République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université Yahia Farés de MEDEA

Faculté des Sciences

Département Informatique et Mathématique

Mémoire de fin de cycle

En Vue de L'obtention du Diplôme de Master en Informatique

Option : Ingénierie des Systèmes et Technologies du Web

Thème

Système de recommandation basé sur les relations sémantiques

Présenté par :

- **M^elle^ Zaoui Fadila**
- **M Khomri Abdelkader**

Dirigé par :

Promotrice : M^me^ Sadouki Fatiha

Année universitaire : 2021/2022

Dédicaces

Je dédie ce mémoire à mes très chers parents.

Ce travail est le fruit de la rigueur de votre éducation que dieu vous bénisse et vous garde.

A mes frères et sœurs

A qui m'admirez tant, soyez sûrs que ce travail est le résultat de votre confiance en moi soyez en remercies.

Je remercie et je salue mes professeurs de l'école primaire à l'université ainsi tous mes amis et mes voisins

Zaoui Fadila

Khomri Abdelkader

Remerciement

Nous tenons tout d'abord à remercier Dieu tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.

En second lieu, nous tenons à remercier particulièrement notre encadreur madame Sadouki Fatiha pour son encadrement et pour l'intérêt qu'il a manifesté à notre travail.

A M Hantabli Hamza pour son aide qui nous a apporté durant l'élaboration de notre projet.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.

Zaoui Fadila

Khomri Abdelkader

Résumé

Les systèmes de recommandation (RS) sont utilisés pour de nombreux produits et services fournis par de grandes entreprises telles qu'Amazon, Google, YouTube, Facebook, Netflix……, et chaque service à ses mécanismes de travail, Dans le cadre de ce mémoire de master, nous voulons réaliser un système de recommandation basé sur les relations sémantiques à partir des descriptions des films en appliquant les algorithmes d'apprentissage en profondeur dans la perspective est de satisfaire l'utilisateur par rapport à ses préférences.

Notre travail est subdivisé en deux parties, la première partie présente l'état de l'art de projet qui explique les systèmes de recommandation en détail et l'apprentissage automatique. La deuxième partie présente l'approche proposée y compris les étapes de création du modèle ainsi que l'évaluation des résultats en utilisant deux métriques différentes et à la fin, la comparaison de nos résultats avec ceux d'autres modèles.

Mots-clés : Système de recommandation, Apprentissage en profondeur, Auto-encodeur, MLP, Sémantique.

Abstract

Recommender systems (RS) are used for many products and services provided by big companies such as Amazon, Google, YouTube, Facebook, Netflix..., and each service has its working mechanisms, In this master thesis, we want to realize a recommender system based on semantic relations from movie descriptions by applying deep learning algorithms with the perspective is to satisfy the user with respect to his preferences.

Our work is subdivided into two parts, the first part presents the state of the art of the project which explains the recommendation systems in detail and the machine learning. The second part presents the proposed approach including the steps of creating the model as well as the evaluation of the results using two different metrics and at the end, the comparison of our results with other models.

Keywords: Recommender system, Deep learning, Auto-encoder, MLP, Semantics.

ملخـــص

تُستخدم أنظمة التوصية (RS) للعديد من المنتجات والخدمات التي تقدمها الشركات الكبيرة مثل Amazon و Google و YouTube و Facebook وNetflix ،.....الخ ، وكل خدمة لها آليات عملها ، في إطار هذه الاطروحة ، نريد تحقيق نظام توصية قائم على العلاقات الدلالية في أوصاف الأفلام من خلال تطبيق خوارزميات التعلم العميق ، بهدف إرضاء المستخدم فيما يتعلق بتفضيلاته.

ينقسم عملنا إلى قسمين ، الجزء الأول يعرض أحدث ما توصل إليه المشروع والذي يشرح بالتفصيل أنظمة التوصية والتعلم الآلي. . يقدم الجزء الثاني النهج المقترح بما في ذلك خطوات إنشاء النموذج وكذلك تقييم النتائج باستخدام مقياسين مختلفين وفي النهاية ، مقارنة نتائجنا بنتائج النماذج الأخرى.

Table des matières

Remerciement

Résumé

Abstract

ملخص

Introduction générale 1

Chapitre 1 : **Systèmes de recommandation**

1.1. Introduction 4

1.2. Historique 4

1.3. Définition 5

1.4. Objectifs des systèmes de recommandation 6

1.5. Forme de collecte des données 7

1.5.1. Feedback explicite - Filtrage dit actif ou réactif 7

1.5.2. Feedback implicite – Filtrage dit passif ou proactif 7

1.5.3. Différences d'interprétation entre les données explicites et implicites 8

1.6. Matrice d'évaluation Utilisateur-Item 8

1.7. Problématique de la recommandation 9

1.8. Approches des systèmes de recommandation 9

1.8.1. Approche basée contenu 9

1.8.2. Approche collaborative 10

I.9. Classifications des systèmes de recommandation 15

1.10. Types des recommandations 16

1.10.1. Recommandation basée sur le filtrage collaboratif 16

1.10.2. Recommandation basée sur le contenu 20

1.10.3. Le Filtrage hybride 22

1.10.4. Autres types 23

1.11. Avantages et inconvénients des méthodes des systèmes de recommandation 25

1.12. Métrique d'évaluation 27

1.12.1. Mesures statistiques de précision 27

1.12.2. Mesures permettant l'aide à la décision 28

1.12.3. Couverture 29

1.12.4. Temps de calcul 30

1.13. Conclusion 30

Chapitre 2 : Apprentissage automatique

2.1. Introduction 31
2.2. Notions de base 31
2.2.1. Apprentissage automatique 31
2.2.2. Apprentissage profond 33
2.3. Réseaux de neurones 36
2.3.1. Historique 36
2.3.2. Définition 36
2.3.3. Les réseaux de neurones artificiels et profonds 37
2.3.4. Fonctions d'activation 38
2.3.5. Hyperparamètres de réseau neuronal 40
2.3.6. Les principales architectures des réseaux de neurones 42
2.3.7. Avantages et inconvénients 47
2.4. Apprentissage automatique et NLP 47
2.4.1. NLP - Natural Language Processing 48
2.4.2. Word2Vec 49
2.5. Présentation de quelques travaux basés sur les systèmes de recommandation et machine learning 50
2.5.1. Filtrage collaboratif utilisant k-voisins les plus proches (KNN) 50
2.5.2. Perceptron multicouches (MLP) 51
2.5.3. Système de recommandation utilisant le classement personnalisé bayésien (BPR) 51
2.5.4 Le filtrage collaboratif neuronal (NCF) 54
2.5.5 Le filtrage collaboratif basé sur un auto-encodeur (ACF) 55
2.7. Conclusion 56

Chapitre 3 : Approche proposé

3.1. Introduction 60
3.2. Architecture de l'approche proposée 60
3.3. Description de l'approche 62
3.4.1. Etape 1 : Prétraitement 63
3.4.2. Etape 2 : Traitement 64
3.4.4. Etape 3 : Modèle 65
3.4.4. Etape 4 : Recommandation 65
3.5. Conclusion 66

Chapitre 4 : Expérimentations

4.1. Introduction 67

4.2. Collection de données 67
4.2.1. Jeu de données 67
4.3. Outils et Langages 69
4.3.1 Python 69
4.3.2 Plateformes de développement 70
4.3.3 Kaggle kernels 72
4.4. Mesures d'évaluation utilisées 72
4.4.1 Erreur quadratique moyenne (RMSE) 73
4.4.2 Erreur absolue moyenne (MAE) 73
4.4.3 Précision TOP-N 73
4.5. Processus d'évaluation 74
4.5.1 Matrice de factorisation 74
4.5.2 Auto-encodeurs 74
4.5.3. MLP 76
4.6. Comparaison et analyse 78
4.6.1. Algorithmes choisis pour la comparaison 79
4.6.2. Analyse des résultats 80
4.7. Conclusion 80
Conclusion générale 78
Références bibliographiques
Webographie

Liste des figures

Figure 1.1 Exemple d'une matrice d'évaluation Utilisateur-Item avec feedback explicite.... 09

Figure 1.2 Architecture du perceptron multicouche du Neural Collaborative Filtering …..... 13

Figure 1.3 Combinaison du MLP non-linéaire et GMF ……..……... 14

Figure 1.4 Principales classifications des systèmes de recommandations ……. 15

Figure 1.5 Recommandation basée sur le filtrage collaboratif …….. .16

Figure 1.6 Exemple de recommandation basé sur le filtrage collaboratif …...…… 17

Figure 1.7 Recommandation basée sur le contenu ……..…….……..…… 21

Figure 1.8 Filtrage hybride …….…...……. 22

Figure 1.9 Recommandation démographique …….…….....….....…23

Figure 1.10 Recommandation à base de connaissance ……....…… 24

Figure 1.11 Recommandation communautaire …….....…..……..25

Figure 2.1 Illustratif de deep learning avec plusieurs couches …….…… 36

Figure 2.2 La différence entre l'apprentissage profond et l'apprentissage automatique 37

Figure 2.3 Illustration d'un Réseau de Neurones …….…… 39

Figure 2.4 Réseau de neurone artificiel (a) et profond(b) ……..……..….. 39

Figure 2.5 Structure simplifié d'un neurone artificiel …….…….…….… 40

Figure 2.6 Architecture standard d'un réseau à convolutions ……..…….……... 45

Figure 2.7 Réseau de neurone déroulé ……...…….…….……. 46

Figure 2.8 Auto-encodeur simple …….…… ..48

Figure 2.9 Illustration une processus classique du NLP …….……..…….... 50

Figure 2.10 Illustration de l'Architecture de modèle : CBOW et Skip-gram ……..…..... 52

Figure 2.11 les interactions existantes entre l'utilisateur et l'item ……..…..…….…... 54

Figure 2.12 les préférences par paire spécifiques à l'utilisateur entre une paire d'items…..... 54

Figure 2.13 la courbe ROC ……....…….. 55

Figure 2.14 : Le schéma de base d'un auto-encodeur variationnel ……..……. 56

Figure 3.1 Architecture du système de recommandation proposé ……..…….. .59

Figure 3.2 Architecture de l'étape 1 : Prétraitement …….. 61

Figure 3.3 Architecture de l'étape 2 : Traitement …….…… .62

Figure 3.4 Architecture de l'étape 3 : Modèle ……..…….…….....……. 69

Figure 4.1 L'environnement Kaggle …….…….. 73

Figure 4.2 Un extrait de l'ensemble de données de films 74

Figure 4.3 Un extrait du fichier texte des notes ……..…..…….. 74

Figure 4.4 Représentation modèle W2V du mot "fun" …….……..…….. 75

Figure 4.5 Exemple de deux films transformés en vecteurs de taille fixe ……..... 75

Figure 4.6 RMSE et MAE changent d'époque pendant la formation MLP……..…..……81

Figure 4.7 Comparaison entre nos deux modèles et différents autres algorithmes ……83

Liste des Tableaux

Tableau 1.1 Avantages et inconvénients des méthodes de recommandation 26

Tableau 2.1 les fonctions d'activation 41

Tableau 2.2 Liste des hyperparamètres..... 43

Tableau 2.3 Avantages et inconvénients ……..……..... 41

Tableau 2.4 Les fonctionnalités des plateformes ……..…...…….…… 58

Tableau 3.1 Une description des formes utilisées dans l'architecture ……..…….. 65

Tableau 4.1 différentes données relatives à l'apprentissage des auto-encodeurs ……..…..79

Tableau 4.2 Métriques d'erreur de MLP pour les deux modèles……..…….. 81

Tableau 4.3 Comparaison entre nos deux modèles et différents autres algorithme ….....….82

Liste des acronymes

AE	Auto-encoder.
ANN	Artificial Neural Networks.
BP	Back-Propagation.
CNN	Convolutional Neural Network.
DNN	Deep Neural Network.
DL	Deep Learning.
FM	Factorisation Matricielle.
HMAE	High Mean Absolute Error.
IMDb	Internet Movie Data base.
IB	Items-Based.
KNN	K-Nearest Neighbors.
MAE	Mean Absolute Error.
ML	Machine Learning.
MSE	Mean Squared Error.
ReLu	Rectified Linear Unit.
RN	Reseau de neurone.
RNN	Récurrent Neural Network.
IA	Intelligence Artificielle
RMSE	Root Mean Squared Error.
SVD	Singular Value Decomposition.
SVM	Support Vector Machine.
MLP	Perceptron Multicouche
GMF	Generalized Matrix Factorization
SR	System Recommendation
NLP	Natural Language Processing
CBOW	Continuous Bag of Words

Introduction Générale

Introduction générale

Sur Internet, là où le nombre de choix est écrasant, il est nécessaire de filtrer, hiérarchiser et fournir efficacement les informations pertinentes afin d'atténuer le problème de la surcharge d'informations, qui a créé un problème potentiel pour de nombreux utilisateurs d'Internet. Les systèmes de recommandation résolvent ce problème en recherchant dans un grand volume d'informations générées dynamiquement pour fournir aux utilisateurs un contenu et des services personnalisés, ils ont la capacité de prédire si un utilisateur particulier préfère un item ou non en fonction de son profil. Ces systèmes font désormais partie de notre vie de tous les jours, de Facebook à Amazon en passant par Google, la quasi-totalité de nos interactions avec le monde d'Internet, à un moment ou à un autre, fait appel à un système de recommandation [1].

Les Systèmes de Recommandation (SR) sont une plateforme pour interaction sociale, c'est une manière de proposer à l'utilisateur des produits qui sont susceptibles de l'intéresser. Ce dernier voit ainsi réduit son temps de recherche mais reçoit également des suggestions de la part du système auxquelles il n'aurait pas prêté attention [2].

En effet, les utilisateurs reçoivent des recommandations d'items (ressources) sur la base de leurs profils. Le profil d'un utilisateur évolue au cours du temps grâce aux interactions entre l'utilisateur et le système, notamment grâce aux évaluations produites par cet utilisateur. Ces évaluations sont sous la forme de notes explicites ou implicites.

Plusieurs approches ont été proposées pour résoudre cette problématique, parmi les plus populaires se trouve l'approche basée modèle, qui se sert de modèles pour représenter les interactions Utilisateur-Item. Parmi ces derniers, se trouve les méthodes classiques basées sur la factorisation matricielle et, plus récemment, des méthodes basées sur l'apprentissage profond qui montrent un potentiel pour résoudre ce problème

Une des techniques les plus répandues des SR et qui reposent principalement sur les notes explicites données par les utilisateurs aux différentes ressources est le Filtrage Collaboratif (FC). Cette méthode base ses calculs sur l'aspect de collaboration et démarre avec ce qu'on appelle une matrice de note $Utilisateur \times Item$, calcule une similarité entre les utilisateurs/items pour désigner un ensemble de voisins et calculer, par la suite, une prédiction pour les notes manquantes.

Nous nous intéressons, dans notre travail, aux relations sémantiques entre les items (ou utilisateurs). En effet, la détection de communautés peut être vue comme une généralisation du principe du filtrage collaboratif où le principe est de recommander à une personne les items le bien évalué par les membres de sa communauté.

La matrice utilisateur-item R peut être vue comme une matrice d'adjacence d'un graphe biparti. Ce graphe biparti comprend l'ensemble des utilisateurs et l'ensemble des

items. Ils sont liés par des notes qui représentent le niveau d'appréciation d'un item par un utilisateur.

L'objectif de ce mémoire est de présenter, dans une première étape, un modèle pouvant encoder des items dans un espace vectoriel grâce à des réseaux de neurones utilisant des données d'entrées textuelles. Cela permet de trouver les relations sémantiques entre les items, c.-à-d. sur la description des films en se basant sur les auto-encodeurs. La seconde étape consiste à prédire des des scores implicites à partir des données explicites. L'algorithme créé pour faire les recommandations avec ces données est un réseau de neurones MLP (Perceptron multicouche).

Pour bien structurer notre travail, nous l'avons élaboré en quatre chapitres, comme suit :

- Le premier chapitre présente un état de l'art des différents aspects abordés tout au long de notre travail, à savoir les systèmes de recommandation.
- Le second chapitre, est consacré à l'analyse de différentes méthodes existantes utilisées dans l'apprentissage automatique pour réaliser un système de recommandation.
- Le troisième chapitre concerne la modélisation. Nous présentons l'approche proposée et choisie pour la réalisation de notre modèle, accompagnée de sa conception.
- L'objectif du quatrième chapitre est de présenter l'expérimentation et l'évaluation.

Enfin, nous concluons ce travail en résumant nos principales contributions et les perspectives de cette étude.

Chapitre 1

Systèmes de Recommandation

1.1. 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, ...

Dans ce chapitre, nous commençons par définir ce qu'un système de recommandation et ses objectifs. Ensuite, nous présentons les différentes approches de recommandations.

Enfin, nous présentons quelques travaux qui sont basés sur les techniques de l'IA, Apprentissage Automatique [5].

1.2. Historique

La capacité des ordinateurs pour faire des recommandations à des utilisateurs a été reconnue assez tôt dans l'histoire de l'informatique. Grundy [78], un système bibliothécaire, était une première étape vers des systèmes de recommandation automatiques. Ce système était assez primitif. Il classait les utilisateurs en "stéréotypes" en se basant sur une courte interview, et utilisait ces stéréotypes pour produire des recommandations de livres. Ce travail constituait une première tentative intéressante dans le domaine des systèmes de recommandation. Cependant, son utilisation est restée très limitée.

Au début des années 1990, le filtrage collaboratif apparait comme une solution pour faire face à la surcharge d'information. L'année 1992 voit l'apparition du système de recommandation de documents Tapestry [74], ainsi que la création du laboratoire de recherche GroupLens, qui travaille explicitement sur le problème de la recommandation automatique dans le cadre des forums de news de Usenet. Tapestry avait pour but de recommander à des groupes d'utilisateurs des documents issus des newsgroups susceptibles de les intéresser. L'approche utilisée était de type "plus proches voisins" à partir de l'historique de l'utilisateur. On parle alors de filtrage collaboratif manuel, comme une réponse au besoin d'outils pour le filtrage de l'information énoncé à la même époque. La recommandation résulte d'une action collaborative des utilisateurs qui recommandent à d'autres utilisateurs des documents en leur attribuant des notes d'intérêt selon certains

critères. Les systèmes de filtrage collaboratif automatiques apparaissent ensuite. GroupLens [75] utilise cette technique pour identifier les articles de Usenet susceptibles d'être intéressants pour un utilisateur donné. Les utilisateurs doivent seulement attribuer des notes ou effectuer d'autres opérations observables (par exemple, lire un article) ; le système résultats personnalisés. Avec ces systèmes, les utilisateurs n'ont aucune connaissance directe des opinions des autres utilisateurs, ni des articles présents dans le système.

Au cours de ces dernières années, les systèmes de recommandation deviennent un sujet d'un intérêt croissant dans les domaines de l'interaction homme-machine, de l'apprentissage automatique ainsi que la recherche d'information. En 1995 apparaissent successivement Ringo [76], un système de recommandation de musique, basé sur les appréciations des utilisateurs et Bellcore [77], un système de recommandation de vidéos. La même année GroupLens crée la société Net Perceptions dont le premier client a été Amazon. De nos jours, les systèmes de recommandation sont devenus des composantes incontournables pour la plupart des sites du e-commerce.

