14 Juin 2015
[47] : Utilisation de l'approche pré-filtrage contextuel des systèmes de recommandation sensible au contexte Réalisé par : - BENOSMAN Imene. - CHARIF Nesrine 2015-2016
[48] Deep Learning based Recommendation Systems, Nishanth Reddy Pinnapareddy, Mai 2018
[49] https://towardsdatascience.com/recommender-system-using-bayesian-personalized-ranking-d30e98bba0b9
[50]: https://www.jeremyjordan.me/autoencoders/
[51] Appariements collaboratifs des offres et demandes d'emploi à Orsay, le 29/06/2018, par Thomas Schmitt
[52] https://d2l.ai/chapter_recommender-systems/movielens.html
[53] https://grouplens.org/datasets/movielens/
[54] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets." in ICDM, 2008, pp. 263–272.

[55] https://towardsdatascience.com/recommender-system-bayesian-personalized-ranking-from-implicit-feedback-78684bfcddf6
[56] https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401
[57] https://www.anaconda.fr/
[58] https://openclassrooms.com/