1.3. Définition

A quel moment peut-on vraiment parler de système de recommandations ? Google est-il un système de recommandations ou un moteur de recherche ? La définition de Wikipédia nous livre une première interprétation : "**Les systèmes de recommandation sont une forme spécifique de filtrage de l'information visant à présenter les éléments d'information qui sont susceptibles d'intéresser l'utilisateur**". Cette définition nous donne les informations essentielles d'un système de recommandations, à savoir le filtrage de l'information combiné à la recherche d'intérêt de l'utilisateur.

Un pur moteur de recherche documentaire donnerait les résultats les plus pertinents en se basant uniquement sur la pertinence des documents, mais sans rechercher l'intérêt de l'utilisateur, celui-ci ne reste qu'un moteur de recherche. Google n'a jamais été un pur moteur de recherche puisque la première version utilisait déjà le célèbre algorithme du PageRank qui permet de classer les résultats en fonction de leur popularité. Cette recherche de popularité a grandement suscité l'intérêt des utilisateurs et a permis de propulser le site en tant que leader mondial de la recherche. Sur base de ces observations, on peut considérer Google en tant que système de recommandations axé sur la pertinence.

Les systèmes de recommandation (SR) sont définis comme étant des outils et techniques logiciels fournissant à un utilisateur des suggestions d'items susceptibles de l'intéresser. Ces suggestions se rapportent à des processus décisionnels variés, elles peuvent être des propositions de livre à acheter, de la musique à écouter, des films à regarder, des articles à lire, un restaurant à choisir, ...

Autrement dit, un système de recommandation est une forme spécifique de filtrage de l'information qui cherche à prédire la valorisation ou la préférence qu'un utilisateur attribuerait à un item [60].

Un système de recommandation requiert généralement 3 étapes :

1. La première consiste à recueillir de l'information sur l'utilisateur, appelée également la collecte de données.
2. La deuxième consiste à bâtir une matrice ou un modèle utilisateur contenant l'information recueillie, également appelé un profil utilisateur.
3. La troisième consiste à extraire à partir de cette matrice une liste de recommandations [60].

1.4. 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 CiteSeer, 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 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 [12].

1.5. Forme de collecte des données

1.5.1. Feedback explicite - Filtrage dit actif ou réactif

Ce type de collecte repose sur le fait que l'utilisateur indique explicitement ses intérêts suite à une demande du système. Par exemple : demander à un utilisateur de commenter, taguer/étiqueter, noter, lier ou encore ajouter comme favoris des contenus (objets, articles qui l'intéressent.

L'avantage de ce genre de systèmes est qu'ils sont faciles à appliquer et ne requièrent aucune connaissance approfondie du domaine de la part de l'utilisateur. Toutefois, les critiques demeurent une arme à double tranchant. En effet, si elles représentent des informations explicites sur les appréciations, elles nécessitent un effort et un investissement de la part de l'utilisateur quant à l'expression de ses avis et appréciations. Les informations recueillies peuvent également contenir un biais dit de déclaration [13].

1.5.2. Feedback implicite – Filtrage dit passif ou proactif

Cette collecte repose sur une observation et une analyse des comportements de l'utilisateur effectués de façon implicite dans l'application qui embarque le système de recommandation, le tout se fait en "arrière-plan", sans rien demander à l'utilisateur.

Par exemple :

- Obtenir la liste des items que l'utilisateur a consulté, les musiques écoutées, les vidéos regardées ou les produits achetés en ligne.
- Analyser la fréquence de consultation d'un contenu par un utilisateur, le temps passé sur une page.
- Analyser son réseau social, ...

L'avantage d'un système implicite est que l'utilisateur n'est plus sollicité pour fournir des informations ou des appréciations, toutes les informations sont collectées automatiquement, et les données récupérées sont a priori justes et ne contiennent pas de biais de déclaration.

Cependant, les données récupérées sont plus difficilement attribuables à un utilisateur et peuvent donc contenir des biais d'attribution, comme un utilisateur peut ne pas aimer certains livres qu'il a achetés, ou il peut les avoir achetés pour quelqu'un d'autre [13].

1.5.3. Différences d'interprétation entre les données explicites et implicites

Dans le cas des données explicites, les données manquantes restent manquantes puisqu'il n'y a aucun moyen de leur trouver une valeur alternative pertinente. En revanche, dans le cas des données implicites, les données manquantes sont prises en compte et sont initialisées à zéro plutôt que d'être ignorée. Une valeur mise à zéro ne signifie pas que l'utilisateur n'apprécie pas l'item, mais simplement que le niveau de confiance est nul puisqu'il n'y a eu aucune interaction. L'absence d'écoute d'une chanson indique par exemple que l'utilisateur n'est peut-être même pas au courant de l'existence de la chanson [13].

1.6. Matrice d'évaluation Utilisateur-Item

La matrice d'évaluation Utilisateur-Item qu'on notera R est une matrice m × n où m est le nombre d'utilisateurs et n est le nombre d'items. Dans le cas d'un feedback implicite binaire, $R_{ij} = 1$ si l'utilisateur i a interagi avec l'item j et $R_{ij} = 0$ sinon. Dans le cas d'un feedback explicite, l'utilisateur i entre une note x représentant sa satisfaction pour l'item j ($R_{ij} = x$) et $R_{ij} = 0$ s'il n'a pas mis de note.

Le système de notation peut varier d'un système à un autre. Par exemple, un système de notation très répandu est le système à *5 étoiles* : L'utilisateur met une note de *1 étoile* s'il n'a pas du tout apprécié l'item et *5 étoiles* si au contraire, il l'a beaucoup apprécié. Toutefois, une note de *3 étoiles* n'a pas forcément la même valeur d'un utilisateur à un autre. C'est pour cette raison qu'il est généralement préférable de normaliser les données avant de les exploiter (centrer les notes par utilisateur par exemple)



Figure 1.1 : Exemple d'une matrice d'évaluation Utilisateur-Item avec feedback explicite (les blancs contiennent des zéros)

Comme on peut le voir dans **la figure 1.1** : une particularité de cette matrice est qu'elle contient beaucoup de zéros, elle est donc creuse. Dès qu'un système commence à avoir un nombre considérable d'utilisateurs et d'items, il devient impossible pour les utilisateurs d'interagir avec tous les items. Pour donner un exemple concret, la matrice d'évaluation du dataset *MovieLens-100k* à une densité de seulement 6.31%. Nous définissons la densité d'une matrice creuse comme étant le nombre d'interactions utilisateur-item I divisé par la taille de la matrice par la formule suivant :

$$Densite\ (R) = \frac{I}{mn} \qquad (1.1)$$

1.7. Problématique de la recommandation

La formalisation courante du problème de la recommandation se réduit à l'estimation de l'évaluation[1] d'un item qui n'a pas encore été vu par l'utilisateur, en se basant sur ses évaluations passées d'autres items ainsi que d'autres informations additionnelles dépendantes du système.

Si le système de recommandation arrive à bien estimer cette évaluation, il pourra recommander aux utilisateurs des items pertinents en les ordonnant selon l'évaluation estimée. Plus formellement, soit U l'ensemble des utilisateurs et I l'ensemble des items. Nous notons I_u l'ensemble des items avec qui l'utilisateur u a interagi et U_i l'ensemble des utilisateurs qui ont interagi avec l'item i.

On a discuté de la recommandation d'un item, mais on peut étendre cette idée pour la recommandation de plusieurs items : pour recommander les k-meilleurs items (top-k recommandation) à un utilisateur u, il suffit de choisir les k items ayant les meilleures évaluations.

1.8. Approches des systèmes de recommandation

Il existe principalement deux approches pour résoudre la problématique de la recommandation : l'approche basée contenu et l'approche collaborative.

1.8.1. Approche basée contenu

Dans cette approche, les utilisateurs et les items sont représentés par des vecteurs dans R^n. Chaque composante de ces vecteurs est une caractéristique particulière de l'utilisateur/item. Pour recommander un item à un utilisateur u, on calcule la similarité entre le vecteur u et les vecteurs de tous les items qui ne sont pas dans I_u et on choisit l'item i^* qui maximise la similarité :

[1] . L'évaluation peut être un feedback implicite ou un feedback explicite.

$$i^* = \max_{i \notin I_u} sim(u, i) \quad (1.2)$$

1.8.2. Approche collaborative

Cette approche exploite la similarité des préférences entre utilisateurs (ou items) pour la recommandation. Si le système trouve qu'un utilisateur ressemble à un autre, il sera plus susceptible de lui recommander les mêmes items et vice-versa. L'approche collaborative peut être séparée en deux types d'algorithmes : les algorithmes basés voisinage et les algorithmes basés modèle.

a. Algorithmes basés voisinage

Ces algorithmes exploitent le comportement des autres utilisateurs pour savoir ce que notre utilisateur pourrait apprécier. Ils peuvent trouver des personnes similaires à notre utilisateur et recommander des items qu'ils ont aimés ou recommander des items que d'autres personnes ont achetés après avoir acheté ce que notre utilisateur a acheté. La même chose peut être faite pour les items.

- **Algorithme basé utilisateur**
  Pour prédire une évaluation $\hat{r}_{u,i}$ d'un utilisateur u pour un item i. On utilise la formule suivante :

$$\hat{r}_{u,i} = \bar{u} + \frac{\sum_{u' \in U} sim(u, u')(r_{u',i} - \bar{u}')}{\sum_{u' \in U} |sim(u, u')|} \quad (1.3)$$

  Où Ū est la moyenne des évaluations de l'utilisateur u et U est l'ensemble des utilisateurs.

- **Algorithme basé item** Il utilise les items les plus similaires aux items déjà notés d'un utilisateur pour générer des recommandations.

b. Algorithmes basés modèle

Les systèmes de recommandation basés voisinage ne sont pas toujours aussi rapides et scalables[2] que nous le souhaiterions, en particulier dans le contexte de systèmes réels qui génèrent des recommandations en temps réel basées sur de très grands dataset. Pour atteindre ces objectifs, des systèmes de recommandation basés sur des modèles sont introduits. Le principe est le suivant : le système extrait certaines informations de l'ensemble de données (évaluations) et les utilisent comme modèle pour faire des recommandations sans avoir à utiliser le dataset dans sa totalité à chaque fois. Les principaux avantages de cette approche sont sa vitesse de prédiction et sa scalabilité.

Il y a un nombre important d'algorithmes basés modèle. On présente dans ce document deux d'entre eux : la décomposition en valeurs singulières (SVD) et le *Neural Collaborative Filtering* [14] basé sur le Deep Learning.

[2] La scalabilité est la capacité d'un algorithme à se généraliser sur de grands datasets efficacement

- **Décomposition en valeurs singulières (SVD)**

La décomposition en valeurs singulières est une méthode qui sert à factoriser la matrice d'évaluation Utilisateur-Item R de dimension m × n de la manière suivante :

$$R = US^T \tag{1.4}$$

Où U est une matrice dimension m × k dont chaque ligne est un vecteur caractéristique d'un utilisateur particulier, S est une matrice diagonale de dimension k × k et I une matrice de Dimension n × k dont chaque ligne est un vecteur caractéristique d'un item particulier. k est un paramètre qui représente la dimension à laquelle doivent être ramenée les vecteurs caractéristiques des utilisateurs et des items. Si on choisit k = min (m, n), la factorisation est parfaite mais elle nous est d'aucune utilité dans ce cas car, puisqu'elle est parfaite, elle mettra des zéros là où il n'y a pas d'interactions Utilisateur-Item. Donc, on prend k < min (m, n) pour obtenir une approximation exploitable.

Pour la suite, on pose U := US ce qui donne : $R = UI^T$ Donc, à partir de cette factorisation, on peut estimer l'évaluation d'un utilisateur u pour un item i en calculant le produit scalaire entre le vecteur caractéristique de u et celui de i : $eval(u, i) = U_u I_i T$.

La décomposition en valeurs singulières est obtenue en minimisant la distance entre l'évaluation réelle de l'utilisateur pour un item et l'évaluation prédite par eval (u, i) :

$$\min_{U,I} \sum_{(u,i)\in K} (\hat{r}_{u,i} - U_u I_i^T)^2 \tag{1.5}$$

où (u, i) ∈ K si et seulement si l'utilisateur u a évalué l'item i, $\hat{\boldsymbol{r}}_{\boldsymbol{u,i}}$ représente cette évaluation. On suppose que le dataset a été normalisé en centrant les évaluations par utilisateur.

On peut résoudre ce problème d'optimisation soit par des méthodes exactes (SVD), soit en utilisant la descente du gradient où les poids sont U et I

Pour améliorer les performances du système de recommandation, d'autres contraintes peuvent être ajoutées à l'équation (1.5) Pour éviter que le modèle ne fasse du sur apprentissage, on ajoute une régularisation L2 à l'équation :

$$\min_{U,I} \sum_{(u,i)\in K} (\hat{r}_{u,i} - U_u I_i^T)^2 + \lambda(\|U_u\|_2^2 + \|I_i\|_2^2) \tag{1. 6}$$

Une autre amélioration serait d'ajouter un biais spécifique à chaque utilisateur et à chaque item, car, comme dit plus haut les utilisateurs n'évalue pas les items de la même manière et de même, un item peut avoir une évaluation élevée parce qu'il est populaire. Donc, ces biais nous aident à prendre en considération ces différences :

$$\min_{U,I,b} \sum_{(u,i)\in K} \left(\hat{r}_{u,i} - U_u I_i^T - \mu - b_u - b_i\right)^2 + \lambda(\|U_u\|_2^2 + \|I_i\|_2^2 + b_u^2 + b_i^2) \tag{1.7}$$

où μ est la moyenne de toutes les évaluations.

➢ **Neural Collaborative Filtering**

Cette méthode soutient que le produit scalaire limite les modèles et ne permet pas la modélisation des interactions complexes entre les utilisateurs et les items. Elle propose donc de prédire les évaluations Utilisateur-Item en utilisant un perceptron multicouche (MLP) et une autre architecture qui se nomme *Generalized Matrix Factorization*. On procède d'abord à l'entraînement de ces deux architectures de manière indépendante, cette phase se nomme le pré-entraînement, et par la suite, on entraîne la combinaison de ces deux architectures qu'on nomme MLP-GMF en gardant les poids de la phase de pré-entraînement. Cette manière de faire donne de meilleurs résultats que si on initialisait les poids de MLP-GMF aléatoirement [14].

- **Perceptron multicouche (MLP) non-linéaire**

Ce MLP qu'on note φ prends en entrée la concaténation des vecteurs caractéristiques (qui sont des poids à ajuster) de l'utilisateur u et de l'item i en question et nous retourne en sortie un scalaire qui représente l'évaluation notée $r_{u,i}$[3], c'est-à-dire, $\varphi(u, i) = r_{u,i}$, $\varphi : U \times I \rightarrow [0, 1]$.

Ce MLP est une succession de couche FC avec comme fonction d'activation ReLU sauf la dernière couche qui a comme fonction d'activation sigmoïde pour avoir un ensemble d'arrivée dans [0, 1].

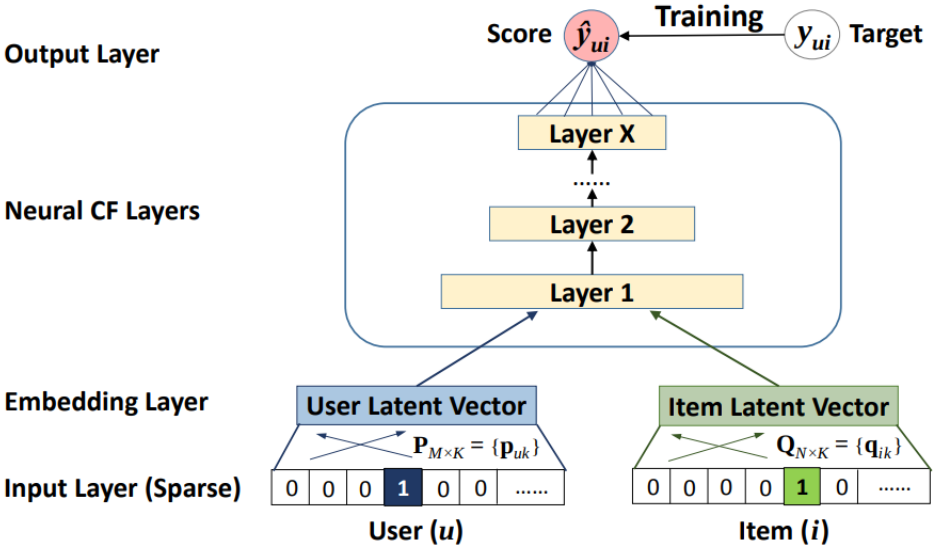


Figure 1.2 : Architecture du perceptron multicouche du Neural Collaborative Filtering

a. Matrix Factorisation (GMF)

Cette architecture est un MLP à trois couches (en comptant la couche d'entrée) qu'on note ψ. Il prend en entrée la concaténation des vecteurs caractéristiques de l'utilisateur u et de

[3] Cette méthode a été implémentée pour des feedbacks implicites mais il peut être facilement généraliser pour des feedbacks explicites

l'item i et nous retourne un scalaire qui représente l'évaluation $r_{u,i}$ ($\psi : U \times I \rightarrow [0, 1]$). Toutes les couches de cette MLP sont linéaires, c'est-à-dire que la fonction d'activation des couches est la fonction identité (f(x) = x) sauf pour la dernière couche qui est une sigmoïde.

b. Combinaison des deux architectures

Le MLP non-linéaire extrait les propriétés non-linéaires des interactions utilisateur-item alors que GMF en extrait les propriétés linéaires. Le but de la combinaison est de tirer profit des deux approches afin de maximiser les performances.

Après avoir entrainer les deux architectures séparément, on leur enlève la dernière couche sigmoïde : φ devient $\varphi' : U \times I \rightarrow R^k$ et ψ devient $\psi' : U \times I \rightarrow R^l$ où R^k et R^l sont respectivement l'ensemble d'arrivée de l'avant dernière couche du MLP non-linéaire et l'avant dernière couche de GMF. On note $\phi : U \times I \rightarrow [0, 1]$ la combinaison des deux architectures où :

$$\varphi(u,i) = sigm\left(\begin{bmatrix} \alpha\psi'(u,i) \\ (1-\alpha)\phi'(u,i) \end{bmatrix}.W\right) \qquad (1.8)$$

est la concaténation du vecteur x avec le vecteur y.

$\begin{bmatrix} x \\ y \end{bmatrix}$: est la concaténation du vecteur x avec le vecteur y.

W : est un vecteur dans R_{k+l} qui est un poids dans la descente de gradient.

sigm : est la fonction sigmoïde.

α est un paramètre déterminant le compromis entre les deux architectures.

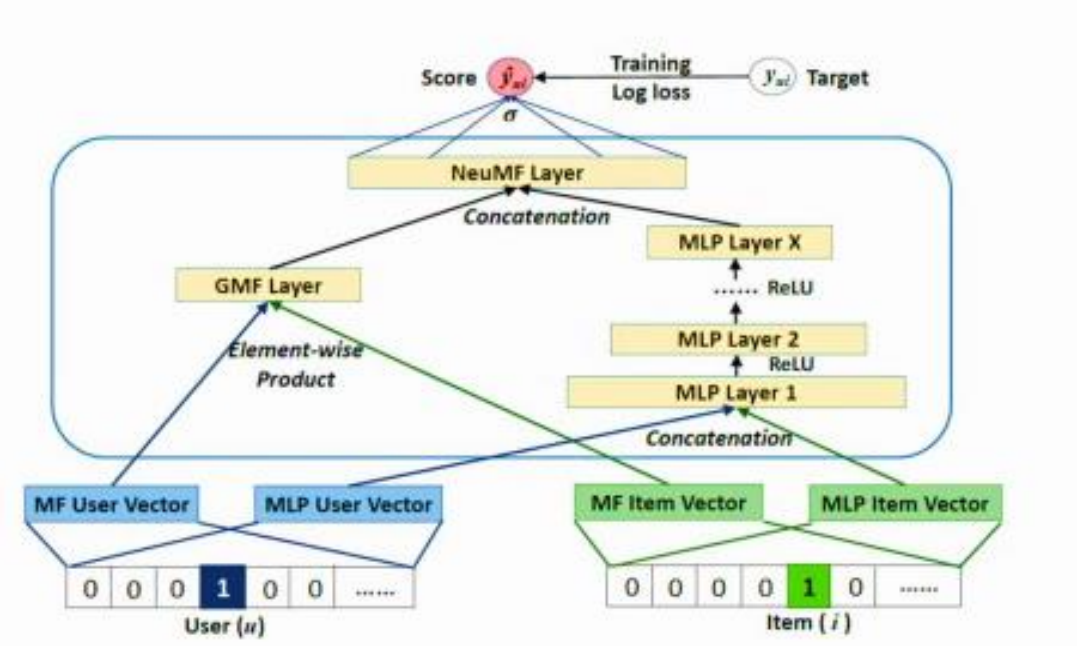


Figure 1.3 : Combinaison du MLP non-linéaire et GMF

Il a été démontré empiriquement, par des expériences [14], que le *Neural Collaborative Filtering* est supérieur aux techniques de factorisation matricielle et aux algorithmes basés voisinage sur des datasets populaires (*Movielens* [15], *Pinterest* [16]).

Il y a donc un intérêt à utiliser le Deep Learning pour modéliser les interactions Utilisateur-Item.

I.9. 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 1.4** Nous citons ici quelques classifications [17].

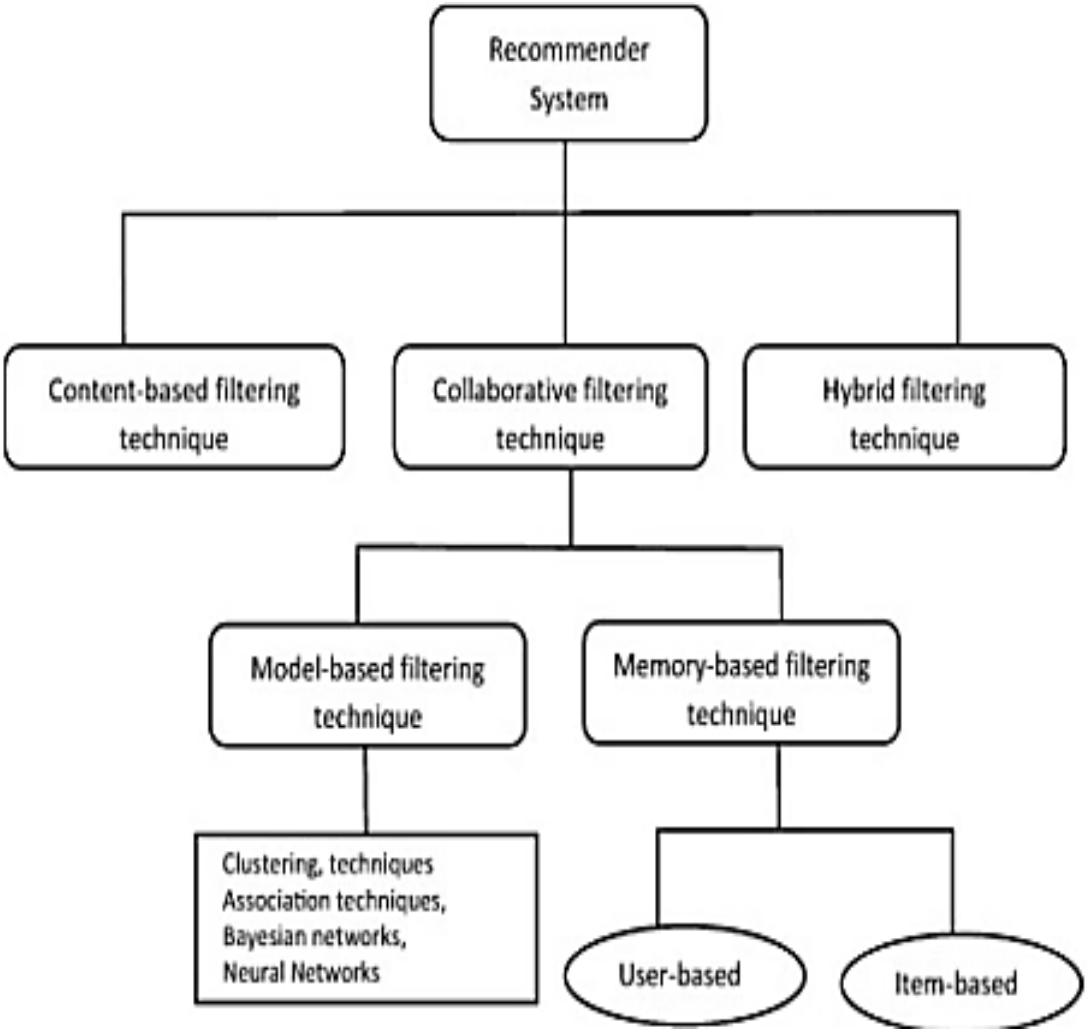


Figure 1.4 : Principales classifications des systèmes de recommandations

1.10. Types des recommandations

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

1.10.1. 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 [18].

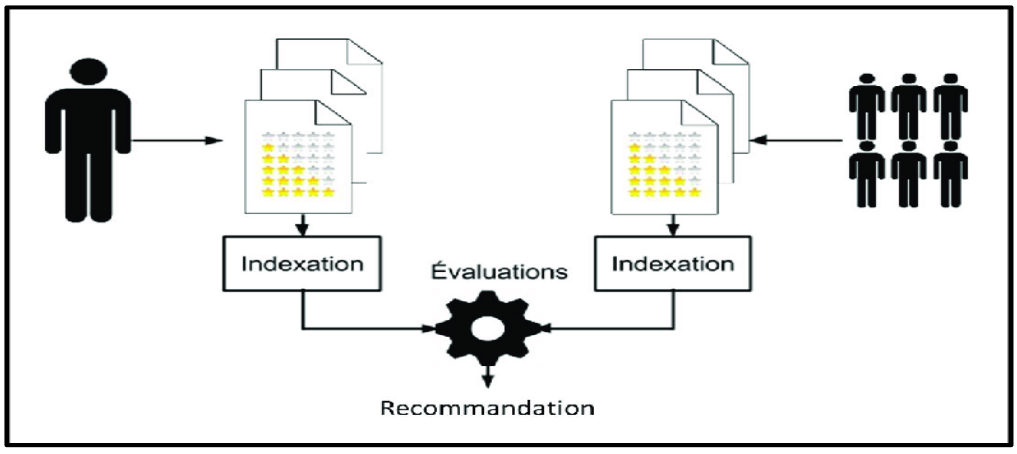


Figure 1.5 : Recommandation basée sur le filtrage collaboratif [18]

La figure 1.6 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 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, ... 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, ... [18].

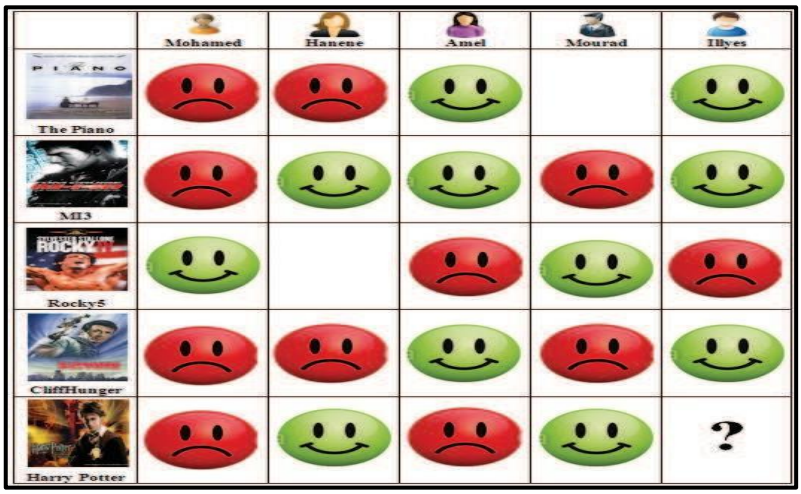


Figure 1.6 : Exemple de recommandation basé sur le filtrage collaboratif [18]

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.

a. 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.

- **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 :

$$Sim(A,B) = \frac{\sum_{i=1}^{n} v_{Ai} \times v_{Bi}}{\sqrt{\sum_{i=1}^{n} v_{Ai}^2} \times \sqrt{\sum_{j=1}^{n} v_{Bi}^2}} \qquad (1.9)$$

$\boldsymbol{n}$: 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 :

$$sim(A,B) = \frac{\sum(v_{Ai} - \bar{v}_{Ai})(v_{Bi} - \bar{v}_{Bi})}{\sqrt{\sum(v_{Ai} - \bar{v}_{Ai})(v_{Bi} - \bar{v}_{Bi})^2}} \qquad (1.10)$$

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

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

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

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

$\bar{\boldsymbol{v}}_{\boldsymbol{Bi}}$: 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 ($P_{A,}$) est calculée à l'aide de la somme pondérée des estimations des voisins les plus proches qui ont déjà estimé l'item j :

$$P_{A,j} = \frac{\sum_{i=1}^{n} Sim(A,i) \times v_{i,j}}{\sum_{i=1}^{n} Sim(A,i)} \qquad (1.11)$$

n : Nombre d'utilisateurs présents dans le voisinage de A, ayant déjà voté sur l'item

j : Vote de l'utilisateur i pour l'objet j .

- **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**

$$sim(i,j) = \frac{(v_{Ai} - \bar{v}_A)}{\sqrt{\sum_{A=1}^{m} (v_{Ai} - \bar{v}_A)^2}} \times \frac{(v_{Aj} - \bar{v}_A)}{\sqrt{\sum_{A=1}^{m} (v_{Aj} - \bar{v}_A)^2}} \qquad (1.12)$$

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

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

- **Corrélation de Pearson**

$$sim(A,B) = \frac{\sum_{A=1}^{m}(v_{Aj}-\bar{v}_A)\times(v_{Bj}-\bar{v}_A)}{\sum_{A=1}^{m}(v_{Ai}-\bar{v}_A)^2\times\sum_{A=1}^{m}(v_{Bj}-\bar{v}_A)^2} \quad (1.13)$$

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

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

$\bar{\boldsymbol{v}}_{\boldsymbol{Bi}}$: 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 :

$$P_{A,i} = \bar{v}_i + \frac{\sum_{j=1}^{m} Sim(A,B)\times(v_{A,j}-\bar{v}_j)}{\sum_{j=1}^{m}|Sim(i,j)|} \quad (1.14)$$

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

vA,j : vote de l'utilisateur A pour l'objet j.

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

$|(i, j)|$: Similarité moyenne.

b. 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.

1.10.2. 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 [19].

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

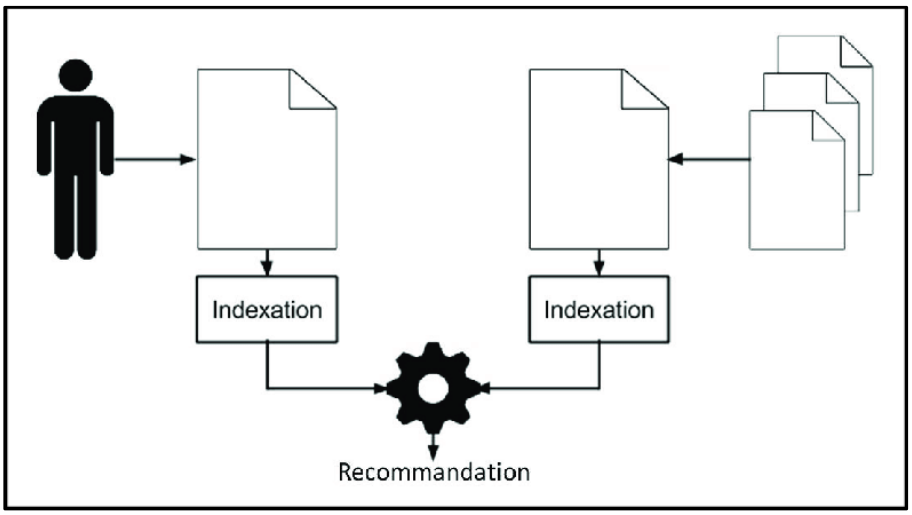


Figure 1.7 : Recommandation basée sur le contenu [19]

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

a. 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, ... 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".

b. 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 [20].

1.10.3. 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 :

- Appliquer séparément le filtrage collaboratif et autres techniques de filtrage pour générer des recommandations candidates.
- 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 [20].

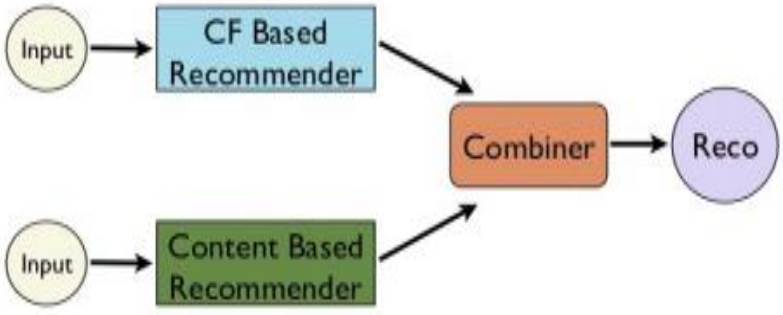


Figure 1.8 : Filtrage hybride

1.10.4. Autres types

a. 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 1.9** 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.

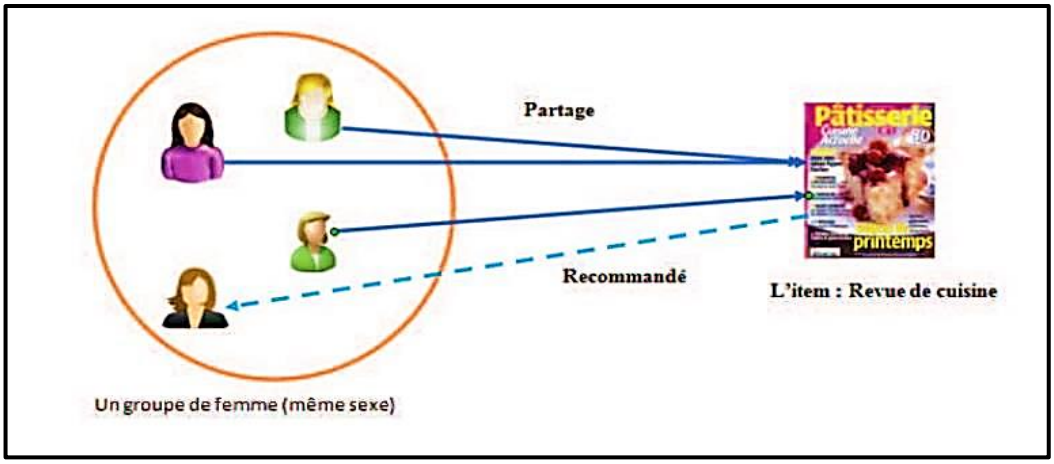


Figure 1.9 : Recommandation démographique [20]

b. 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 ») [21].

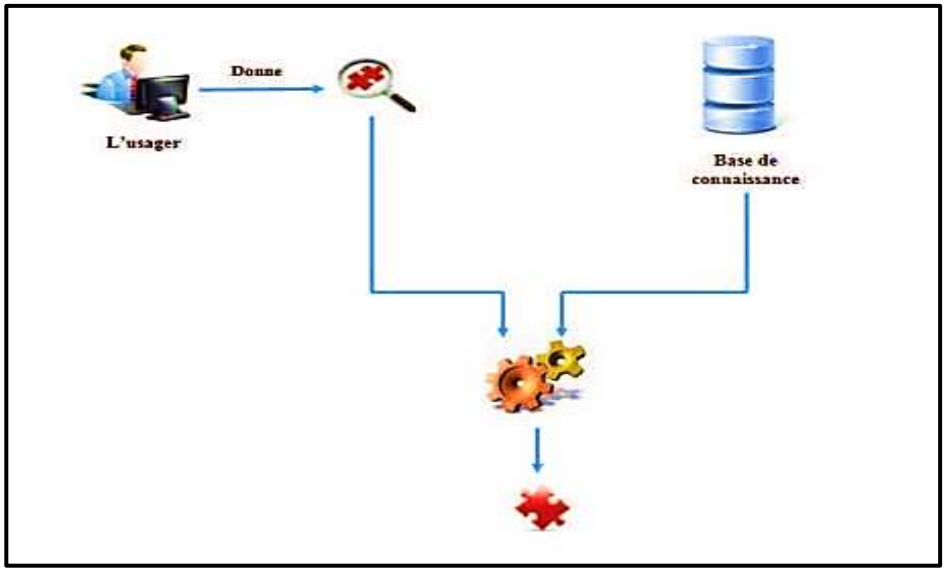


Figure 1.10 Recommandation à base de connaissance [21]

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

- **Recommandation 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 [22].

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.

- **Recommandation 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.

c. 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'il

spart agent aussi les mêmes goûts dans le futur. Par exemple sur **la figure 1.11**, 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 [23].

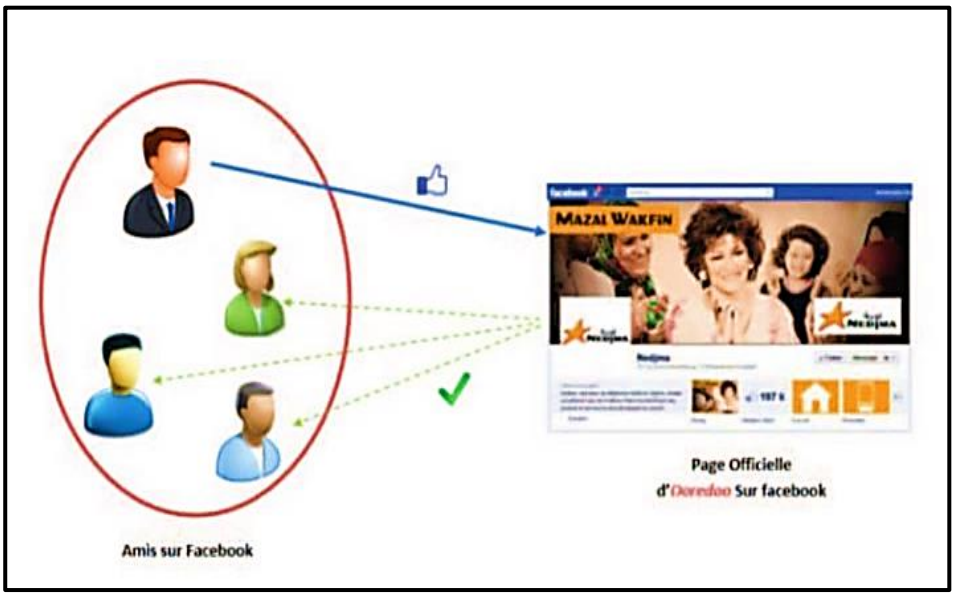


Figure 1.11 Recommandation communautaire [23]

1.11. 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.

Recommendation Approach	Recommendation Technique	
	Heuristic-based	Model-based
Content-based	Commonly used techniques: • TF-IDF (information retrieval) • Clustering Representative research examples: • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997	Commonly used techniques: • Bayesian classifiers • Clustering • Decision trees • Artificial neural networks Representative research examples: • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999, 2000 • Zhang et al. 2002
Collaborative	Commonly used techniques: • Nearest neighbor (cosine, correlation) • Clustering • Graph theory Representative research examples: • Resnick et al. 1994 • Hill et al. 1995 • Shardanand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado & Ishii 1999 • Pennock & Horwitz 1999 • Sarwar et al. 2001	Commonly used techniques: • Bayesian networks • Clustering • Artificial neural networks • Linear regression • Probablistic models Representative research examples: • Billsus & Pazzani 1998 • Breese et al. 1998 • Ungar & Foster 1998 • Chien & George 1999 • Getoor & Sahami 1999 • Pennock & Horwitz 1999 • Goldberg et al. 2001 • Kumar et al. 2001 • Pavlov & Pennock 2002 • Shani et al. 2002 • Yu et al. 2002, 2004 • Hofmann 2003, 2004 • Marlin 2003 • Si & Jin 2003
Hybrid	Combining content-based and collaborative components using: • Linear combination of predicted ratings • Various voting schemes • Incorporating one component as a part of the heuristic for the other Representative research examples: • Balabanovic & Shoham 1997 • Claypool et al. 1999 • Good et al. 1999 • Pazzani 1999 • Billsus & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002	Combining content-based and collaborative components by: • Incorporating one component as a part of the model for the other • Building one unifying model Representative research examples: • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Tableau 1.1 Avantages et inconvénients des méthodes de recommandation

1.12. Métrique d'évaluation

1.12.1. Mesures statistiques de précision

Consistent à évaluer la différence existante entre les notes prédites et les notes réellement attribuées par les utilisateurs.

a. MAE : Erreur Absolue Moyenne

La MAE *(Mean Absolute Error*), est la mesure la plus populaire pour l'évaluation des systèmes de recommandation. Son principe est pour chaque paire <note-prédiction>, calcule la moyen d'erreur absolue entre les notes prédites *Pred* (u_a, i) et les notes réelles des utilisateurs v (u_a, i). n Représente le nombre d'items prédits présents dans le corpus test.

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |v(u_a,i) - \boldsymbol{Pred}\,(u_a,i)|}{n} \qquad (1.15)$$

Plus la valeur de MAE est faible, plus les prédictions sont précises et le système de recommandation est performant **[68]**.

Il existe d'autres mesures statistiques de précision évaluant les prédictions numériques, notamment : "Root Mean Squared Error", "Mean Squared Error" qui attribue un poids plus important aux prédictions dont l'erreur est élevée, par rapport aux prédictions précises **[68]**.

b. RMSE : Racine de l'Erreur Quadratique Moyenne

La RMSE (*Root Mean Squared Error*) est devenue la mesure de référence pour la recommandation de films. Comparée à la MAE, la RMSE pénalise les écarts importants, ainsi un écart de deux (2) points aura pour effet d'augmenter la somme des erreurs par deux (2) au niveau de la MAE, alors qu'il augmentera la somme par quatre (4) au niveau de la RMSE. On définit par T l'ensemble des couples (u; i).

$$\mathbf{RMSE} = \sqrt{\frac{\sum_{(u,i)\in T} (v(u_a,i) - \boldsymbol{Pred}\,(u_a,i))^2}{|T|}} \qquad (1.17)$$

Plus la RMSE est faible, meilleure est la précision du système [69].

c. HMAE

Les systèmes de recommandations ont pour objectif de calculer les prédictions des notes manquantes concernant le maximum de pairs <utilisateur-item>. Une fois ces prédictions calculées, les items ne sont pas tous recommandés par la suite aux utilisateurs. En effet, seuls les items ayant les valeurs de prédiction les plus élevées sont proposées. Dans ce cas, l'erreur concernant les items ayant de faibles valeurs de prédiction n'est pas utile quant à l'évaluation de la performance des systèmes de recommandation, tandis que l'erreur relative aux items ayant des notes prédites élevées est d'une grande importance en matière d'évaluation [70].

La HMAE permet en effet d'évaluer les "faux positifs" qui représentent les items juges pertinents par le système, alors qu'ils ne le sont pas réellement. Avec la détection des faux

Positifs, le système ne risque pas d'être pénalisé suite à une recommandation d'item non pertinent susceptible d'engendrer une insatisfaction chez l'utilisateur.

Selon l'équation qui calcule cette mesure, HMAE est similaire à la MAE, mais elle a la particularité de considérer uniquement les prédictions élevées $Pred'(u_a, i)$, est m représente ici le nombre d'items prédits avec des valeurs élevées [70].

$$\mathbf{HMAE} = \frac{\sum_{i=1}^{m} |v(u_a,i) - \boldsymbol{Pred}(u_a,i)|}{m} \quad (1.17)$$

Plus la valeur de HMAE est faible, plus le système de recommandation est performant [70].

d. Top-N

Multiples systèmes de recommandation ne calculent pas la prédiction de la valeur d'un vote soit, parce qu'ils ne disposent que de votes binaires ou unaires soit, parce que l'algorithme utilisé ne calcule pas de prédictions de votes. De tels systèmes recommandent plutôt à l'utilisateur courant une liste contenant N items susceptibles de l'intéresser. On parle alors de recommandation Top-N. La mesure de la précision revient alors à mesurer la capacité du système à décider si un item est pertinent ou pas pour l'utilisateur courant. Si les votes sont numériques, il est primordial de définir un seuil de vote permettant de classer les items de la base de test en items pertinents et items non pertinents. Ainsi, pour des votes définis sur une échelle de 1 à 5, on peut considérer que les items ayant un vote ≥4 sont pertinents pour l'utilisateur et ceux ayant un vote < 4 sont non pertinent [69].

1.12.2. Mesures permettant l'aide à la décision

Les mesures permettant l'aide à la décision consistent à évaluer jusqu'à quel point le système de recommandation peut recommander des items potentiellement pertinents pour l'utilisateur (les items susceptibles d'être très appréciés) [71].

En d'autres termes, ces mesures évaluent la pertinence des recommandations en calculant, dans une liste de recommandation, la proportion d'item qui sont effectivement utiles et pertinents pour l'utilisateur actif. Pour les besoins d'évaluation en matière d'aide à la décision, les appréciations ou les notes des utilisateurs doivent être transformées dans le cadre d'une échelle binaire ("Aime" ou "Aime pas") afin de distinguer les items pertinents de ceux qui ne le sont pas, pour un utilisateur donné [70].

Les mesures principales permettant l'aide à la décision sont notamment :

a. Précision

La précision évaluée si un item sélectionné par un utilisateur est réellement perçu comme étant pertinent par ce même utilisateur [72].

Un item sélectionné représente un item qui est proposé par le système de recommandation à l'utilisateur actif et qui est contenu en même temps dans le corpus test [70].

La précision est calculée sur la base de l'équation comme étant le rapport entre le nombre d'items pertinents sélectionné N_{ps} et le nombre d'items sélectionnés par un utilisateur actif N_s.

$$\boldsymbol{P = \frac{N_{ps}}{N_s}} \quad (1.18)$$

La précision générale du système de recommandation correspond ainsi à la moyenne des précisions calculée pour chaque utilisateur actif. Plus cette précision est élevée, plus le système de recommandation est performant [70].

Le tableau 1.5 présente les catégories d'items repartis selon l'intersection entre les listes de recommandation et les appréciations réelles des utilisateurs [73].

	Sélectionné (s)	Non sélectionné (ns)	Total
Pertinent (p)	N_{ps}	N_{pns}	N_p
Non pertinent (np)	N_{nps}	N_{npns}	N_{np}
Total	N_s	N_{ns}	N

TABLEAU 1.2 CATEGORIES D'ITEMS BASES SUR L'INTERSECTION ENTRE LISTES DE RECOMMANDATION ET PREFERENCES REELLES [73].

b. Rappel

Le rappel mesure la probabilité qu'un item pertinent soit sélectionné par l'utilisateur actif. Il est calculé sur la base de l'équation comme étant le ratio entre le nombre d'items pertinents sélectionné par l'utilisateur N_{ps} et le nombre total d'items pertinents disponibles N_{ps} [73].

$$\mathbf{R= \frac{N_{ps}}{N_s}} \quad (1.19)$$

Comme pour la précision, le rappel relatif à la totalité du système est évalué comme étant la moyenne des rappels calculée individuellement **[70]**.

a) F-mesure :

Autre mesure qui combine entre la précision et le rappel comme une moyenne harmonique, la qualité des recommandations est parfaite lorsque les scores de précision et de rappel sont similaires **[70]**.

$$\mathbf{F\text{-}MESURE=\frac{2PR}{P+R}} \quad (1.20)$$

1.12.3. Couverture

La couverture mesure la capacité du système à fournir des recommandations. En FC basé sur la mémoire, la couverture peut être évaluée par rapport à la capacité du système de recommandation à générer des prédictions pour toutes les notes manquantes au niveau de la

matrice de notes “Utilisateur x Item”. Elle peut être également évaluée en prenant en considération uniquement les prédictions contenues dans le corpus test [70].

1.12.4. Temps de calcul

Le temps de calcul peut être un facteur important dans l’évaluation de la performance d’un système de recommandation. Il s’agit de calculer le temps réel pour évaluer le temps requis pour l’exécution des algorithmes et l’obtention des résultats escomptés [70]

1.13. Conclusion

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.

Chapitre 2
Apprentissage Automatique

2.1. 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.

2.2. Notions de base

2.2.1. 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 [24].

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.

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

- **La classification** [25]

La classification est un type d'apprentissage supervisé, qui identifié la catégorie à laquelle un nouvel élément appartient sur la base d'un dataset d'entrainement de données contenant des observations dont la catégorie est connue.

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 sur apprentissage.
- **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, 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.

- **La régression** [26]

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.

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 Ridge** : 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).

2.2.2. Apprentissage profond

a. 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 [27].

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 15 millions d'images toutes catégorisées en fonction de leurs caractéristiques propres (véhicules, animaux, ...) [27].

En 2012 le Deep Learning est remis 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 [27].

b. 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 petites à petit à travers chaque couche avec une intervention humaine [28].

La Figure 2.1 nous montre un schéma illustratif de deep learning avec les multiples couches.

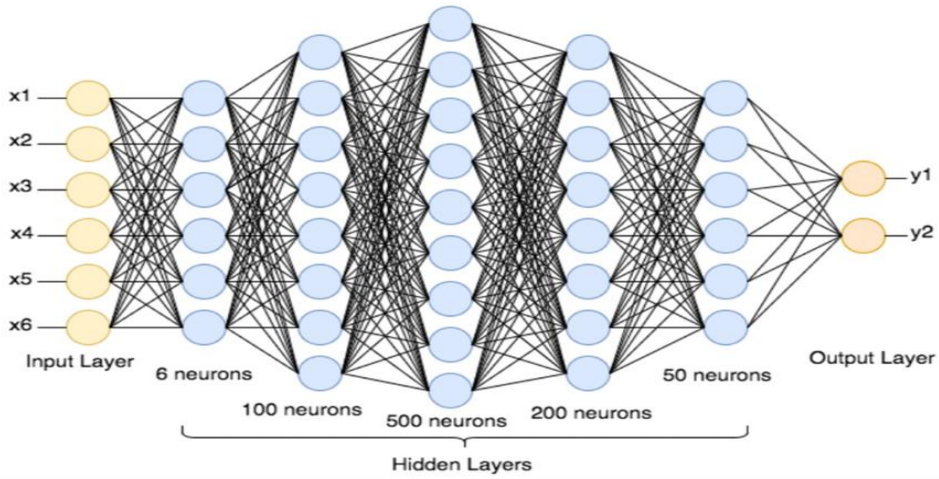


Figure 2.1 : Illustratif de deep learning avec plusieurs couches

c. 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 data.
- S'adapter à plusieurs types de problème.
- Extraire les caractéristiques de façon automatique.

La Figure 2.2 ci-dessous explique la différence entre le machine learning 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.

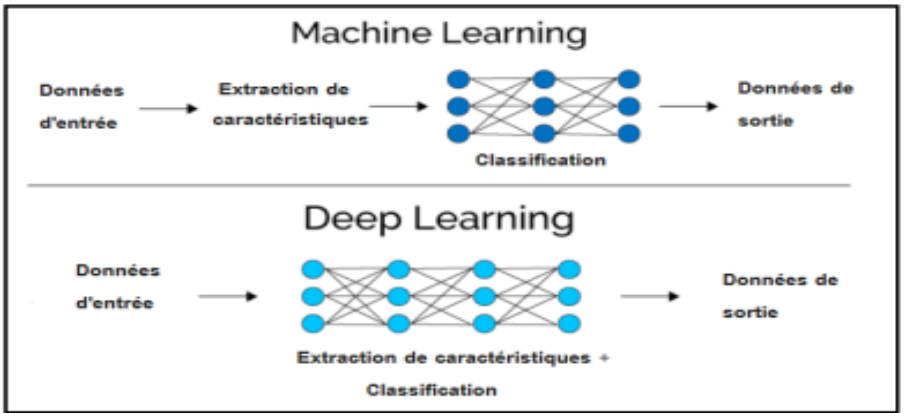


Figure 2.2 : La différence entre l'apprentissage profond et l'apprentissage automatique

d. Objectif de l'apprentissage profond

L'apprentissage profond a pour objectif d'explorer comment les ordinateurs peuvent exploiter les données pour développer des fonctionnalités, utiliser les données disponibles pour apprendre à bien décider, à acquérir de bonnes connaissances pour d'être capable de donner de bonnes réponses sûres de nouveaux exemples, inconnus du système, en généralisant les exemples connus, pour apprendre mieux [29, 30].

Il est nécessaire de donner des données prétraitées, par exemple, on décompose les images en formes simples, qui seront ensuite données au réseau de neurones artificiels afin qu'il tente de les reconnaître. L'extraction automatique de caractéristiques est un autre des grands avantages de l'apprentissage en profondeur [31, 32].

2.3. Réseaux de neurones

2.3.1. 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 [33] 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 [34] 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 [35], 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 [36] 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 [37].

2.3.2. Définition

Réseau de neurones (Neural Network) : C'est une composition de couches. Pour la phase d'entraînement, les données vont se propager de couche en couche jusqu'à la dernière couche. Suite à cela, on calcule le coût des labels en sortie par rapport aux labels du training set. Ensuite, on calcule le gradient et on met à jour les poids. C'est le même principe que la régression vue précédemment.

Une autre définition de réseau neurones : 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 de 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 [38].

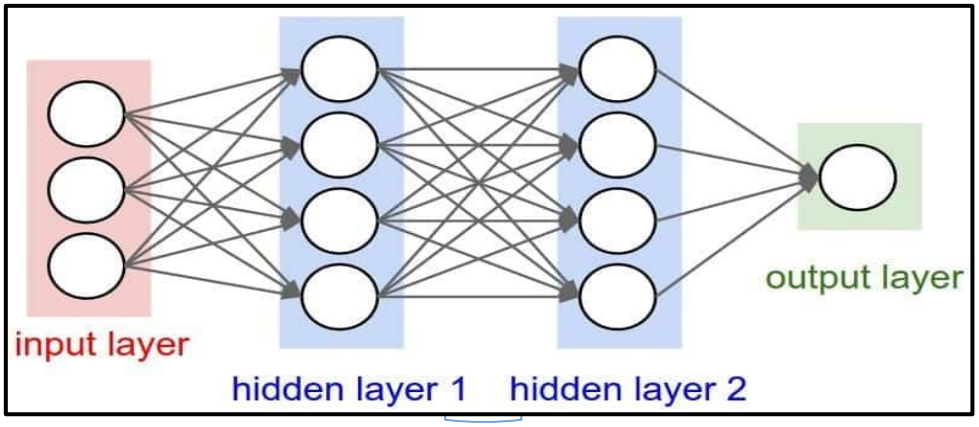


Figure 2.3 : Illustration d'un Réseau de Neurones

2.3.3. 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 **figure 2.4 (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 2.4 (b)**). Good fellow Bengio et Courville [16] 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.

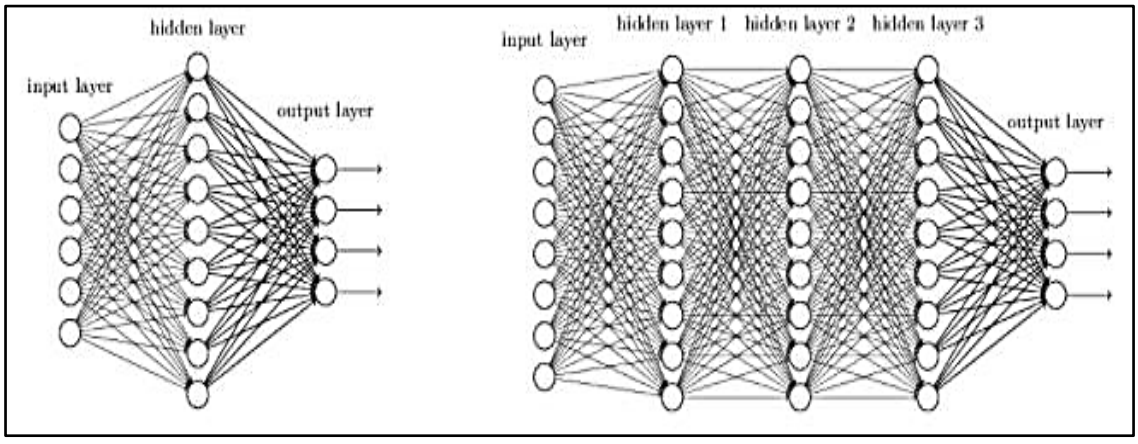


Figure 2.4 : Réseau de neurone artificiel (a) et profond (b)

2.3.4. 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 2.5,** 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 [40].

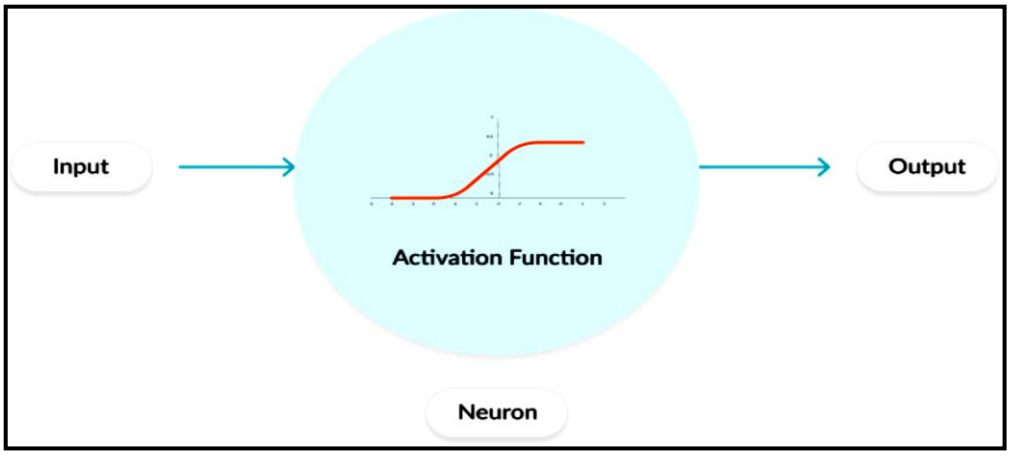


Figure 2.5 : Structure simplifié d'un neurone artificiel [41]

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 [42].

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 :

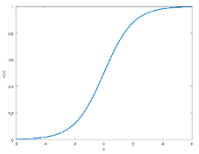
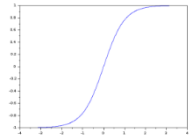
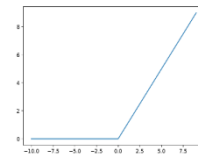
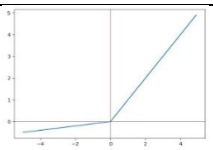
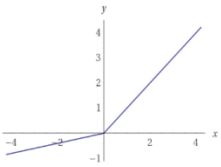
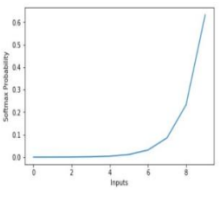
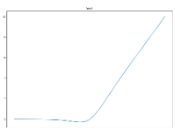
Noms	Description	Equation	Graphe
Sigmoïde	Elle génère des valeurs comprises entre zéro et un. Pour des valeurs très élevées ou basses des paramètres d'entrée, le réseau peut être très lent à atteindre une prédiction.	$(x) = 1/(1 + e\text{-}x)$	
TanH	Centrée sur zéro, ce qui facilite la modélisation d'entrées fortement négatives, fortement positives ou neutres.	$(x) = 2/(1 + e\text{-}2x\text{-}\ 1)$	
ReLu	Très efficace en termes de calcul mais n'est pas en mesure de traiter des entrées qui approchent de zéro ou négatives.	$(x) = \{x0\ si\ si\ xx <\geq 0$ 0	
Leaky ReLu	Petite pente positive dans sa zone négative, lui permettant de traiter des valeurs nulles ou négatives.	$(x) = \{0{,}01\ x\ for\ x\ for\ x \geq x <0\ 0$	
Parametrric 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.	$(x) = \{\alpha x\ x\ si\ sixx \geq <0$	
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.	$(x) = exi/\sum k\ i{=}1\ exk$ 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.	$f(x) = x.$ Sigmoïde(x)	

Tableau 2.1 : Les fonctions d'activation

2.3.5. 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.

a. 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 rétropropagation (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.

b. 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'hyperparamétre	Description
Hyper paramètres liés à la structure du réseau neuronal	Nombre de Couches Cachées	Il s'agit de cours de perceptron, là où les relations entre les variables vons être mises en exergues, le nombre a utilisé 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.
Hyper paramètres 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 2.2 : Liste des hyperparamètres

c. 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 [44] a montré que l'utilisation de valeurs hyperparamétriques aléatoires est en fait plus efficace que la recherche manuelle ou la recherche dans 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 [44].

2.3.6. 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.

a. 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) [45].

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.

Les réseaux de neurones de type CNN sont utilisés notamment :

- Dans la reconnaissance faciale.
- Pour identifier et classer les objets du quotidien en images.
- Pour alimenter la vision dans les robots et les véhicules autonomes.
- Pour reconnaître les scènes et suggérer des légendes pertinentes.

La figure 2.6 suivante montre l'architecture d'un réseau de neurone de type CNN.

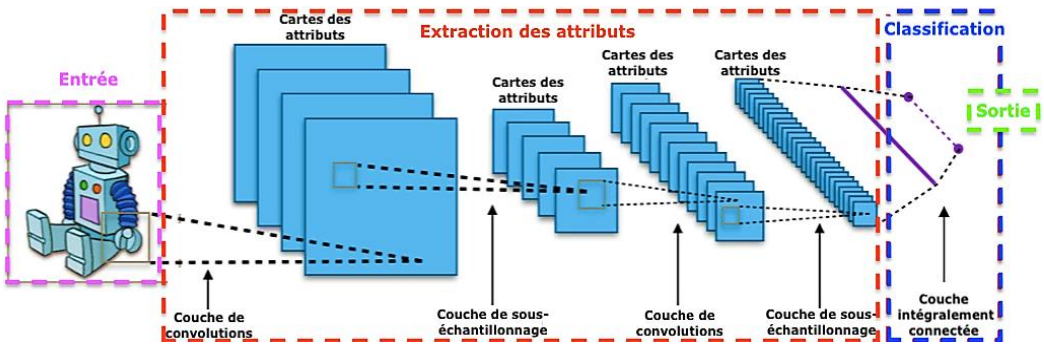


Figure 2.6 : Architecture standard d'un réseau CNN

b. Les réseaux 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 [46].

La figure 2.7 ci-dessous 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 [46].

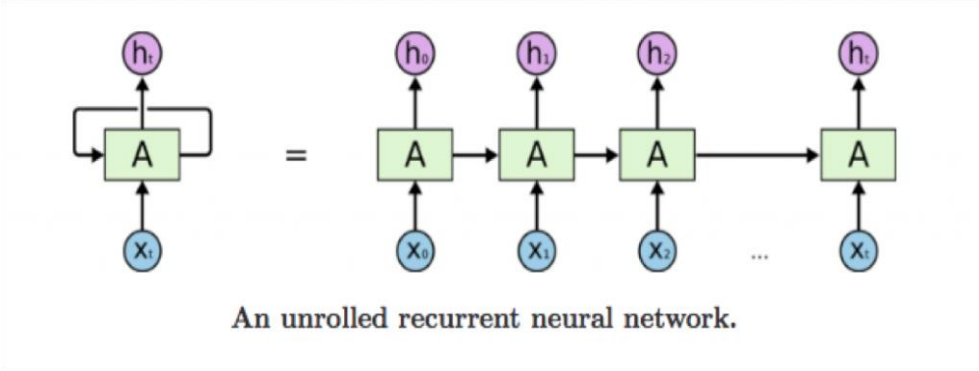


Figure 2.7 : Réseau de neurone déroulé

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.

c. Les modèles Auto-encodeur (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 [47].

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 2.8 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 (encoder représentation), puis décode cette représentation (décoder), afin de retrouver les données de départ (output) [47].

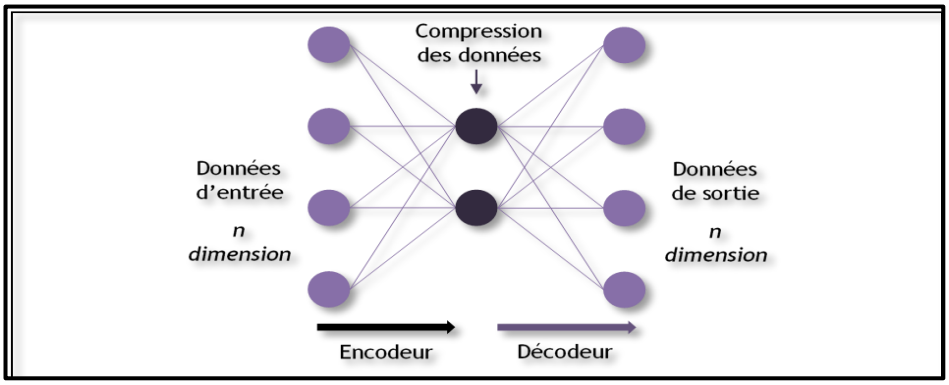


Figure 2.8 : Auto-encodeur simple

d. Les modèles Perceptron multicouches (MLP)

Le Perceptron multicouche est un Classifieur linéaire de type réseau neuronal formel organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement ; il s'agit donc d'un réseau de type feedforward. Chaque couche est constituée d'un nombre variable de neurones, les neurones de la couche de sortie correspondant toujours aux sorties du système. Le MLP (perceptron multicouche) est composé de couches successives [109] :

- Deux entrées (dans le cas de la couche 1), trois entrées sinon
- Un biais b variable.
- Un système de poids liées à ces entrés sachant que le dernier poids correspond au biais et est toujours égal à -1
- Une fonction de transfert (la fonction sigmoïde)
- Une seule sortie (où sont présentées les sorties calculées par le MLP).

Il existe plusieurs méthodes pour faire l'apprentissage des MLP, parmi elles :

- algorithme de rétro
- propagation du gradient
- algorithme de gradient conjugué
- méthodes de second ordre [109]

2.3.7. 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 le tableau de ses avantages et inconvénients comme suit :

Les avantages	Les inconvénients
• 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).	• Apprentissage long (dû au nombre de paramètres souvent élevés. • Difficile de comprendre le modèle. • Non évidence du choix des hyperparamètres.

Tableau 2.3 : les avantages et les inconvénients

2.4. Apprentissage automatique et NLP

Au cours des dix dernières années, le traitement du langage naturel est devenu l'un des domaines clés de l'intelligence artificielle avec des réseaux de neurones et des techniques d'apprentissage approfondi améliorant les performances dans plusieurs tâches.

Le traitement du langage naturel est l'un des sujets le plus en vogue dans le domaine des technologies de la parole et du langage. L'apprentissage automatique peut également comprendre comment effectuer des tâches importantes du NLP [48, 49]

L'arrivée à maturité d'analyse des données avec la convergence de nouvelles technologies telles que l'intelligence artificielle, la narration automatique, l'activation vocale, le traitement de texte et d'images, a fait de Deep Learning le composant central pour l'utilisation du traitement du langage naturel dans les applications [50].

2.4.1. NLP - Natural Language Processing

a. Introduction

Auparavant, les informaticiens divisaient un langage en ses formes grammaticales, telles que des parties de discours, des phrases, ..., en utilisant des algorithmes complexes, c.-à-d. le NLP traitait des données statiques (la *figure 2.9)*. Aujourd'hui, l'apprentissage en profondeur est la clé pour effectuer les mêmes exercices [51], [52].

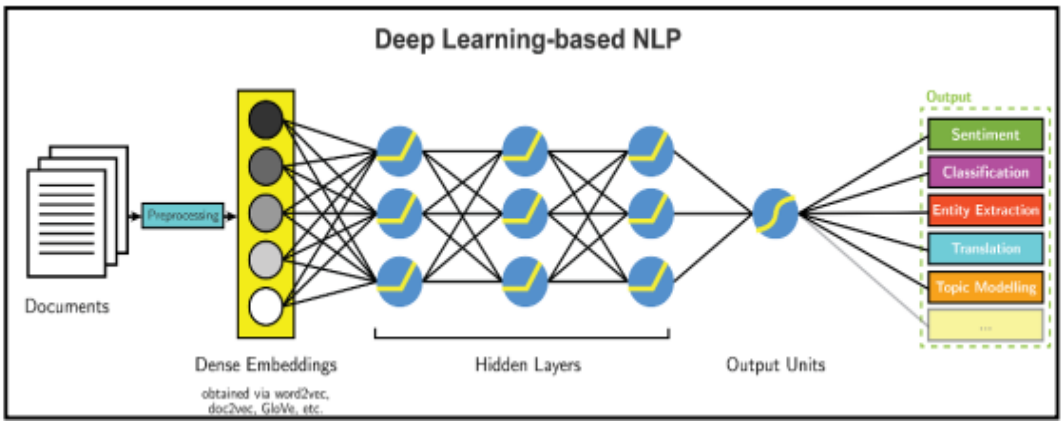


Figure 2.9 : Exemple d'application du NLP avec le Deep learning

Le langage naturel est un système complexe utilisé pour exprimer des significations. Dans ce système, les mots sont l'unité de base du sens. Comme leur nom l'indique, les vecteurs de mots sont des vecteurs utilisés pour représenter des mots et peuvent également être considérés comme des vecteurs de caractéristiques ou des représentations de mots. La technique consistant à mapper des mots sur des vecteurs réels s'appelle l'incorporation de mots. Ces dernières années, l'incorporation de mots est progressivement devenue la connaissance de base du traitement du langage naturel.

b. Définition

Le NLP pour Natural Language Processing ou Traitement du Langage Naturel est une discipline qui porte essentiellement sur la compréhension, la manipulation et la génération du langage naturel par les machines. Ainsi, le NLP est réellement à l'interface

entre la science informatique et la linguistique. Il porte donc sur la capacité de la machine à interagir directement avec l'humain.

2.4.2. Word2Vec

Cet algorithme de word embedding est parmi les plus connus. Il a été développé par une équipe de recherche de Google sous la direction de Tomas Mikolov. Il repose sur des réseaux de neurones à deux couches et cherche à apprendre les représentations vectorielles des mots composant un texte, de telle sorte que les mots qui partagent des contextes similaires soient représentés par des vecteurs numériques proches.

Word2Vec possède deux architectures neuronales, appelées CBOW et Skip-Gram, parmi lesquelles l'utilisateur peut choisir. CBOW reçoit en entrée le contexte d'un mot, c'est à dire les termes qui l'entourent dans une phrase, et essaye de prédire le mot en question. Skip-Gram fait exactement le contraire : elle prend en entrée un mot et essaye de prédire son contexte. Dans les deux cas, l'entraînement du réseau se fait en parcourant le texte fourni et en modifiant les poids neuronaux afin de réduire l'erreur de prédiction de l'algorithme.

Word2Vec possède différents paramètres, dont les plus importants sont :

- La dimensionnalité de l'espace vectoriel à construire, c'est à dire le nombre de descripteurs numériques utilisés pour décrire les mots (entre 100 et 1000 en général).
- La taille du contexte d'un mot, c'est à dire le nombre de termes entourant le mot en question (les auteurs suggèrent d'utiliser des contextes de taille 10 avec l'architecture Skip-Gram et 5 avec l'architecture CBOW).

Étant donné que Word2Vec n'est composé que de deux couches, cet algorithme est rapide à entraîner et à exécuter, ce qui se révèle être un avantage important par rapport à d'autres méthodes de word embedding [53].

a. CBOW (Continuous Bag of Words)

Cette architecture est très similaire à un réseau de neurones à anticipation. Cette architecture de modèle essaie essentiellement de prédire un mot cible à partir d'une liste de mots de contexte. L'intuition derrière ce modèle est assez simple, étant donné une phrase "Have a great day", nous choisirons notre mot cible comme étant "a" et nos mots de contexte comme ["have", "grey", "day"]. Ce que ce modèle va faire, c'est prendre les représentations distribuées des mots de contexte pour essayer de prédire le mot cible [53].

b. Continuos Skip-Gram Model

Le modèle de saut de gramme est un simple réseau de neurones avec une couche cachée entraînée afin de prédire la probabilité qu'un mot donné soit présent lorsqu'un mot d'entrée est présent. Intuitivement, vous pouvez imaginer que le modèle skip-gram est

l'opposé du modèle CBOW. Dans cette architecture, il prend le mot courant en entrée et essaie de prédire avec précision les mots avant et après ce mot courant. Ce modèle essaie essentiellement d'apprendre et de prédire les mots de contexte autour du mot d'entrée spécifié.

Sur la base d'expériences évaluant la précision de ce modèle, il a été constaté que la qualité de la prédiction s'améliore compte tenu d'une large gamme de vecteurs de mots, mais cela augmente également la complexité de calcul. Le processus peut être décrit visuellement comme indiqué ci-dessous [53].

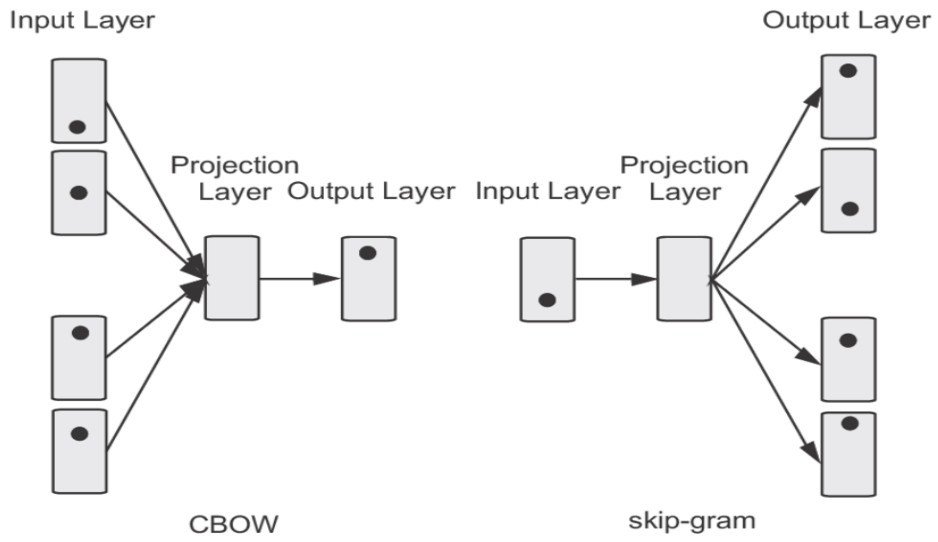


Figure 2.10 : Illustration de l'Architecture de modèle : CBOW et Skip-gram

2.5. Présentation de quelques travaux basés sur les systèmes de recommandation et machine learning

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

2.5.1. 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, ... 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).

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 1.9).
- Corrélation de Pearson (voir formule 1.10).

Une fois que la similarité a été calculée, on calcule la prédiction (voir formule 1.12) afin de prédire la note de l'utilisateur cible pour un item.

2.5.2. Perceptron multicouches (MLP)

Un perceptron multicouche (multi-layer 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 multicouche. 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 Vu et Vi 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 [54].

2.5.3. Système de recommandation 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 [55].

Soit « U » l'ensemble de tous les utilisateurs et « I » l'ensemble de tous les items. **La figure 2.11** ci-dessous montre comment les données implicites sont traitées dans le cas de recommandations d'items.



Figure 2.11 : Les interactions existantes entre l'utilisateur et l'item

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 [55], 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 :

$$(u, i, j) \in D_s \tag{2.1}$$

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

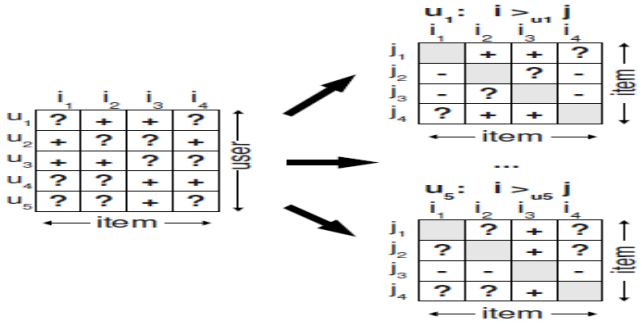


Figure 2.12 : Les préférences par paire spécifiques à l'utilisateur entre une paire d'items

Dans la **figure 2.12** ci-dessus, l'utilisateur *u1* a vu l'item *i2* mais pas l'item *i1*, de sorte que l'algorithme suppose que cet utilisateur préfère l'item *i2* à *i1* (*i* 2> *i1*) 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 *i1* et *i4* pour l'utilisateur *u1*). Au contraire, vous pouvez observer un signe négatif pour (*i1*, *j2*) car l'utilisateur préfère item 2 (*i2*) à item 1 (*i1*).

Les auteurs [31] 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*) [55].

$$\sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda\|\Theta\|^2 \qquad (2.2)$$

Ou σ est la Fonction logistique sigmoïde :

$$\sigma(x) := \frac{1}{1+e^{-x}} \qquad (2.3)$$

$\hat{x_{uij}}$ (Θ) 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). λ, Θ 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 :

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta)) \qquad (2.4)$$

Ici, $>_u$ est la structure de préférence souhaitée mais latente pour l'utilisateur u. Il est également important de noter que $p(i >_u j|\ \theta)$ 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

$$P(\Theta|>u)\ \alpha\ p\ (>u\ |\Theta)\ p\ (\Theta) \qquad (2.5)$$

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

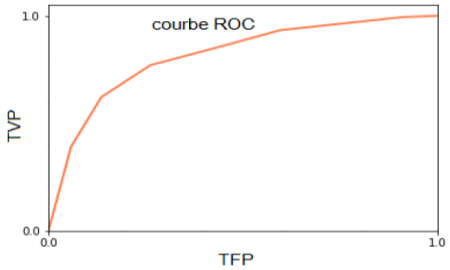


Figure 2.13 : La courbe ROC

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 :

$$TVP = \frac{VP}{VP+FN} \tag{2.6}$$

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

$$TFP = \frac{FP}{FP+VN} \tag{2.7}$$

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

L'équation ci-dessous 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 :

$$AUC = \frac{1}{|U|}\sum_{u\in U}\frac{1}{|I_u^+||I/I_u^+|}\sum_{i\in I_u^+}+\sum_{j\in|I/I_u^+|}\delta(\hat{x}_{uij} > 0) \tag{2.8}$$

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.

2.5.4 Le filtrage collaboratif neuronal (NCF*)*

La factorisation matricielle en anglais factorisation 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 :

- 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).
- 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.

2.5.5 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 [56].

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'auto-encodeur 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 (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 Auto-encoder 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.

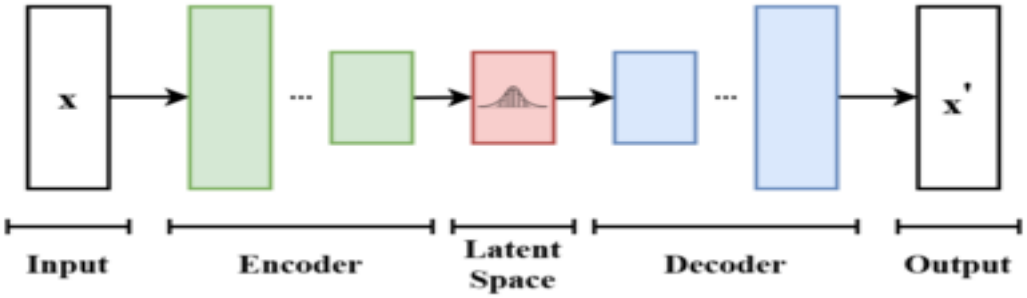


Figure 2.14 : Le schéma de base d'un auto-encodeur variationnel

e. AutoRec

AutoRec est 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.

2.7. Conclusion

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

La 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.

Chapitre 3
Approche proposé

3.1. Introduction

Le mécanisme de modélisation permet la représentation d'un système informatique afin d'obtenir une solution convenable. Son principe est l'abstraction de la réalité pour bien comprendre le système à réaliser, donc connaitre mieux le concept de travail, ses démarches et les méthodes utilisées.

Dans ce chapitre Nous présentons l'architecture de notre modèle et nous détaillons chaque partie avec son rôle. Nous allons faire la préparation (prétraitement) des données, et à l'aide de Word2Vec nous valorisons ces données. L'étape suivante c'est l'application d'un Auto-encodeur pour extraire la sémantique entre les données puis utiliser un MLP afin de prédire des évaluations. La dernière étape consiste à faire la recommandation avec la nouvelle approche proposée qui est différente par rapport aux approches classiques, afin de recommander une liste des ressources avec des scores aux utilisateurs.

3.2. Architecture de l'approche proposée

Le modèle de recommandation que nous proposons est basé sur deux étapes principales, chacune est composée de sous-étapes, Nous commençons tout d'abord par illustrer notre architecture du modèle que nous avons créé, nous allons expliquer tous ses étapes dans les prochaines lignes, nous avons quatre sous-étapes à suivre : prétraitement, traitement, prédiction et recommandation comme dans la figure 3.1 suivant :

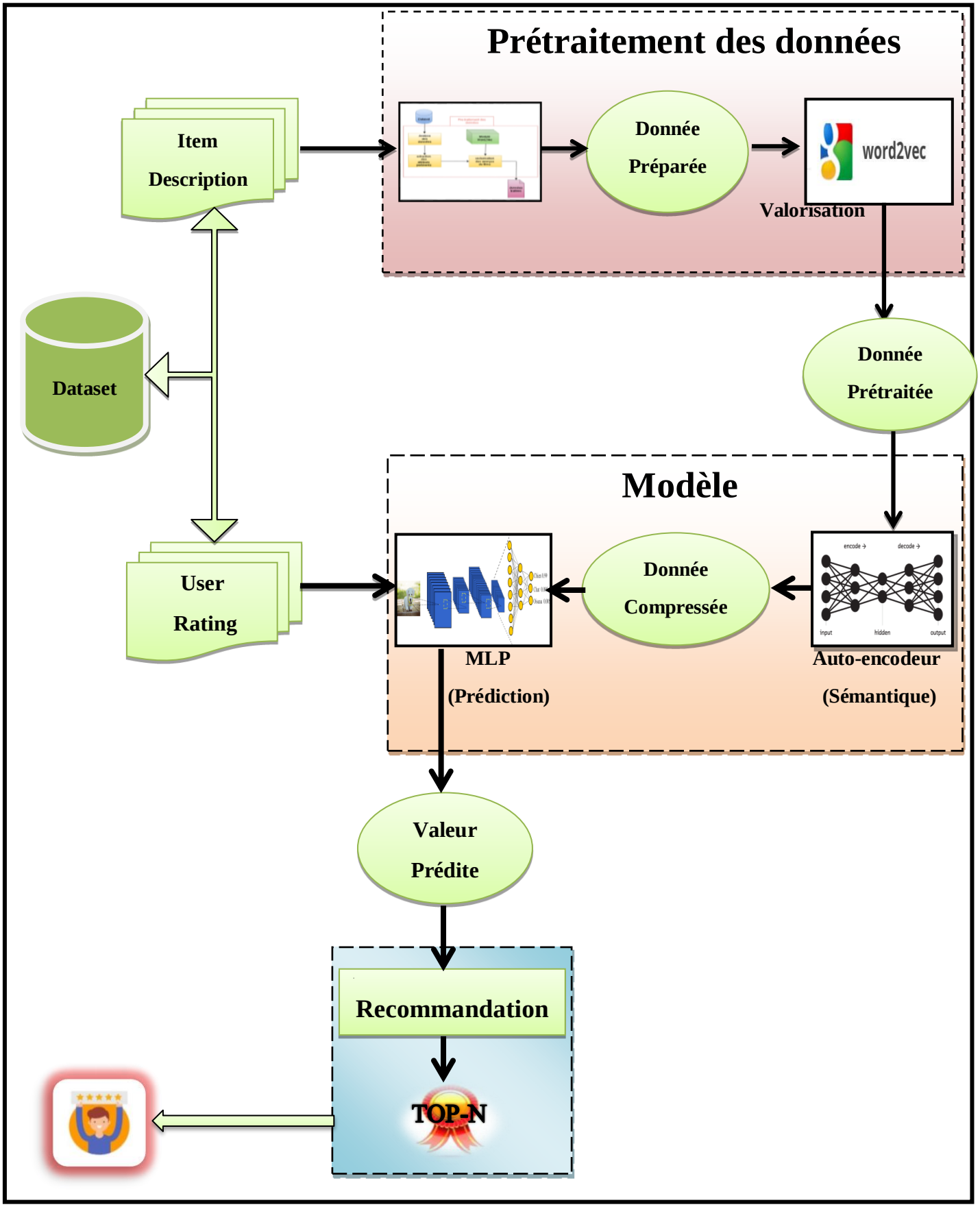


Figure 3.1 : Architecture du système de recommandation proposé

3.3. Description de l'approche

Nous dressons un tableau qui présente toutes les formes utilisées dans notre architecture

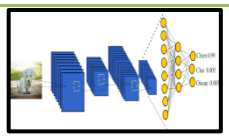
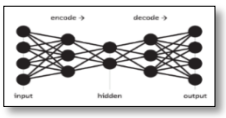
Schéma	Description
	Traitement et prétraitement des données
	Data set
	Fichiers de données
	Données traitées, préparées, prédites
	Modèle MLP
	Auto-encodeur
	Top N
	Word2Vec
	Utilisateur
	Prétraitement

Tableau 3.1 Une description des formes utilisées dans l'architecture.

3.4.1. Etape 1 : Prétraitement

L'objectif de cette étape est de comprendre, de nettoyer et de préparer les données, cela inclut la suppression de tout le bruit inutile ainsi que la correction des incohérences indésirables dans l'ensemble de données.

Tout d'abord, nous avons supprimé tous les attributs dont nous n'avions pas besoin et n'avons conservé que ceux qui étaient essentiels à notre modèle et corriger les éventuelles incohérences ou erreurs dans l'ensemble de données.

Ensuite, nous avons dû vectoriser nos descriptions d'articles à l'aide du module Word2Vec, puisque celles-ci seront utilisées dans notre prochaine étape, chaque description d'article sera transformée en un vecteur de grande dimension.

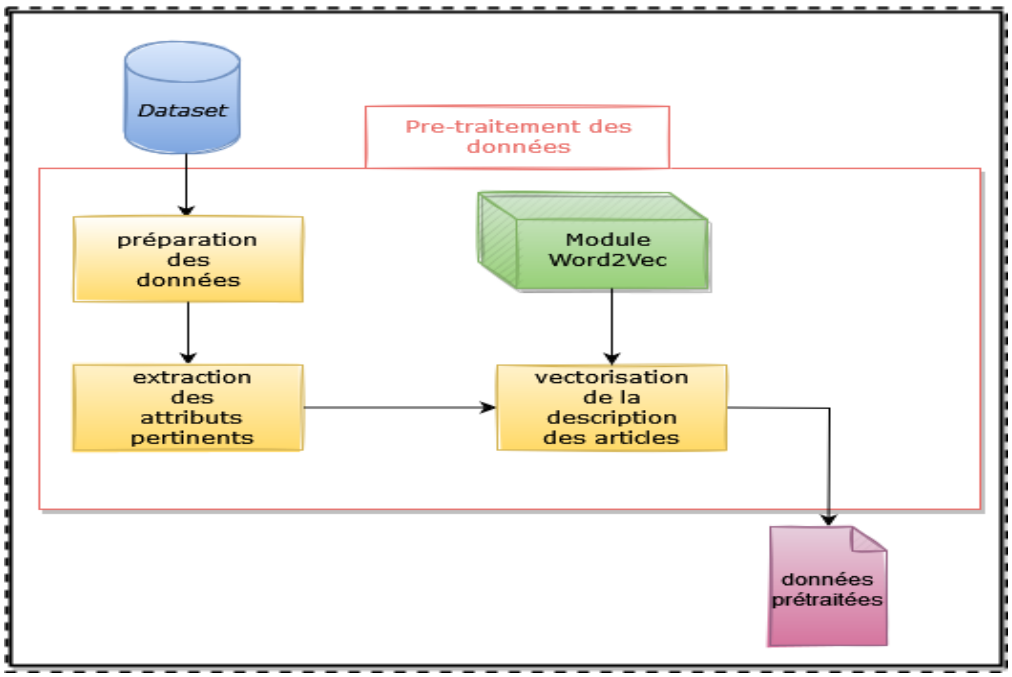


Figure 3.2 : Architecture de Prétraitement

3.4.2. Etape 2 : Traitement

La deuxième étape consiste à réduire la dimensionnalité de la dernière vue d'ensemble du film vectorisé. La réduction de la dimensionnalité est définie comme le processus de réduction du nombre de caractéristiques qui ont certaines données, cette réduction est effectuée en extrayant un nombre inférieur de nouvelles caractéristiques qui sont créées sur la base des anciennes caractéristiques [79].

Dans notre cas, cela se fait à l'aide de l'auto-encodeur variationnel composé de deux parties, un encodeur qui produit ces nouvelles représentations avec moins de caractéristiques à partir de la représentation des "anciennes caractéristiques", et un décodeur qui effectue le processus inverse [79]. Une fois les données d'entrée encodées, elles sont présentes sous une nouvelle forme qui préserve les éventuelles similarités entre les données.

Dans cette étape, nous utiliserons deux modèles, le modèle 1 avec un auto-encodeur normal (AEN) et le modèle 2 avec un auto-encodeur variationnel (AEV). Au départ, nous utilisions la fonction d'activation ReLu pour nos auto-encodeurs, mais cela nous a amenés à rencontrer le problème "The Dying ReLu", où la plupart de nos neurones sortaient 0 pour n'importe quelle entrée, cette situation étant irrécupérable. Afin d'éviter ce problème, nous avons utilisé la fonction d'activation « LeakyReLu » à la place. Appelée "Leaky" car elle échappe les petits nombres négatifs au lieu de zéro pour les entrées négatives. [80, 81].

$$\mathbf{LeakyRelu} \;=\; \boldsymbol{f(x) = max(0.1x\,, x)} \qquad (4.1)$$

Où : x est la valeur d'entrée et la sortie, c'est le maximum entre 0.1x et la valeur d'entrée.

$$\mathbf{f(x)} = \begin{cases} \mathbf{0.1x, si\ x < 0} \\ \mathbf{x, si\ x \geq 0} \end{cases} \qquad (4.2)$$

EN UTILISANT LA COUCHE DE GOULOT D'ETRANGLEMENT DE NOTRE AUTO-ENCODEUR, NOUS EXTRAYONS LA REPRESENTATION COMPRESSEE POUR NOS VECTEURS DE GRANDE DIMENSION QUI SERONT ENSUITE UTILISES DANS LA PROCHAINE ETAPE.

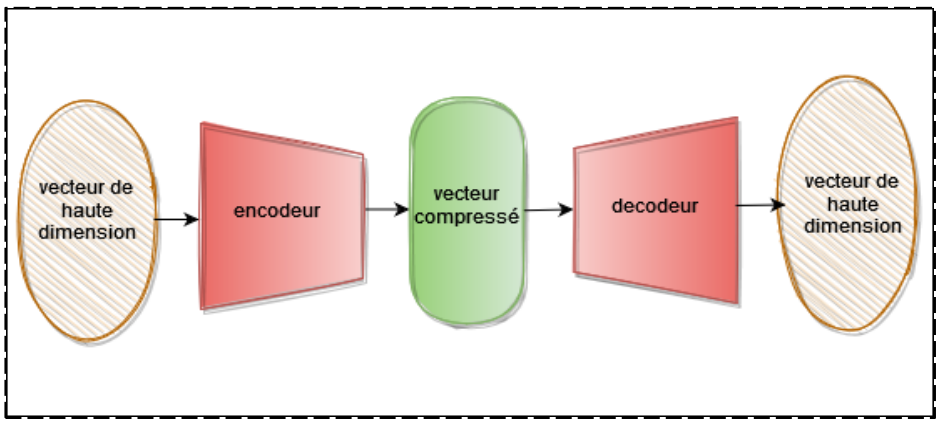


Figure 3.3 Architecture de l'étape 2 : Traitement

3.4.4. Etape 3 : Modèle

Pour notre prochaine étape, nous allons utiliser un autre type de réseau de neurones appelé MLP (Multi Layer Perceptron), notre MLP prendra en entrée un identifiant d'utilisateur et un identifiant de film et produira une note que cet utilisateur donnerait au film, les identifiants d'utilisateur et l'identifiant de film doivent être transformés en intégrations à l'aide d'une couche d'intégration.

L'intégration est définie comme le processus de représentation de variables discrètes sous forme de vecteurs continus, pour les identifiants de film, nous utiliserons les vecteurs compressés comme leurs intégrations, mais pour les identifiants d'utilisateur, nous devons créer de nouvelles intégrations [82].

Ces couches d'intégration déconnectées sont ensuite connectées les unes aux autres après avoir traversé chacune une couche dense, puis cette sortie concaténée traverse 5 couches denses avant de produire une évaluation.

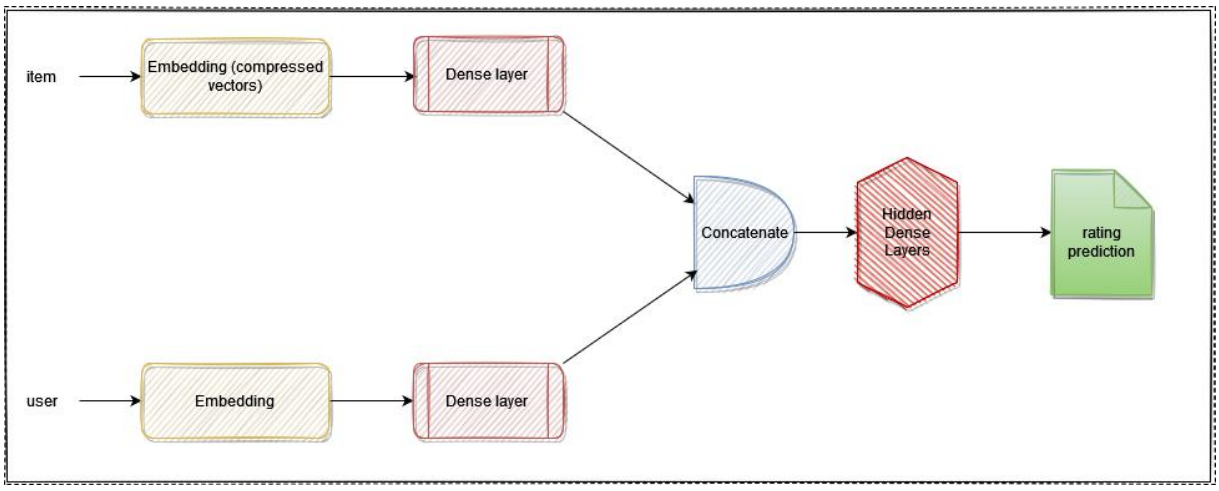


Figure 3.4 : Architecture de l'étape 3 : Modèle

Pour entraîner et tester nos modèles, nous utilisons la méthode de validation croisée K-fold où k est fixé à 5.

3.4.4. Etape 4 : Recommandation

Dans la dernière étape de notre projet, nous recommanderons les éléments qui pourraient intéresser ou satisfaire un utilisateur en utilisant les prédictions de notre modèle. Pour chaque utilisateur, il y aura une liste de tous les éléments triés par note en l'ordre décroissant. A partir de la liste, nous choisirons les N films les mieux notés et les recommanderons à l'utilisateur.

3.5. Conclusion

Nous avons présenté dans ce chapitre notre méthodologie de travail et détaillé tous ses étapes, nous avons créé un nouveau modèle de système de recommandation sans utilisé une des techniques classiques, et à l'aide de l'apprentissage profond avec l'algorithme de réseau de neurones et nous allons voir les résultats de l'expérimentation dans le chapitre suivant.

Chapitre 4
Expérimentations

4.1. Introduction

L'objectif de ce dernier chapitre est de présenter l'expérimentation et l'évaluation de nos modèles. Les expérimentations ont été réalisées sur une collection de tests basée sur le jeu de données "MovieLens", plusieurs métriques d'évaluation ont été utilisées pour évaluer nos deux modèles et les comparer à quelques algorithmes sélectionnés.

Ce chapitre est structuré comme suit, en premier lieu nous présentons le jeu de données, ensuite les outils et langages utilisés pour implémenter les modèles, puis nous présentons les différentes métriques d'évaluation utilisées pour évaluer les algorithmes, une description succincte des algorithmes est présentée. A la fin de ce chapitre nous présentons les résultats obtenus pour les différents algorithmes et nous les discutons.

4.2. Collection de données

Dans notre travail, nous avons utilisé la collection Movies Dataset fournie par le contributeur kaggle Rounak Banik. L'ensemble de données se compose de films sortis en juillet 2017 ou avant et contient des métadonnées pour environ 45 000 films répertoriés dans l'ensemble de données MovieLens. Il contient également environ 26 millions de notes de 270 000 utilisateurs pour les 45 000 films. Les notes sont sur une échelle de 0,5 à 5 et ont été recueillies sur le site officiel "GroupLens". Chaque film est caractérisé par près de deux douzaines d'attributs dont sa description, sa date de sortie et son budget [86].

4.2.1. Jeu de données

Ce jeu de données se compose des fichiers suivants :

- movies_metadata.csv : le fichier principal de métadonnées de films. Contient des informations sur 45 000 films présentés dans l'ensemble de données Full MovieLens. Les fonctionnalités incluent les affiches, les arrière-plans, le budget, les revenus, les dates de sortie, les langues, les pays de production et les entreprises.
- keywords.csv : contient les mots-clés de l'intrigue du film pour nos films MovieLens.
- credits.csv : contient des informations sur les acteurs et l'équipe de tous nos films.
- links.csv : Le fichier qui contient les identifiants TMDB et IMDB de tous les films présentés dans le jeu de données Full MovieLens.
- links_small.csv : contient les identifiants TMDB et IMDB d'un petit sous-ensemble de 9 000 films de l'ensemble de données complet.
- ratings_small.csv : le sous-ensemble de 100 000 évaluations de 700 utilisateurs sur 9 000 films. L'ensemble de données complet MovieLens composé de 26 millions d'évaluations et de 750 000 applications de balises de 270 000 utilisateurs sur les 45000 films.

Nous n'avions besoin que de l'identifiant du film, du titre original et de l'aperçu qui est une brève description du film.

	id	original_title	overview
0	2	Ariel	Taisto Kasurinen is a Finnish coal miner whose...
1	3	Varjoja paratiisissa	An episode in the life of Nikander, a garbage ...
2	5	Four Rooms	It's Ted the Bellhop's first night on the job....
3	6	Judgment Night	While racing to a boxing match, Frank, Mike, J...
4	11	Star Wars	Princess Leia is captured and held hostage by ...
5	12	Finding Nemo	Nemo, an adventurous young clownfish, is unexp...
6	13	Forrest Gump	A man with a low IQ has accomplished great thi...
7	14	American Beauty	Lester Burnham, a depressed suburban father in...
8	15	Citizen Kane	Newspaper magnate, Charles Foster Kane is take...
9	16	Dancer in the Dark	Selma, a Czech immigrant on the verge of blind...

Figure 4.2 : Un extrait de l'ensemble de données de films

Chaque utilisateur n'était caractérisé que par son identifiant. Et chaque note consistait en un identifiant d'utilisateur, un identifiant de film et des notes attribuées à ce film par l'utilisateur.

	userId	id	rating
0	1084	10389	3.5
1	4302	8689	4.0
2	366	2054	4.0
3	1918	9687	3.0
4	1149	568	4.0
5	4325	3052	1.0
6	4916	10533	3.5
7	3748	14	2.0
8	4651	7326	3.5
9	3785	996	4.0

Figure 4.3 : Un extrait du fichier texte des notes

Notre phase NLP prend les aperçus de films en entrée, nous avons donc dû supprimer les films qui n'avaient pas d'aperçu (chaîne vide). Nous avons également supprimé les films dans lesquels leur aperçu contient des lettres non latines, et les films qui avaient moins de 20 notes.

La suppression d'un film implique la suppression de son entrée de l'ensemble de données de films ainsi que la suppression de toutes ses notes associées du fichier de notes.

Nous avons sélectionné les 1 000 utilisateurs qui ont évalué le plus grand nombre de films. Notre ensemble de données final se compose de 1 000 utilisateurs et de 6 248 films, chaque note est comprise entre 0,5 et 5, le nombre total de notes faites par ces utilisateurs est de 881 179 notes.

Notre premier but est de transformer chaque description de film en vecteur, pour cela nous utilisons Word2Vec qui prend en entrée toutes nos descriptions de films et attribue à chaque mot un vecteur d'une taille fixe choisie par nous, chaque mot unique n'a qu'un vecteur unique. Nous avons choisi notre taille de vecteur à 100, ce qui signifie que chaque mot est représenté en utilisant une taille de vecteur de 100 valeurs.

```
print(model.wv["fun"])
```

```
[-0.03055757  0.5305596   0.12383054  0.07252285  0.00367015 -0.3053976
  0.2450483   0.73783106 -0.37329483 -0.20041195 -0.00082908 -0.21379119
 -0.24246253  0.07515674  0.14043221 -0.39709148  0.3162501  -0.07101355
 -0.22375375 -0.47613963  0.04646353  0.10651503  0.2428732  -0.13165712
 -0.09692822 -0.14548685 -0.12902066 -0.03944501 -0.24575602 -0.03842051
  0.2817965  -0.3826401   0.15233319 -0.43262717  0.02651627  0.25986484
  0.20376392 -0.18704636 -0.00395096 -0.58131516  0.06809156 -0.01083586
 -0.36789176  0.08659188  0.24545608 -0.01799328 -0.32636338 -0.02026772
  0.07358503  0.10224366  0.1025242  -0.21617529 -0.17796369 -0.17237987
  0.08551988  0.00942971  0.13565686 -0.26226103 -0.27840585  0.11644179
  0.03733495 -0.09409679  0.05163316  0.15091573 -0.23645386  0.545616
  0.07809174  0.22773932 -0.48005137  0.297715   -0.13415998  0.21972571
  0.47049147  0.01015609  0.17461322  0.05999811  0.02258878  0.16310549
 -0.22067477 -0.06768791 -0.4067611  -0.07651101 -0.12914534  0.40970597
 -0.08083916 -0.21333551  0.4770046   0.01009933  0.18509781  0.05981459
  0.2922249   0.04711837  0.22763072  0.00923389  0.73480624  0.33756462
  0.05394157 -0.2716295   0.05421627  0.09255812]
```

Figure 4.4 : Représentation modèle W2V du mot "fun"

Les mots similaires auront des vecteurs plus proches les uns des autres dans l'espace vectoriel. En utilisant ces vecteurs (incorporations de mots), nous transformons nos aperçus de films en grands vecteurs en concaténant simplement les incorporations de mots correspondants des mots qui sont présents dans l'aperçu du film.

Étant donné que chaque aperçu de film a un nombre variable de mots, nos vecteurs résultants seront de taille variable, donc avant d'alimenter nos vecteurs dans un réseau de neurones, nous devons trouver la taille maximale qu'un vecteur peut avoir et remplir tous les autres vecteurs avec des zéros afin que leur taille est égale à cette taille maximale.

```
Star Wars
[-0.00999757  0.01136627  0.01354597 ...  0.          0.
  0.        ]
length of our movie vector is :21000


The Lord of the Rings: The Return of the King
[-1.42488718  1.39290059 -0.13615039 ...  0.          0.
  0.        ]
length of our movie vector is :21000
```

Figure 4.5 : Exemple de deux films transformés en vecteurs de taille fixe

Le nombre maximum de mots qu'un aperçu de film dans notre jeu de données de film peut avoir est de 210, et étant donné que notre intégration de mots est de taille 100, cela se traduit par un vecteur de taille 21000 pour chaque aperçu de film.

4.3. Outils et Langages

Dans cette partie, nous allons spécifier les outils utilisés pour expérimenter notre approche.

4.3.1 Python

Python est un langage de programmation interprété, orienté objet et de haut niveau avec une sémantique dynamique. La syntaxe Python simple et facile à apprendre met l'accent sur la lisibilité et réduit donc le coût de maintenance du programme, c'est pourquoi nous avons choisi python pour l'implémentation de nos modèles [83].

4.3.2 Plateformes de développement

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

a. 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 [57].

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++.

b. 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 [58].

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

c. 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 CUDA1. Il est issu des équipes de recherche de Facebook [59].

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 Numpy2 et d'effectuer des calculs efficaces sur CPU ou GPU (par exemple, des produites de matrices ou des convolutions) ;
- ✓ calculer des gradients pour appliquer facilement des algorithmes d'optimisation par descente de gradient. PyTorch utilise la bibliothèque autograd.

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 2.4 Les fonctionnalités des plateformes

4.3.3 Kaggle kernels

Kaggle Kernels est un environnement de calcul cloud qui permet une analyse reproductible et collaborative. Les noyaux prennent en charge les scripts en R et Python [84].

Les spécifications du noyau Kaggle que nous avons utilisé sont les suivantes [85] :

Mémoire (RAM) : 16 GB.
Processeur : Intel(R) Xeon(R) CPU @ 2.30GHz

Carte Graphique : Tesla P100 16 GB

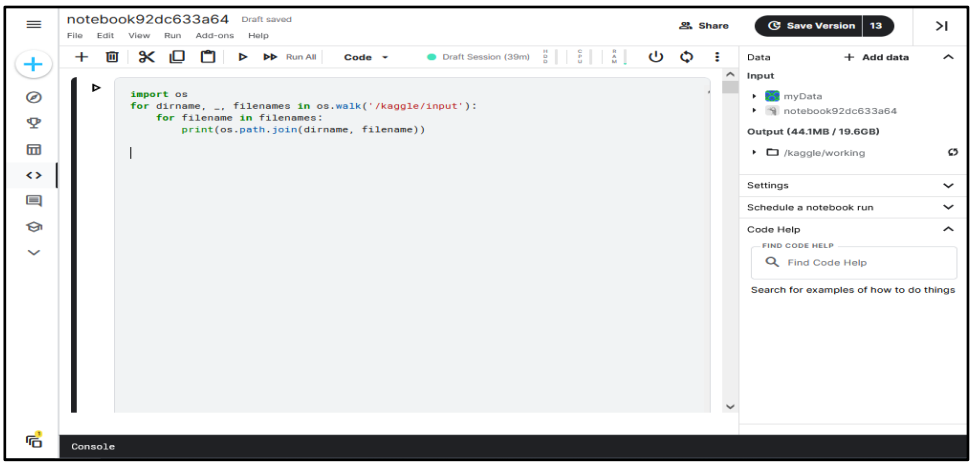


Figure 4.1 : L'environnement Kaggle

4.4. Mesures d'évaluation utilisées

Afin d'évaluer notre modèle, nous avons proposé l'utilisation des métriques d'évaluation suivantes :

4.4.1 Erreur quadratique moyenne (RMSE)

L'erreur quadratique moyenne est la racine carrée de la moyenne du carré de toutes les erreurs. L'utilisation de RMSE est très courante et est considérée comme une excellente mesure d'erreur à usage général pour les prédictions numériques. L'erreur quadratique moyenne entre les notes réelles et les notes prédites est donnée par [87] :

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \quad (4,1)$$

- y_i: est la note réelle.
- $\hat{y}_i$: est la note prévue.
- n : est le nombre de notes

4.4.2 Erreur absolue moyenne (MAE)

L'erreur absolue fait référence à l'ampleur de la différence entre la prédiction d'une observation et la valeur réelle de cette observation. L'erreur absolue moyenne prend la moyenne des erreurs absolues pour un groupe de prédictions et d'observations comme mesure de l'ampleur des erreurs pour l'ensemble du groupe. [88]

$$\text{MAE} = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \quad (4,2)$$

- y_i: est la note réelle.
- $\hat{y}_i$: est la note prévue.
- n : est le nombre de notes

4.4.3 Précision TOP-N

La précision Top-N prend les prédictions du modèle N avec une probabilité plus élevée, et si l'une d'entre elles est supérieure à un seuil prédéfini, la prédiction est étiquetée comme correcte ou pertinente. [88]

Pour chaque utilisateur, nous prenons les évaluations TOP-N prédites et voyons combien d'entre elles sont pertinentes.

$$TOP - N(u) = \frac{P_u}{N} \quad (4.3)$$

P_u : nombre de prédictions pertinentes dans les TOP-N prédictions pour l'utilisateur u

Et puis nous calculons la précision TOP-N moyenne pour tous les utilisateurs :

$$avg_{acc} = \frac{\sum_{i=1}^{n} TOP-N(u_i)}{U} \quad (4,4)$$

U : nombre total d'utilisateurs.

4.5. Processus d'évaluation

4.5.1 Matrice de factorisation

Avant le prétraitement de nos jeux de données, la densité de notre matrice de factorisation était extrêmement faible :

$$Densité(R_0) = \frac{26{,}024{,}289}{270{,}896 * 45{,}115} = 0.0021 = 0.21\%$$

Après avoir prétraité nos ensembles de données et sélectionné les films et utilisateurs pertinents nécessaires à l'évaluation de nos modèles, notre densité de matrice de factorisation est devenue :

$$Densité(R) = \frac{881{,}179}{1{,}000 * 6{,}248} = 0.141 = 14.1\%$$

Le prétraitement a augmenté la densité de notre matrice de factorisation de 13,89 %, alors que notre matrice de factorisation est encore un peu clairsemée, elle a une densité beaucoup plus élevée que celle d'origine, ce qui améliorera considérablement les performances de notre modèle.

4.5.2 Auto-encodeurs

Les nouveaux vecteurs représentant la vue d'ensemble de films sont assez grands (chaque vecteur contient 21 000 caractéristiques), notre prochaine étape consistera à appliquer une réduction de dimensionnalité à ces vecteurs à l'aide d'un auto-encodeur.

Nos deux encodeurs automatiques contiennent 10 couches, l'encodeur automatique variationnel contient 10 couches convolutionelles tandis que l'encodeur normal contient 10 couches denses, ils prennent chacun une entrée de 21000 caractéristiques et compressent le nombre de caractéristiques à 25 à l'aide de l'encodeur, le décodeur utilisera alors ce vecteur de caractéristiques compressé pour reconstruire l'entrée d'origine de l'encodeur.

Nous donnons le pseudo code de l'algorithme de l'auto-encodeur :

```
Entrée : movie_vectors – Matrice [NB_MOVIES, SIZE]
Sortie : compressed_movie_vectors  –  Matrice [NB_MOVIES, 25]
Début
    model = train (input = movie_vectors, output = movie_vectors )
   encoder = model.get_layer("encoder" )
   pour i de 1 a length(movie_vectors)
        compressed_movie_vectors[i] = encoder( movie_vectors[i] )
  Fin pour
retourn compressed_movie_vectors
Fin
```

Afin de comparer les deux auto-encodeurs, nous devions les former pour le même nombre d'époques, initialement nous voulions former chaque auto-encodeur pour 1000 époques, mais il s'est avéré que l'auto-encodeur normal prendrait une semaine pour terminer la formation dans ce cas, et depuis Kaggle ne permet que 30 heures d'utilisation de la carte graphique par semaine et comme nous avons remarqué cette grande amélioration concernant RMSE et MAE pour notre auto-encodeur variationnel à 1000 époques, nous avons fini par former nos deux auto-encodeurs pour 20 époques seulement.

Le tableau suivant présente la moyenne RMSE et MAE pour les deux auto-encodeurs, ainsi que le nombre d'époques et le temps nécessaire pour terminer une époque pour chacun d'eux :

Type d'AE / Données	AEV	AEN
RMSE	**0.1798**	**0.1727**
MAE	**0.0762**	**0.075**
Nombre d'époques	**20**	**20**
Temps par époque	**~4 seconds**	**~9 minutes**

Tableau 4.1 : Différentes données relatives à l'apprentissage des auto-encodeurs

Nous remarquons que les deux Auto-encodeurs ont des valeurs RMSE et MAE similaires et acceptables, mais l'Auto-encodeur normal (AEN) était massivement plus lent que l'Auto-encodeur variationnel (AEV).

Nous utilisons ensuite les deux encodeurs de nos encodeurs automatiques pour transformer chaque grand vecteur en un vecteur compressé de dimension 25, notre modèle sera bifurqué en deux modèles, l'un qui utilise les vecteurs compressés de l'encodeur automatique variationnel (AEV) et l'autre modèle qui utilise les vecteurs compressés de l'encodeur automatique normal (AEN).

4.5.3. MLP

Pour notre prochaine étape, nous allons utiliser un autre type de réseau de neurones appelé MLP (Multi Layer Perceptron), notre perceptron prendra en entrée un identifiant d'utilisateur et un identifiant de film et produira une note que cet utilisateur donnerait au film.

Les identifiants d'utilisateur et l'identifiant de film doivent être transformés en intégrations à l'aide d'une couche d'intégration, l'intégration est définie comme le processus de représentation de variables discrètes sous forme de vecteurs continus, pour les identifiants de film, nous utiliserons les vecteurs compressés comme leurs intégrations, mais pour les identifiants d'utilisateur, nous allons doivent créer de nouvelles intégrations qui sont générées automatiquement par la couche d'intégration de notre MLP [91] .

Nous donnons le pseudo code de l'algorithme de MLP :

Entrées: compressed_movie_vectors – Matrice de [NB_MOVIES, 25]

Ratings – Matrice de [NB_MOVIES, NB_USERS]

Actual_ratings – Matrice de [NB_MOVIES, NB_USERS]

Sorties : matrice_factorization – Matrice de [NB_MOVIES, NB_USERS]

Début

model=train(input=[compressed_movie_vectors,user_id_embeddings],output=ratings)

MAE=test(model)

k=i=0

Tantque (I <NB_USERS)

j=0

Tantque (j < NB_MOVIES)

predictions[k] = predict(model,i,j)

```
        k=k+1
        j=j+1
      Fin Tantque
      i=i+1
    Fin Tantque
    Predictions = max(round(predictions), actual_ratings)
    matrice_factorization = transform(predictions)
    retourn matrice_factorization
    Fin
```

Pour former et tester nos deux modèles, nous avons utilisé la validation croisée, qui implique les étapes suivantes : [92]

1- Mélangez notre jeu de données au hasard
2- Diviser notre ensemble de données de mélange en un ensemble d'entraînement et de test.
3- Entraînez notre modèle à l'aide de l'ensemble de données d'entraînement
4- Valider notre modèle à l'aide de l'ensemble de test
5- Répétez les étapes 1 à 4 quatre fois de plus

Une fois notre modèle formé, nous testons notre modèle à l'aide de l'ensemble de test et nous obtenons les valeurs suivantes :

Métrique	Model 1 (AEV)	Model 2 (AEN)
RMSE	**0.9453**	**0.8176**
MAE	**0.7393**	**0.6281**

Tableau 4.2 : Métriques d'erreur de MLP pour les deux modèles

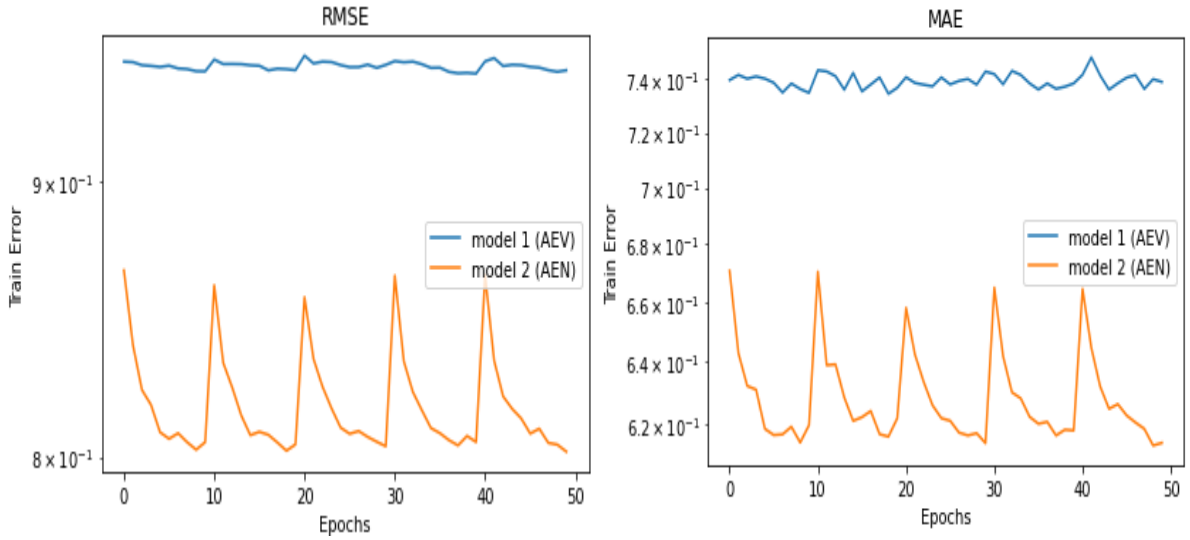


Figure 4.6 : RMSE et MAE changent d'époque pendant la formation MLP

Une fois la formation et les tests terminés, nous commençons à utiliser notre MLP pour prédire les notes, les prédictions sont arrondies à l'entier le plus proche ou le plus proche (entier + 0,5), par exemple, si notre modèle fait une prédiction de 4,29, elle est arrondie à 4,5.

Voici un exemple de résultat des 10 premières notes de notre fichier de notes mélangées, puisque nous avons 6248 films et 1000 utilisateurs dans notre ensemble de données, le nombre de classements possibles est 6, 248,000, cela signifie que notre table de prédictions contiendra 6, 248,000 prédictions, qui sont ensuite transformées en une matrice de dimensions (1000, 6248).

En utilisant notre matrice de factorisation nouvellement remplie, nous calculons le TOP-N pour obtenir la précision moyenne (Avg_acc). Pour N=100 et des notes égales ou supérieures à 4.5 considérées comme pertinentes, la précision moyenne est :

$$avg_{acc}\ (model\ 1) = 80.2\% \qquad avg_{acc}(model\ 2) = 91.3\%$$

Nous remarquons que de manière inattendue, le modèle qui utilise l'auto-encodeur normal (modèle 2) a mieux performé que le modèle qui utilise l'auto-encodeur variationnel plus compliqué (model 1).

4.6. Comparaison et analyse

Nous comparons nos modèles à quelques autres algorithmes populaires sur exactement le même ensemble de données :

Algorithm Métrique	SVD	NMF	K-NN	Co-clustering	Model 1 (AEV)	Model 2 (AEN)
RMSE	0.7782	0.9091	0.8198	0.8316	0.9453	0.8176
MAE	0.5955	0.6946	0.6268	0.6489	0.7393	0.6281
Avg_acc	80.5%	94.15%	80.06%	83.67%	80.21%	91.31%

Tableau 4.3 : Comparaison entre nos deux modèles et différents autres algorithmes

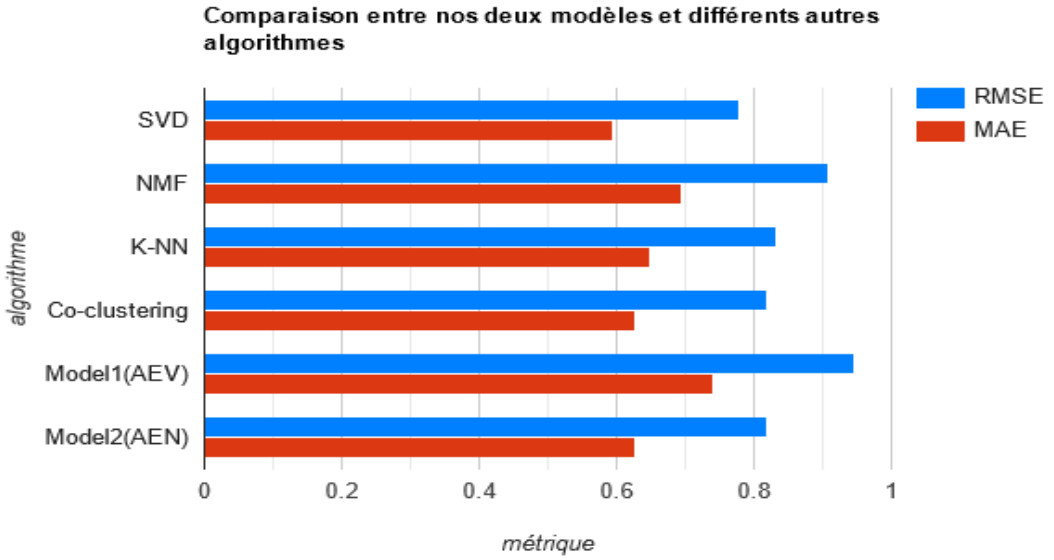


Figure 4.7 : Comparaison entre nos deux modèles et différents autres algorithmes

4.6.1. Algorithmes choisis pour la comparaison

- **SVD :** une technique de factorisation matricielle qui décompose notre matrice de factorisation en 3 matrices génériques et familières, qui sont ensuite utilisées pour faire nos prédictions [93].
- **NMF :** un algorithme qui décompose notre matrice de factorisation, ne contenant que des coefficients non négatifs, en le produit de deux autres matrices non négatives avec des rangs réduits qui sont ensuite utilisées pour faire nos prédictions [94].
- **K-NN :** est une méthode d'apprentissage paresseux non paramétrique. Il utilise une base de données dans laquelle les points de données sont séparés en plusieurs groupes pour faire des inférences pour de nouveaux échantillons [95].

- **Co-clustering :** est une technique de datamining qui permet de regrouper simultanément les lignes et les colonnes d'une matrice, Étant donné un ensemble d'échantillons représentés par un vecteur de caractéristiques dimensionnel, l'ensemble de données entier peut être représenté sous forme de lignes dans des colonnes (c'est-à-dire une matrice). L'algorithme de co-clustering génère des co-clusters - un sous-ensemble de lignes qui présentent un comportement similaire dans un sous-ensemble de colonnes, ou vice versa [96].

4.6.2. Analyse des résultats

Notre premier modèle (AEV) a obtenu de moins bons résultats que tous les algorithmes présentés, mais notre deuxième modèle (AEN) avait des performances exceptionnelles et était à égalité avec la plupart des algorithmes, et seul SVD a réussi à le surpasser sur cet ensemble de données.

4.7. Conclusion

Dans ce chapitre, nous avons présenté l'expérimentation et l'évaluation de notre modèle, également fourni les métriques utilisées ainsi que leurs résultats correspondants et les résultats de comparaison entre nos modèles et d'autres algorithmes/modèles. Nous avons conclu que le modèle a montré des résultats prometteurs et il y avait encore beaucoup de place à l'amélioration.

Conclusion générale

Nous avons essayé de créer un tout nouveau modèle et de l'implémenter sur un ensemble de données favorable, un ensemble de données dans lequel la factorisation n'est pas aussi clairsemée qu'elle l'est habituellement.

l'apprentissage en profondeur a été la base sur laquelle nous avons construit notre modèle, nous l'avons utilisé dans notre préparation de données (Word2Vec), dans notre extraction de relations sémantiques (auto-encodeurs), et dans notre processus de prédiction (MLP), les résultats que nous avons à double tranchant, d'un côté, notre modèle MLP dépendant de l'encodeur automatique variationnel a fonctionné quelque peu modestement, et de l'autre côté, notre modèle MLP dépendant de l'encodeur automatique normal a montré des résultats vraiment impressionnants, et mieux encore, il y a encore beaucoup de place pour l'amélioration.

Malgré toutes les difficultés et les obstacles rencontrés, nous avons constaté que nous avons beaucoup appris de cette expérience, et que notre travail a fourni beaucoup d'informations sur le sujet de l'apprentissage en profondeur.

Notre perspective à long terme serait d'expérimenter davantage avec le modèle et de peaufiner son fonctionnement interne, et d'appliquer ce modèle à d'autres ensembles de données tels que les ensembles de données de livres, les ensembles de données musicales, ..., tout en augmentant la taille de nos ensembles de données.

REFERENCES BIBLIOGRAPHIQUES

[1] Mémoire : Conception et Réalisation d'un Système de Recommandation à l'aide d'un GAN Sujet Proposé par : Mme BENSAOU Nacera et M. DJOUADA Moussa l'année 2021

[2] N. Vaidya and A. R. Khachane, "Recommender systems-the need of the ecommerce ERA," 2017 International Conference on Computing Methodologies and Communication (ICCMC), 2017, pp. 100-104, doi : 10.1109/ICCMC.2017.8282616.

[3] Adomavicius, G., & 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.

[4] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context awareness. In Handheld and ubiquitous computing, pages 304–307. Springe

[5] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

[6] Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*,23(1):103–145

[7] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6) :734–749

[8] Adomavicius,G. and Tuzhilin,A. (2011). Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer.

[9] Ahn, J.-w., Brusilovsky, P., Grady, J., He, D., and Syn, S. Y. (2007). Open user profiles for adaptive news systems: help or harm? In *Proceedings of the 16th international conference on World Wide Web*, pages 11–20. ACM.

[10] Aimé, X. (2011). Gradients de prototypicalité, mesures de similarité et de proximité sémantique : une contribution à l'Ingénierie des Ontologies. PhD thesis, Université de Nantes.

[11] Amer-Yahia, S., Bonchi, F., Castillo, C., Feuerstein, E. Méndez-Díaz I., and Zabala, P. (2013). Complexity and algorithms for composite retrieval. In Proceedings of the 22nd International Conference on World Wide Web, pages 79–80. ACM.

[12] 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.

[13] Majda MAATALLA, Thèse, Une Technique Hybride pour les Systèmes de Recommandation, Université Badji Mokhtar–Annaba, 2015/2016

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua,Neural Collaborative Filtering, arXiv :1708.05031v2

[15] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets : History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015),19 pages. DOI=http://dx.doi.org/10.1145/2827872

[16] Xue Geng, Hanwang Zhang, Jingwen Bian, Tat-Seng Chua, School of Computing, National University of Singapore, Learning Image and User Features for Recommendation in Social Networks

[17] 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.

[18] Système de Recommandation de Cours à Base d'Ontologie Présentée par Ait Ahmed Nora Idris Khodja Asma 2018

[19] Utilisation de l'approche pré-filtrage contextuel des systèmes de recommandation sensible au contexte Réalisé par : - BENOSMAN Imene. - CHARIF Nesrine 2015-2016

[20] Utilisation de l'approche pré-filtrage contextuel des systèmes de recommandation sensible au contexte Réalisé par : - BENOSMAN Imene. - CHARIF Nesrine 2015-2016

[21] Système de recommandation des services web sémantiques Réalisé par : Mme KARAOUZENE Meryem Présenté le 14 Juin 2015

[22] Système de recommandation des services web sémantiques Réalisé par : Mme KARAOUZENE Meryem Présenté le 14 Juin 2015

[23] Utilisation de l'approche pré-filtrage contextuel des systèmes de recommandation sensible au contexte Réalisé par : - BENOSMAN Imene. - CHARIF Nesrine 2015-2016

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

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

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

[29] Richard S, Yoshua B, Christopher D. Manning, Deep Learning for NLP.

[30] De https://interstices.info/la-revolution-de-lapprentissage-profond/

[31] Josh Patterson & Adam Gibson ,2017. Deep Learning A Practitioner's Approach,1ère (Ed), O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, Mike Loukides & Tim McGovern, 532 p, (pp. 28)

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

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

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

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

[43] Bergstra, J. et Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization.Journal of Machine Learning Research, 13(Feb), 281-305.

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

[47] Christopher W., Gerasimos S, Gerhard W., Flexible Deep Neural Network structure with application to Natural Language Processing. Department of Knowledge Engineering, Maastricht University, The Netherlands.

[48] Fateme Behzadi, Natural Language Processing and Machine Learning: A Review. International Journal of Computer Science and Information Security. (IJCSIS)

[50] Ahammad Fahad, Abdulsamad Ebrahim Yahya, *Inflectional Review of Deep Learning on Natural Language Processing.* International Conference on Smart Computing and Electronic Enterprise. (ICSCEE2018) (IEEE)

[51] Palash G., Sumit P. & Karan J..,2018. *Deep Learning for Natural Language Processing:Creating Neural Networks with Python,* India , Celestin Suresh John, 290p,(pp. 1)

[52] Appariements collaboratifs des offres et demandes d'emploi à Orsay, le 29/06/2018, par Thomas Schmitt

[55] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets." in ICDM, 2008, pp. 263–272

[60] Romain Picot-Clémente. *Une architecture générique de Systèmes de recommandation de combinaison d'items. Application au domaine du tourisme*. PhD thesis, Université de Bourgogne, 2011.

[68] Eve, Mathieu-Dupas. (2010). *Algorithme des k plus proches voisins pondérés et application en diagnostic*. 42èmes Journées de Statistique, Marseille, France

[69] Ben Ticha, Sonia. (2015). Thèse : Recommandation Personnalisée Hybride, Préparée sous la convention de cotutelle UTM-FST (Tunisie) - UL (France).

[70] Esslimani. Ilham. (2010). Thèse : Vers une approche comportementale de recommandation : apport de l'analyse des usages dans un processus de personnalisation. Interface homme-machine [cs.HC]. Université Nancy II, France.

[71] Adomavicius, Gediminas. Et Tuzhilin, Alexander. (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, VOL. 17, NO. 6.

[72] Sarabjot, Singh Anand. Et Bamshad, Mobasher. (2005). Intelligent Techniques for Web Personalization, Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK, Center for Web Intelligence, School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, Illinois, USA.

[73] Herlocker, Jonathan L. Konstan, Joseph A. Loren G. And Riedl, John T. (2004). Evaluating Collaborative Filtering Recommender Systems, Oregon State University, University of Minnesota, USA.

[74] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12) :61–70.

[75] Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. arXiv preprint cmp-lg/9511007.

[76] Shardanand, U. and Maes, P. (1995a). Social information filtering: algorithms for automating? word of mouth? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210– 217. ACM Press/Addison-Wesley Publishing Co.

[77] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co.

[78] Rich, E. (1979). User modeling via stereotypes*. *Cognitive science*, 3(4) :329–354.

[87] *David Christie, Simon P. Neill, in* Comprehensive Renewable Energy, *2022*

[94] Lopes, N., Ribeiro, B. (2015). Non-Negative Matrix Factorization (NMF). In: Machine Learning for Adaptive Many-Core Machines - A Practical Approach. Studies in Big Data, vol 7 Springer, Cham.

[96] *G. Govaert; M. Nadif (2013). Co-clustering: models, algorithms and applications. ISTE, Wiley.*

WEBOGRAPHIE

[25] http://blog.varonis.fr/classification-des-donnees-kesako/

[27] http://fr.wikiversity.org/wiki/R%C3%A9seaux_de_neurones/Allez_plus_loin_le_deep_Learning

[32] https://www.gralon.net/articles/materiel-et-consommables/materiel informatique-et consommable-informatique/article-comprendre-le-deep-learning-en-6-questions-10956.htm

[37] https://www.futura-sciences.com/tech/dossiers/robotique-presentation-historiquereseaux-neuronaux-31/

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

[39] https://halamu.archives/ouvertes.fr/ha1338010/file/Les_reseaux_de_neurones_artificiels.pdf

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

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

[42] https://www.racinely.com/post/les-fonctions-d-activation-part

[44] https://datasciencetoday.net/index.php/en-us/deep-learning/173-les-reseaux-deneurones-convolutifs

[46] http://dataanalyticspost.com/Lexique/auto-encodeur/

[49] http://blogshells.com/how-deep-learning-and-data-science-work-with-natural-languageprocessing/

[53] https://d2l.ai/chapter_recommender-systems/movielens.html

[54] https://grouplens.org/datasets/movielens/

[56] https://towardsdatascience.com/recommender-system-bayesian-personalized rankingfrom-implicit-feedback-78684bfcddf6

[57] https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401

[58] https://www.anaconda.fr/

[59] https://openclassrooms.com/

[79] https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

[80] https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks

[81] https://www.quora.com/What-is-leaky-ReLU

[82] https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526

[83] https://www.python.org/doc/essays/blurb/

[84] https://www.quora.com/What-is-a-kernel-in-Kaggle/answer/Bin-Zhu-33

[85] https://www.kaggle.com/docs/notebooks

[86] https://www.kaggle.com/rounakbanik/the-movies-dataset

[88] https://c3.ai/glossary/data-science/mean-absolute-error/

[89] https://www.baeldung.com/cs/top-n-accuracy-metrics

[90] https://www.analyticsvidhya.com/blog/2021/06/dimensionality-reduction-using-autoencoders-in-python/

[91] https://msirtadgrp2.files.wordpress.com/2015/02/perceptron-multicouche.pdf

[92] https://neptune.ai/blog/cross-validation-in-machine-learning-how-to-do-it-right

[93] https://machinelearningmastery.com/using-singular-value-decomposition-to-build-a-recommender-system/

[95] https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea