



République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie
Houari Boumediene

FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE
DEPARTEMENT INFORMATIQUE

Mémoire de Licence

Filière : Informatique

Spécialité : Ingénierie des Systèmes d'Information et des Logiciels

THÈME

RECOMMANDATION HYBRIDE BASÉE DEEP LEARNING ET
ANALYSE DE SENTIMENTS

Sujet Proposé et encadré par :
Dr BERKANI Lamia

Soutenu le : **07/07/2021**

Présenté par :
DIRAHOUI Balkis Bouthaina
SAID Faten Racha

Devant le jury composé de :
Dr SELMOUNE N. Président
Dr HACHEMI A. Membre

Binôme n° : 043/21

Résumé

Avec l'émergence du web 2.0, les goûts et préférences des utilisateurs face aux produits proposés deviennent de plus en plus accrus. Ceux-ci sont traduits de différentes manières ; le plus souvent par le biais de notation.

Cependant, les systèmes de recommandation basés uniquement sur ce critère souffrent généralement de plusieurs problèmes, notamment du manque de données (*data sparsity*) ou encore du démarrage à froid (*cold start problem*). Afin d'atténuer l'impact de ces problèmes, nous introduisons l'analyse de sentiments de façon à étudier la dimension sémantique des données sur la recommandation.

Nous proposons dans ce travail une nouvelle approche nommée SC-NHF (*Sentiment Convolutional – Neural Hybrid Filtering*) qui combine l'hybridation des deux algorithmes de filtrage collaboratif et basé contenu selon une architecture neuronale (NHF combinant GMF et HybMLP) avec une analyse des sentiments utilisant des réseaux de neurones convolutifs (CNN).

Notre travail s'articule principalement sur trois axes : le premier exploite les avis des utilisateurs ; le second intègre les informations de ces utilisateurs et des *items* ; et le troisième se base sur les notes relatives à ces mêmes utilisateurs et *items*. L'objectif étant la prédiction d'une note sur une échelle de 1 à 5 décrivant l'intérêt que portera un utilisateur à un *item* donné.

Ainsi, la combinaison d'une analyse de sentiment à un filtrage hybride d'une part et collaboratif d'autre part, a permis d'étudier le problème sous différents aspects. Les résultats des expérimentations effectués sur différentes bases de tests ont montré la bonne performance de nos systèmes de recommandation par rapport aux différents modèles avec lesquels ils ont été comparés.

Mots-clés : Recommandation Hybride, Filtrage Collaboratif, Analyse de Sentiments, Deep Learning, CNN, GMF, MLP.

Abstract

With the emergence of web 2.0, the tastes and preferences of users regarding suggested products are becoming increasingly accessible. These preferences can be translated to several data collection methods, more commonly to ratings.

However, recommendation systems based exclusively on ratings usually suffer from a variety of problems, more particularly the data sparsity problem or the cold start problem. In order to reduce the impact of these issues, we introduce sentiment analysis to study the semantic dimension of recommendation data.

We proposed in this work a novel hybrid recommendation approach called SC-NHF (*Sentiment Convolutional – Neural Hybrid Filtering*) that is based on sentiment analysis (using CNN) and that combines both collaborative and content-based algorithms (in an architecture based on GMF and Hybrid MLP.)

Our work mainly revolves around three axes; the first exploits the opinions of users, the second integrates the information of the users and items, and the third is based on the ratings of these users and items. The main purpose being the prediction of the user's rating that describes the interest of a user for a given item.

Hence, the combination of sentiment analysis with hybrid filtering on one hand and collaborative filtering on the other allowed this research to study the problem from different perspectives. Our recommender system proved its efficiency when compared to existing models.

Keywords: Hybrid Recommendation, Collaborative Filtering, Sentiment Analysis, Deep Learning, CNN, GMF, MLP.

REMERCIEMENTS

Nous rendons tout d'abord grâce à Dieu tout-puissant pour nous avoir donné la force et la patience d'achever ce modeste travail.

Nous voudrions ensuite adresser toute notre reconnaissance à notre encadreur, Dr. Lamia BERKANI, pour sa disponibilité, sa patience et surtout ses judicieux conseils, qui ont aidé à alimenter notre réflexion.

Nous remercions également les membres du jury, pour nous avoir fait l'honneur d'accepter d'examiner notre travail.

Enfin, nous tenons à témoigner toute notre gratitude aux personnes ayant contribué de près ou de loin à la réalisation de ce mémoire, en commençant par notre chère famille et amis qui nous ont encouragé et soutenu tout au long de la préparation de ce projet.

Table des matières

INTRODUCTION	1
1 ÉTAT DE L'ART	2
1.1 INTRODUCTION.....	2
1.2 SYSTEMES DE RECOMMANDATION	2
1.2.1 <i>Définition et objectifs</i>	2
1.2.2 <i>Approches de recommandation</i>	3
a. Filtrage à base de contenu	3
b. Filtrage collaboratif	3
c. Filtrage hybride	6
1.2.3 <i>Profil utilisateur</i>	6
1.3 RECOMMANDATION ET APPRENTISSAGE AUTOMATIQUE / PROFOND.....	7
1.3.1 <i>Apprentissage automatique</i>	7
1.3.2 <i>Apprentissage profond et réseaux de neurones</i>	7
1.3.3 <i>Quelques travaux liés</i>	10
a. Recommandation basée sur l'apprentissage profond	10
b. Recommandation hybride basée apprentissage profond	12
1.4 SYSTEMES DE RECOMMANDATION ET ANALYSE DES SENTIMENTS.....	13
1.4.1 <i>Notion d'analyse de sentiments</i>	13
1.4.2 <i>Techniques d'analyse de sentiments</i>	14
a. Approche automatique	14
b. Approche à base de règles (lexique)	16
c. Approche hybride	16
1.4.3 <i>Systèmes de recommandation et analyse des sentiments</i>	16
1.5 CONCLUSION	18
2 CONCEPTION	19
2.1 INTRODUCTION.....	19
2.2 APPROCHE PROPOSEE	19
2.2.1 <i>Motivation</i>	19
2.2.2 <i>Prétraitements</i>	20
a. Prétraitements sur les avis.....	20
b. Prétraitements sur les notes.....	20
2.2.3 <i>Système de recommandation SC-NHF</i>	21
a. Description générale du modèle.....	21
b. Description des couches du modèle	22
2.3 PROCESSUS DE RECOMMANDATION	28
2.4 ENTRAÎNEMENT	29
2.4.1 <i>Gestion du jeu de données (Dataset)</i>	29
a. Exploitation des données.....	29
b. Type d'évaluation et répartition des données du dataset	30
2.4.2 <i>Entraînement</i>	30
a. Fonction de coût (<i>loss function</i>)	30
b. Algorithme d'optimisation (<i>optimizer</i>).....	31
2.5 CONCLUSION	31
3 IMPLEMENTATION ET EXPERIMENTATIONS	32
3.1 INTRODUCTION.....	32
3.2 ENVIRONNEMENT DE DEVELOPPEMENT	32
3.2.1 <i>Environnement logiciel</i>	32
a. Langage de programmation	32
b. Bibliothèques	32
c. Google Colaboratory	33
3.2.2 <i>Description de notre système de recommandation</i>	33

3.3	METRIQUE D'ÉVALUATION	36
3.4	MODELES DE COMPARAISON	37
3.5	ÉVALUATIONS DES MODELES	37
3.5.1	<i>Description des bases de données</i>	37
3.5.2	<i>Évaluations préliminaires</i>	38
3.5.3	<i>Impact de l'analyse de sentiments sur la recommandation collaborative</i>	42
3.5.4	<i>Impact de l'analyse de sentiments sur la recommandation hybride</i>	42
a.	Évaluation avec utilisateurs connus (classique)	42
b.	Évaluation avec utilisateurs inconnus	43
3.5.5	<i>Discussion générale</i>	43
3.6	CONCLUSION	44
CONCLUSION		45
LISTE DES TABLEAUX		46
TABLE DES FIGURES		47
REFERENCES		48

Introduction

La taille des données et des informations sur le web a vu une importante évolution durant ces dernières années. Cependant, même si cette évolution a apporté de nombreux avantages, il faut noter qu'elle a également engendré plusieurs inconvénients. En effet, les utilisateurs peuvent facilement se retrouver perdus à travers des milliers, voire des millions d'informations, et la prise de décision, sans perte de temps deviendra l'une des tâches les plus complexes.

Afin de remédier à ce problème, les systèmes de recommandation s'avèrent être une solution incontournable. Ces derniers réussissent à cerner les goûts des utilisateurs, et facilitent nettement leurs choix à travers les données récoltées. En effet, les nouvelles technologies ont permis une meilleure communication particulièrement dans les sites e-commerce (commerce électronique) où l'avis et les notes de l'utilisateur définissent en grande partie son intérêt pour les articles proposés. Pour analyser ces données, il existe diverses méthodes utilisées par ces systèmes, à l'instar des différentes techniques de filtrage ou encore celles basées sur l'analyse de sentiments des utilisateurs.

Dans le cadre de notre projet de fin d'études de licence, nous nous intéressons à l'impact des avis des utilisateurs sur la recommandation. Celui-ci s'appuie sur des techniques d'apprentissage profond (*Deep learning*) et d'analyse de sentiments. Ainsi, nous présentons une nouvelle approche nommée SC-NHF (*Sentiment Convolutional – Neural Hybrid Filtering*) qui combine l'hybridation des deux algorithmes de filtrage collaboratif et basé contenu à une analyse des sentiments. Notre modèle s'inspire des travaux de Kerouba et Zeghoud [1], basé sur le filtrage hybride neuronal, et Liu et al. [2], proposant une recommandation qui utilise des techniques d'analyse de sentiments.

Notre mémoire est structuré principalement en trois chapitres présentés comme suit :

Chapitre 1 – État de l'art : Dans cette partie, nous expliquons en premier lieu les concepts de base de la recommandation, ensuite, nous introduirons les notions d'apprentissage automatique/ profond ainsi que les notions d'analyse de sentiments, pour enfin aborder quelques travaux liés à l'utilisation de l'analyse de sentiments dans la recommandation.

Chapitre 2 – Conception : Dans ce chapitre, nous décrivons notre approche de conception combinant l'hybridation et l'analyse de sentiments selon une structure neuronale multicouches.

Chapitre 3 – Implémentation et expérimentation : Ce chapitre présente les détails de l'implémentation de notre système de recommandation ainsi que les étapes d'évaluation de notre approche en utilisant le *dataset* de Yelp et deux autres bases connues d'Amazon.

Nous concluons ce mémoire par un résumé de notre approche et quelques perspectives de développement futures.

Chapitre 1 :

État de l'art

1.1 Introduction

L'utilisation des systèmes de recommandation est de plus en plus fréquente dans les applications web. En effet, ces derniers sont devenus un atout incontournable pour plusieurs sites, notamment, les géants comme Amazon¹, EBay² et Alibaba³.

Ces géants, utilisent à travers leurs systèmes, différentes techniques de filtrage afin d'offrir une meilleure expérience au client. Leur objectif principal étant la satisfaction de l'utilisateur.

Parmi les différentes approches adoptées, nous retrouvons le filtrage collaboratif et le filtrage basé contenu. Il est vrai que ces derniers ont apporté une nouvelle ère dans les sites web, néanmoins, les résultats qu'ils produisent manquent souvent de précision, ainsi, de nouvelles approches ont vu le jour telle que l'hybridation ou encore l'introduction de l'analyse de sentiments qui, permettent d'améliorer l'efficacité et la précision de ces systèmes.

Ce chapitre est organisé comme suit : dans la section 2 nous introduisons les systèmes de recommandation et les différentes approches utilisées pour le filtrage d'informations. Les notions d'apprentissage automatique, apprentissage profond et analyse de sentiments sont ensuite introduites dans la section 3. Quelques travaux relatifs aux techniques de *deep learning* et à l'analyse de sentiments seront enfin exposés dans la section 4.

1.2 Systèmes de recommandation

1.2.1 Définition et objectifs

Les systèmes de recommandations sont définis comme étant des systèmes de filtrage d'informations qui ont pour but principal la satisfaction de l'utilisateur. Cela, à travers une expérience personnalisée où du contenu jugé pertinent lui est suggéré en fonction de ses goûts et préférences. Ces systèmes jouent également un rôle important dans la gestion de la déferlante d'informations, et minimisent considérablement le temps de recherche de l'utilisateur.

¹ <https://www.amazon.com/>

² <https://www.ebay.com/>

³ <https://www.alibaba.com/>

Il existe plusieurs approches de recommandation, dont les principaux sont : le filtrage à base de contenu, collaboratif et hybride.

1.2.2 Approches de recommandation

a. Filtrage à base de contenu

Le filtrage cognitif aussi appelé filtrage basé contenu ou FCB s'appuie principalement sur la recommandation de différents *items*⁴ en fonction de leurs descriptions. Ce type de filtrage n'a ainsi pas besoin des données des autres utilisateurs afin d'effectuer une recommandation.

Le filtrage cognitif présente plusieurs avantages :

- **Élimination du problème de démarrage à froid** : en effet, de nouveaux items peuvent être suggérés sans avoir à être notés au préalable par un nombre important d'utilisateurs (plus de détails sur le problème de démarrage à froid dans la section 2 partie filtrage collaboratif).
- **Indépendance des utilisateurs** : cette méthode n'a besoin que du profil d'un seul utilisateur, de ce fait, cette dernière n'a pas besoin d'une large communauté d'utilisateur pour faire des recommandations, seul le profil utilisateur suffit.

Néanmoins, on retrouve quelques inconvénients :

- **Effet entonnoir** : l'utilisateur ne verra que les *items* qui ont les mêmes caractéristiques que les *items* qu'il a déjà évalués.
- **Limitation par la description des items** : la technique FCB dépend de la description des *items*, ainsi, afin de produire une bonne recommandation, la description des *items* doit être riche et le profil utilisateur doit être bien organisé, autrement, la recommandation serait biaisée.

b. Filtrage collaboratif

Le filtrage collaboratif ou FC se base sur l'idée que les personnes à la recherche d'informations devraient se servir de ce que d'autres ont déjà trouvé et évalué [3]. Elle consiste à mettre en correspondance ou en collaboration les utilisateurs avec des préférences similaires en calculant la similitude entre leurs profils, si l'utilisateur u_1 a interagit, explicitement ou implicitement, avec un *item* i_1 et s'il est jugé que l'utilisateur u_2 a un profil similaire à l'utilisateur u_1 , alors l'*item* i_1 sera recommandé à ce dernier.

Les techniques du filtrage collaboratif peuvent être divisées en deux catégories : le filtrage basé mémoire et le filtrage basé modèle.

⁴ **Item** : élément d'un ensemble, peut être un produit de tous genre.

■ Filtrage collaboratif basé mémoire

Le filtrage collaboratif basé mémoire est principalement caractérisé par le fait qu'il n'utilise que les informations de la matrice *items-utilisateurs*⁵. Ainsi, les algorithmes du filtrage collaboratif basé mémoire passent par deux étapes majeures, la première qui est l'établissement de communautés (voisinages), ceci, en calculant la similarité entre les utilisateurs ou les items selon différentes méthodes telles que la similarité vectorielle (*Cosine similarity*) ou la corrélation de Pearson, la seconde est celle du calcul de prédiction en se basant sur les communautés établies précédemment, l'une des méthodes les plus utilisées est celle de la somme pondérée. [1]

Le filtrage collaboratif basé mémoire peut être réalisé de 2 façons :

- Le filtrage basé utilisateur (*user-based CF*)

Avant de suggérer un *item* à un utilisateur, cette méthode calcule si les autres utilisateurs qui ont un profil similaire à ce dernier ont interagi avec.

- Le filtrage basé item (*item-based CF*)

Contrairement à la méthode basée utilisateur, cette méthode est centrée sur les *items*. Pour se faire, les *items* doivent d'abord être regroupés puis comparés, deux *items* sont dits similaires si les utilisateurs interagissent avec eux de la même manière.

■ Filtrage collaboratif basé modèle

Le filtrage basé modèle fait appel à des modèles d'apprentissage automatique pour la recommandation. En effet, le processus de création de modèle peut être effectué à l'aide de techniques d'apprentissage automatique, de *data mining*⁶ ou de décomposition matricielle. Des exemples de ces techniques incluent *Singular Value Décomposition* (SVD), *Matrix Completion Technique*, *Latent Semantic methods*, *Regression* et *Clustering*, [4] ou encore *Matrix Factorization* (MF).

- Factorisation matricielle (*matrix factorization*) :

Par définition, la factorisation d'une matrice consiste à décomposer une matrice en plusieurs sous-matrices (il suffira de calculer le produit de ces sous matrices pour obtenir la matrice initiale). Ainsi, appliquer ce procédé sur la matrice d'évaluation regroupant les *users*⁷ et les *items* permet à la fois de regrouper les caractéristiques de chaque *user* et *item* séparément mais aussi et surtout réduire la matrice d'évaluation qui est bien souvent creuse⁸ ce qui a pour avantage d'atteindre des temps de traitement bien plus compétitif et une économie de l'espace de stockage grâce à une structure de données de plus concise et pertinente.

⁵ **Matrice items-utilisateur** : matrice qui a pour lignes les utilisateurs et pour colonnes les *items*, tel que le croisement d'une ligne *u* et d'une colonne *i* représente la note de l'utilisateur *u* sur l'*item* *i*.

⁶ **Data mining** : représente l'extraction d'un savoir ou d'une connaissance à partir de grandes quantités de données, par des méthodes automatiques ou semi-automatiques.

⁷ **Users** : Mot anglais qui signifie utilisateurs.

⁸ **Creuse** : contient un grand nombre de relations ligne/colonne (item/user) nulles qui nécessitent un traitement ce qui représente dans ce cas une perte considérable d'espace mémoire et de temps de calcul.

La figure suivante décrit le procédé de factorisation matricielle :

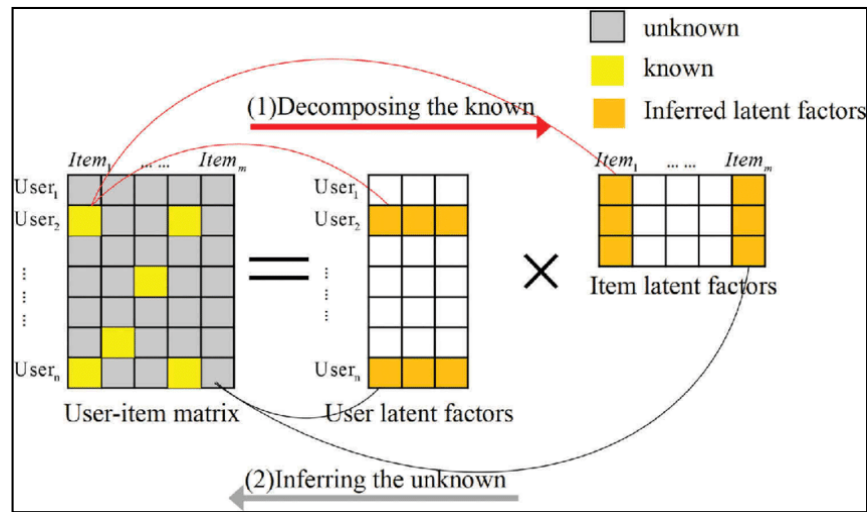


Figure 1.1: Factorisation de la matrice d'évaluation [5]

Cela se traduit par la formule mathématique suivante :

$$R_{(m,n)} = U_{(m,k)} \times I_{(k,n)} \quad (1.1)$$

Avec :

$R_{(m,n)}$: Matrice des évaluations avec m utilisateurs et n items.

$U_{(m,k)}$: Matrice des utilisateurs.

$I_{(k,n)}$: Matrice des items.

Le filtrage collaboratif (FC) présente ainsi plusieurs avantages :

- **Effet de surprise (*serendipity*)** : un *item* qui n'apparaît pas dans le profil de l'utilisateur peut tout de même être recommandé à ce dernier.
- **Élimination du problème de dépendance de description** : La technique FC n'a pas besoin de connaître la description des *items* pour effectuer une recommandation, les interactions (comme les notes par exemple) suffisent.

Malgré le succès des techniques FC, leur utilisation a révélé quelques désavantages :

- **Le problème de démarrage à froid (*cold start problem*)** :

C'est l'un des problèmes majeurs qui mènent à la réduction des performances des systèmes de recommandation. Il existe deux types de problèmes de démarrage à froid, le problème pour un nouvel utilisateur (*user cold start*) et le problème pour un nouvel item (*item cold start*). Le *user cold start problem* se produit quand un utilisateur se connecte au système pour la première fois. Le deuxième type, l'*item cold start problem* est le résultat de l'ajout d'un

nouvel *item* dans le catalogue. Ainsi, le système ne disposant pas d'assez d'informations (manque d'interactions) sur l'utilisateur ou l'*item*, il se retrouve dans la difficulté à effectuer une recommandation.

- **Le problème de La rareté d'information (*Data Sparsity*) :**

Le manque d'information consiste un réel problème pour ce type de filtrage. Étant donné que les utilisateurs n'évaluent pas tous les *items*, la matrice *items*-utilisateur peut rapidement se retrouver creuse, ce qui rend la recommandation moins efficace.

Nous pouvons ainsi conclure que les deux méthodes de filtrages classiques présentent des problèmes, afin d'y pallier, l'approche hybride a été conçue. Celle-ci repose sur des techniques d'hybridation des deux méthodes citées précédemment permettant d'exploiter les avantages de chacune, et de ce fait, en tirer de meilleures performances.

c. Filtrage hybride

L'approche hybride a pour principe la combinaison de plusieurs algorithmes de filtrages dans le but d'optimiser leur performance et, donc, minimiser leurs inconvénients lorsqu'ils sont utilisés seuls. Ainsi, cette approche permet limiter les principaux problèmes des méthodes de filtrages classiques, dont le problème de démarrage à froid, et cela en adoptant par exemple l'hybridation entre le FC et le FCB.

Plusieurs systèmes de recommandation hybrides se basant sur les profils utilisateurs existent, l'un des premiers est le système FAB [6]. Ce système recommande un *item* à l'utilisateur si le contenu de l'*item* est similaire au profil de l'utilisateur et si les voisins de cet utilisateur l'ont apprécié. Il combine ainsi le filtrage basé contenu et collaboratif en se basant sur les profils utilisateurs.

1.2.3 Profil utilisateur

Un profil utilisateur peut être défini comme étant le résultat du processus d'identification des données relatives aux goûts de l'utilisateur. Il consiste en l'ensemble des informations collectées à propos de l'utilisateur, ces informations peuvent ainsi être ses centres d'intérêts, ses préférences sous forme d'évaluations, les *items* avec lesquels il a interagi, etc. Il représente ainsi l'ensemble des caractéristiques de l'utilisateur.

Il existe deux principales approches de profilage des utilisateurs : le profilage explicite et implicite.

- **Profilage explicite** : consiste en l'analyse statique des utilisateurs et de leurs caractéristiques explicites. Il s'agit dans cette approche de collecter des données sur l'utilisateur d'une façon explicite en remplissant des formulaires, ou en évaluant les *items*.
- **Profilage implicite** : contrairement à la première approche, celle-ci utilise les caractéristiques implicites de l'utilisateur ; qui se base principalement sur le comportement de l'utilisateur tiré de ses interactions passées avec son environnement (exemple le click d'une souris, le temps passé sur un article, le suivi du curseur etc.).

Bien que les méthodes de filtrage cités précédemment apportent des résultats importants, l'introduction de l'apprentissage automatique à révolutionner les systèmes de recommandation et s'est montré très performant dans ce domaine.

1.3 Recommandation et apprentissage automatique / profond

1.3.1 Apprentissage automatique

Le *Machine Learning* ou l'apprentissage automatique est un sous-domaine de l'intelligence artificielle. Selon son inventeur, Arthur Samuel (1959) le *Machine Learning* est la science de donner à une machine la capacité d'apprendre, sans la programmer de façon explicite. Les algorithmes d'apprentissage automatique réussissent donc à apprendre d'une façon autonome à effectuer certaines tâches. Les techniques de *Machine Learning* peuvent être divisées en deux techniques principales, l'apprentissage supervisé (*Supervised Learning*) et L'apprentissage non supervisé (*Unsupervised Learning*).

- **Apprentissage supervisé** : L'apprentissage supervisé est la technique la plus utilisée dans l'apprentissage automatique, elle consiste à manipuler un ensemble de données étiquetées⁹ qui seront ensuite exploitées par les algorithmes d'apprentissage afin d'avoir comme résultat des caractéristiques ou des *patterns*. Ces derniers, devons permettre au modèle d'apprentissage d'associer à son tour une étiquette correctement lorsqu'il fait face à de nouvelles données.
- **Apprentissage non supervisé** : Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé n'a pas de données étiquetées. Afin d'obtenir ses résultats, cette technique utilise le plus généralement des algorithmes de *clustering*¹⁰ en regroupant les données ayant les mêmes caractéristiques.

A travers le temps, plusieurs techniques d'apprentissage supervisé ont vu le jour, notamment les réseaux de neurones (ou RNs) qui sont devenus une technologie presque incontournable pour permettre aux machines d'apprendre de façon autonome, nous présentons ci-dessus cet algorithme tout en donnant des exemples concrets de réseaux de neurone connus.

1.3.2 Apprentissage profond et réseaux de neurones

Les réseaux de neurones peuvent être aperçus comme étant une imitation simple d'un réseau de neurones biologique. Comme le montre la figure ci-dessous, un neurone biologique reçoit un signal et le capte avec ses récepteurs (dendrites), ensuite ces informations sont traitées par le corps cellulaire qui est chargé de faire les traitements (ou des sommes d'informations recueillis par les dendrites). Après avoir été traité, ce signal passe par l'axone (qui représente

⁹ **Données étiquetées** : données auxquels on a préalablement associé une étiquette, qu'on collecte afin d'entraîner les modèles d'apprentissage automatique.

¹⁰ **Clustering** : une méthode d'analyse statistique utilisée pour organiser des données brutes en silos homogènes.

le fil conducteur entre neurones), les synapses quant à elles jouent le rôle de liaison entre neurones (et donc leur permet de communiquer entre eux).

L'architecture d'un neurone artificielle peut être représentée en parallèle. Les pondérations de chaque élément ($w_{ij} * x_i$) peuvent être vues comme étant les dendrites ou des synapses, le corps cellulaire comme une application d'une fonction d'activation¹¹ à la somme des entrées pondérées et enfin, l'axone comme étant la sortie (le résultat final) du modèle.

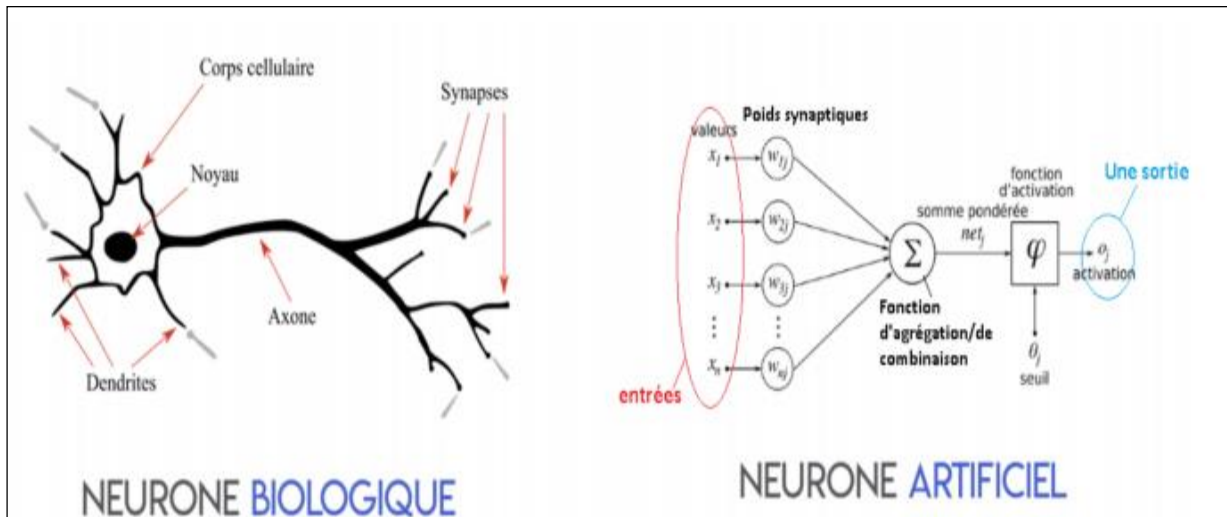


Figure 1.2 : Schématisation d'un neurone biologique par un neurone artificiel [7].

Afin de créer un réseau de neurones, il suffit de connecter un nombre de neurones entre eux en les regroupant en colonnes (appelées couches). Comme montré par la figure ci-dessous, un réseau de neurones contient en général une couche d'entrée (à gauche) où les données brutes sont reçues, une couche de sortie (à droite), et une couche cachée qui regroupe N autres couches qui communiquent entre elles, tel que, chaque couche reçoit une entrée de la couche précédente, effectue un traitement et renvoie sa sortie à la couche suivante.

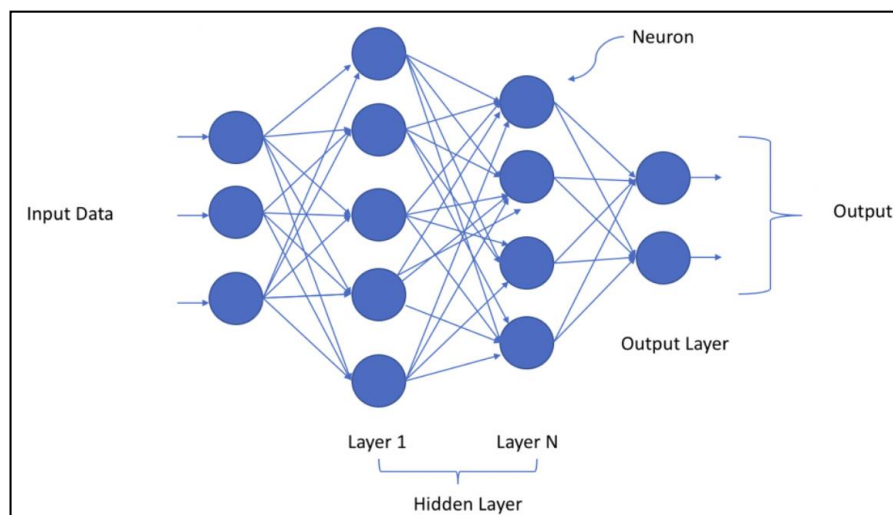


Figure 1.3: Schéma d'un réseau de neurones [8].

¹¹ **Fonction d'activation** : Fonction mathématique généralement non linéaire a pour rôle de déterminer si on active ou non une réponse du neurone. Il en existe plusieurs, nous pouvons citer la fonction d'activation « ReLU » (Rectified Linear Unit).

Au fil du temps, la taille des couches des réseaux de neurones a significativement augmenté, ce qui a mené à la naissance de l'**apprentissage profond**.

L'apprentissage profond est considéré comme un sous-domaine de l'apprentissage automatique puisqu'il se base sur des réseaux de neurones.

Ses performances sont particulièrement remarquées lorsque la taille des données est considérable et ses techniques sont notamment utilisées dans l'analyse de sentiments, le traitement naturel du langage (NLP) de manière plus générale, le *computer vision* (ou vision par ordinateur) ainsi que de nombreux autres sous domaines de l'intelligence artificielle.

Il existe plusieurs modèles de réseaux de neurones, nous détaillerons à titre d'exemple le Perceptron Multicouche (MLP) ainsi que le Réseau de Neurones Convolutifs (CNN) :

■ Perceptron multicouche (Multilayer perceptron MLP)

Le perceptron multicouche est organisé comme son nom l'indique en plusieurs sous couches, chacune constituée d'un nombre variable de neurones. Au sein de ces couches circule une information dans un sens unique qui est : de la couche d'entrée vers la sortie (dernière couche) ; c'est ce qu'on appelle un réseau à propagation directe (*feedforward*).

■ Réseaux de neurones convolutifs (Convolutional neural network CNN)

Les réseaux de neurones convolutifs particulièrement connus pour leurs remarquables résultats lorsqu'il s'agit de classification d'images s'introduisent de plus en plus dans le domaine du traitement automatique du langage dû aux performants résultats qu'ils offrent lors de la classification de texte.

La structure des CNNs est souvent composée de couches convolutives permettant d'extraire les caractéristiques des données en entrées (*feature extraction*) suivies de couches de classifications qui permettent de prédire le résultat final selon les classes définies pour la sortie.

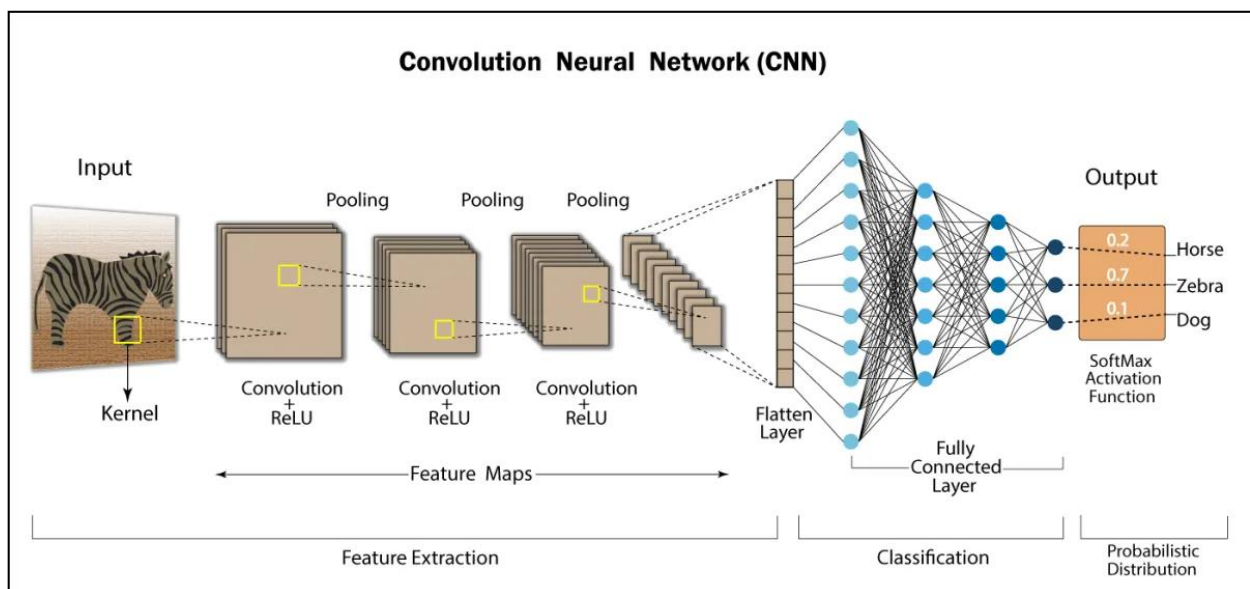


Figure 1.4: Architecture d'un réseau de neurones convolutif ayant pour entrée une image. [9]

La convolution est un outil mathématique qui permet d'extraire des informations en appliquant un ou plusieurs filtres¹² à une matrice donnée. Son principe se résume à faire déplacer une fenêtre (matrice) à travers une autre matrice de dimension supérieure et effectuer des opérations selon un pas. Cette couche est souvent suivie de *Max pooling* qui est une opération qui consiste réduire la taille de la matrice en la découpant en sous matrices de la même dimension, puis garder la valeur maximale de chaque sous matrice. Ceci nous permet de réduire à chaque fois le nombre d'opérations mathématiques tout en préservant leurs caractéristiques importantes.

Pour ce qui est de la couche *flatten* (mis-à-plat), celle-ci prend la totalité des valeurs des matrices précédemment calculées, les empile, et les envoie à la couche d'entrée du réseau de neurones qui aura pour sortie le résultat de la classification.

1.3.3 Quelques travaux liés

Les systèmes de recommandation personnalisés sont devenus incontournables dans divers domaines, et bien que les algorithmes classiques de filtrage se sont prouvés efficaces, ceux-ci ont quelques inconvénients. Afin d'améliorer la précision de ces algorithmes, les différentes techniques d'apprentissage profond ont été introduites dans les systèmes de recommandation.

Nous présentons ci-dessous quelques travaux de recherches se basant sur le l'apprentissage profond.

a. Recommandation basée sur l'apprentissage profond

Nous présentons dans cette partie les travaux de recherche récentes de Ni et al [10] , leur approche **RM-DL** (*Recommendation Model based on Deep Representation Learning*) a pour problématique principale d'améliorer les systèmes de recommandation classiques basé sur l'apprentissage profond en essayant de résoudre 3 problèmes principaux qui sont récurrents dans ces derniers : le manque de précision, l'inexploitation des informations auxiliaire (texte par exemple) et le fait qu'ils ignorent l'influence de l'historique de préférences de l'utilisateur sur ses préférences futurs. Afin de remédier à ces problèmes, ces derniers proposent une approche qui introduit l'analyse des caractéristiques du texte et images et qui est basé principalement sur 2 étapes : le prétraitement des informations (*Information Preprocessing*) et la représentation des caractéristiques (*Feature Representation*), comme le montre la figure ci-dessous.

¹² **Filtre** : Une fonction mathématique qui peut être de n'importe quel type : le maximum, la moyenne, une somme pondérée, etc.

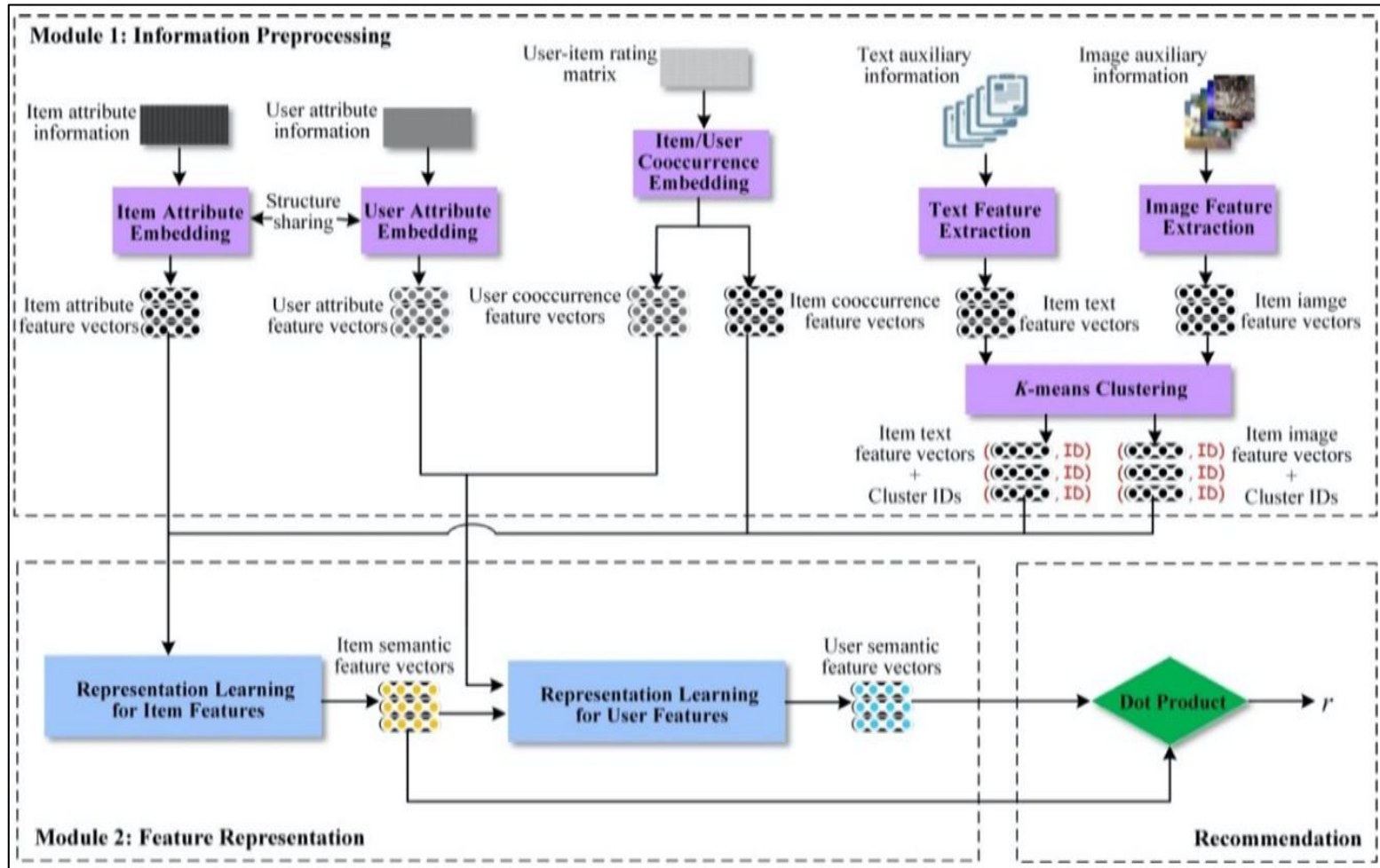


Figure 1.5 : Schéma représentant l'architecture de l'approche RM-DL [10]

Dans le premier module (Prétraitement des informations), le but principal est de collectionner les vecteurs primitifs pour les *items* et *users*, pour qu'ils soient par la suite exploités dans la prochaine étape en utilisant plusieurs techniques tel que l'*embedding*¹³, K-means clustering [10] ou encore BERT [10] qui permet d'extraire les caractéristiques du texte. Ces vecteurs seront ensuite exploités dans le deuxième module (représentation des caractéristiques), qui aura pour sortie deux vecteurs (un vecteur pour user et un autre pour item). Ils utilisent par après un réseau de neurone CNN¹⁴ à 4 couches afin d'obtenir le vecteur des caractéristiques sémantiques des items, ainsi qu'un nouveau réseau de neurone AIGRU [10] avec la méthode BPR [10] qui aura pour sortie un vecteur des caractéristiques sémantiques pour les utilisateurs. Ces 2 derniers vecteurs sont ensuite exploités pour obtenir les top-N recommandations pour un *user*.

Les auteurs ont comparé l'approche RM-DL avec plusieurs modèles de comparaison et cette dernière fournie de meilleurs résultats.

En résumé, dans ce travail les auteurs ont exploité les différentes caractéristiques des informations auxiliaires (image et texte) afin d'améliorer les systèmes de recommandations classiques basé sur l'apprentissage profond.

¹³ *Embedding* : plus de détails dans la section 1.4.2.

¹⁴ *CNN* : plus de détails sur l'algorithme dans la section 1.3.2.

b. Recommandation hybride basée apprentissage profond

Dans le travail de Kerouba et Zeghoud [1], les auteurs essayent d'améliorer une approche existante de He et al [11] qui traite la problématique du MF classique en introduisant le filtrage cognitif (FCB) et donc apportée une hybridation en exploitant les données de l'utilisateur déjà recueillis qui seront données en entrée.

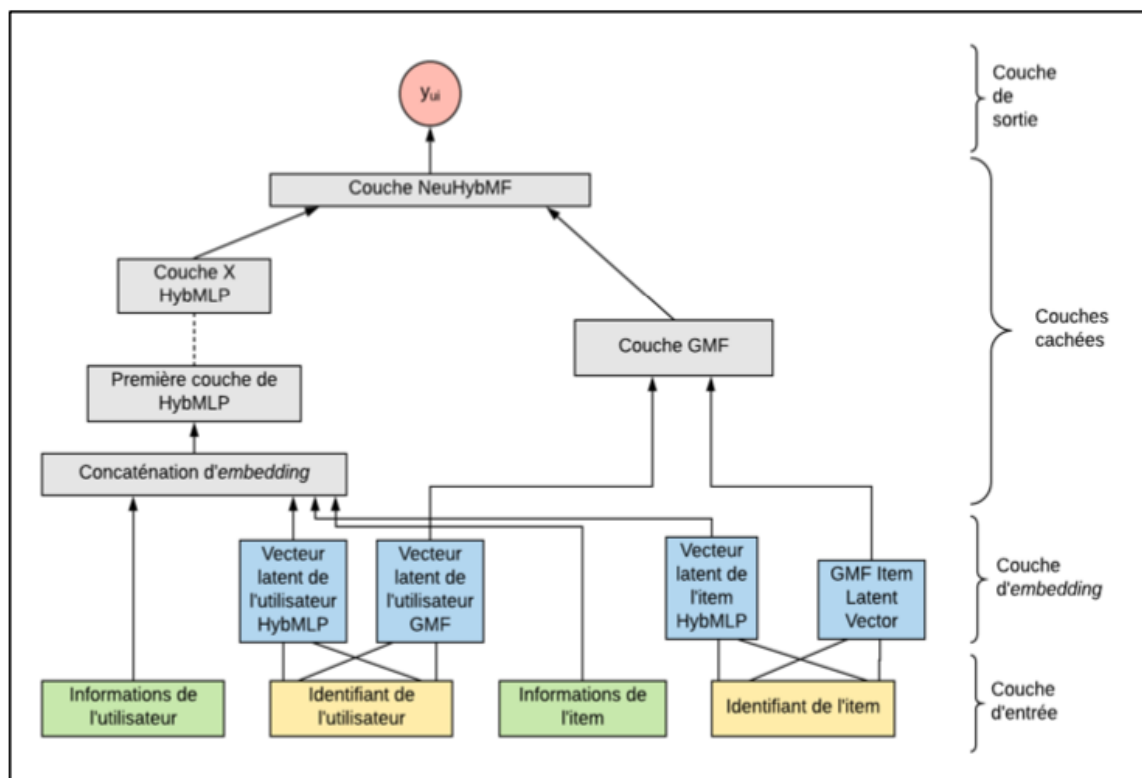


Figure 1.6 : Architecture du modèle avec l'hybridation [1]

Le modèle est constitué principalement de 4 couches, la première couche est la couche d'entrée ou on collectionne les différentes informations de l'utilisateur ou *item*. Ces informations seront ensuite concaténées dans la couche d'*embedding* qui représente la deuxième couche. Cette couche a pour but de représenter les caractéristiques des utilisateurs et *items* sous forme de vecteurs de nombres réels, et donc, d'avoir les différentes informations des utilisateurs et *items* sous forme de vecteurs de facteurs latents (i.e. caractéristiques cachées), ces vecteurs sont le résultat du produit des mots codé en *One-Hot Encoding*¹⁵ et le RN [1]. Ensuite, dans la troisième couche (couches cachées) ces 2 vecteurs seront donnés comme entrée au HybMLP (MLP hybride) qui a pour objectif d'entraîner la fonction prédictive, et donc de retourner 1 si l'utilisateur u a interagit avec l'item i , 0 sinon. Séparément, la couche GMF aura comme entrée les 2 vecteurs latents des utilisateurs et *items*, et servira comme une couche de multiplication. Il faut noter qu'ici que la GMF¹⁶ est utilisé et non la MF (voir détails section 2). Enfin, comme le montre la figure ci-dessus, la dernière sous-couche de cette troisième couche s'agit de la couche NeuHybMf (ou NHF) qui a pour but de combiner les 2 dernières couches obtenus afin d'avoir une meilleure précision. Le résultat de cette concaténation sera ensuite donné comme entrée à la 4ème couche (couche de

¹⁵**One-Hot Encoding** : plus de détails dans le chapitre 2.

¹⁶**GMF** : matrice de factorisation généralisée plus de détails dans le chapitre 2.

sortie), qui aura pour objectif de retourner le résultat de prédiction (qui est compris entre 0 et 1) ; cette couche utilise une fonction d'activation « Sigmoidale » : $\sigma(x) = 1 / [1 + e^x]$. [1]

Ce travail a ensuite été testé sur des jeux de données (MovieLens 100k¹⁷ et MovieLens 1M¹⁷, ainsi que la base Yelp¹⁸) afin de démontrer son efficacité en se basant sur différentes métriques (HR@K et NDCG@K) et semble donner de meilleurs résultats que l'approche classique.

En résumé, les auteurs de ce travail ont introduit les informations des utilisateurs/*items* et donc ont créé une hybridation grâce à l'introduction du filtrage cognitif dans le but d'améliorer l'article d'origine, ce qui semble avoir été prouvé après l'avoir testé avec des modèles de comparaisons.

Comme vu précédemment, le développement des techniques de l'apprentissage profond a réalisé dernièrement un progrès important dans la recommandation, la classification du texte ou l'analyse de sentiments est l'une des techniques dans lesquelles l'apprentissage profond se révèle particulièrement performant.

1.4 Systèmes de recommandation et analyse des sentiments

1.4.1 Notion d'analyse de sentiments

Le concept informatique d'analyse de sentiments (*sentiment analysis*) aussi connu sous le nom de fouille d'opinions (*opinion mining*) tire ses origines des sciences de la psychologie, la sociologie et de l'anthropologie.

Cette approche consiste à analyser le ressenti voir les émotions des individus vis-à-vis de certaines entités (que ce soit produit, documents ou encore services) proposées par différents types d'organisations. Elle est souvent appliquée dans les domaines tel que :

- Économie : Cibler les préférences des consommateurs, cela, dans une perspective d'amélioration de leurs produits et ainsi augmenter leurs ventes et revenus.
- Politique : Recueillir l'opinion des citoyens sur un politicien lors des élections présidentielles ou encore sonder leurs avis sur les nouvelles lois avant de les déclarer.
- Éducation : Reconnaître les méthodologies de d'enseignement les plus efficaces et proposer ainsi les programmes de cours les plus adéquats.

L'analyse de sentiments exploite les avis/commentaires des utilisateurs et les classifie selon un modèle préalablement défini, nous pouvons citer à titre d'exemple, les modèles qui classent les sentiments de leurs utilisateurs selon la polarité et l'intensité du jugement émis, une polarité est soit positive, soit négative, soit un mélange de ces deux valeurs, tandis que l'intensité montre le degré de positivité ou de négativité, et varie de faible à forte [12]; il existe également des modèles de classification plus élaborés qui notamment évaluent si le sentiment éprouvé est un sentiment de joie, colère, déception ...etc.

¹⁷ <https://grouplens.org/datasets/movielens>

¹⁸ <https://www.yelp.com/dataset/>

Pour ce faire, ces systèmes utilisent différentes techniques et algorithmes souvent basées sur le traitement automatique du langage naturel¹⁹, l'apprentissage automatique (*machine learning*) ou encore l'apprentissage profond (*deep learning*).

1.4.2 Techniques d'analyse de sentiments

Afin de construire des outils automatiques capable d'extraire des informations à partir de textes en langage naturel de manière à les structurer et à les rendre exploitables, nous utilisons différentes approches dont les principales sont schématisées dans la figure ci-dessous et expliquées dans les points qui suivent :

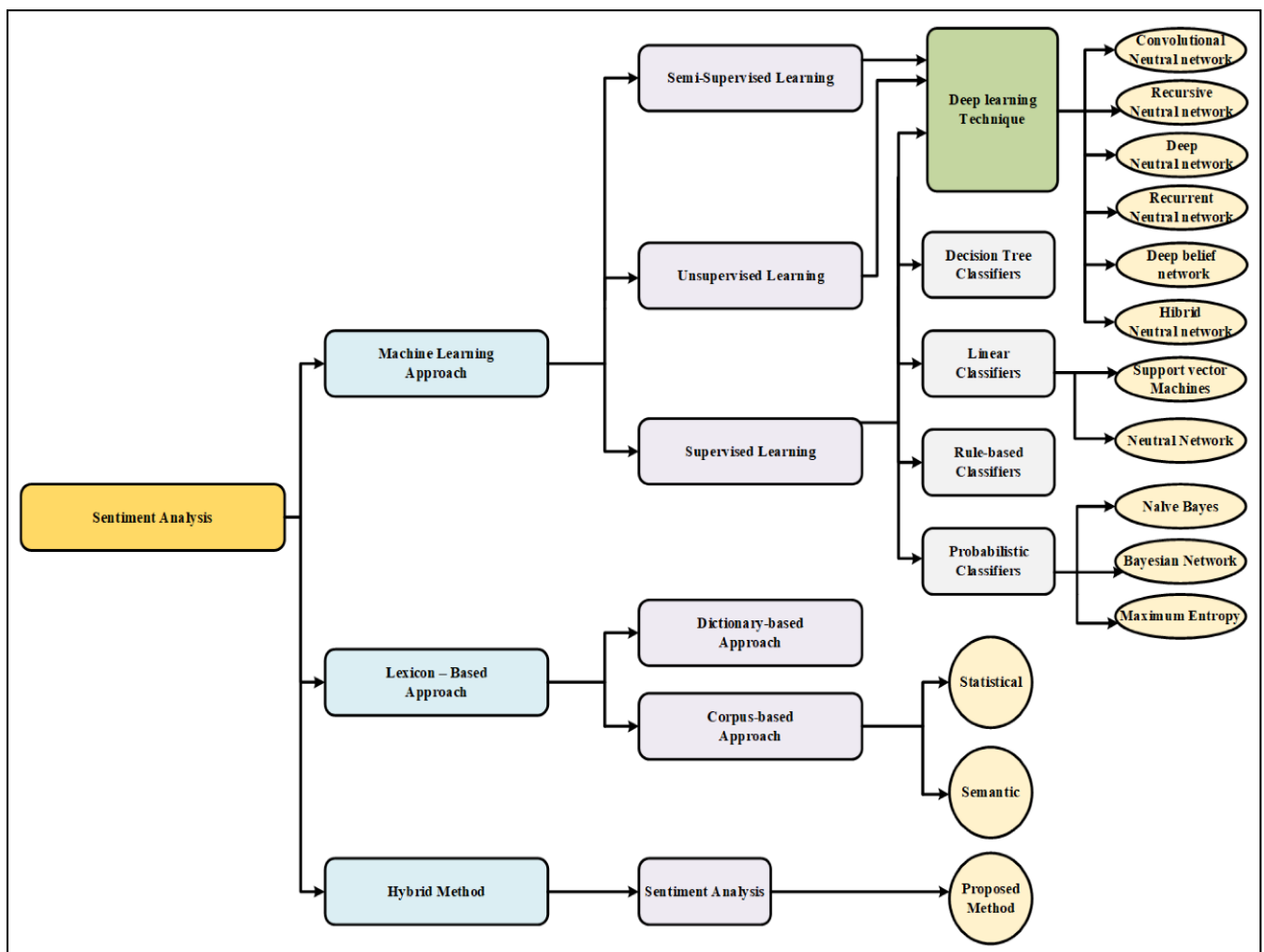


Figure 1.7 : Techniques d'analyse de sentiments [13]

a. Approche automatique

L'approche automatique repose sur les diverses techniques d'apprentissage automatique et d'apprentissage profond qui modélisent et déterminent des solutions au problème en intégrant un à plusieurs algorithmes et classifications.

¹⁹ **Traitement automatique du langage naturel** : Le traitement automatique du langage naturel, ou *natural language processing* (NLP) en anglais, est une branche de l'apprentissage automatique permettant aux modèles de comprendre et d'extraire des informations sémantiques du langage humain.

Les principales classifications que comporte cette approche sont la **classification probabiliste** qui utilise généralement l'algorithme du *Naïve Bayes* (NB), la **classification linéaire** qui fait le plus souvent appel à des algorithmes comme le *Support Vector machine* (SVM), ou encore, les **techniques de deep learning** qui sont de plus en plus adoptés contenu des résultats qu'ils apportent.

- **Technique de deep learning :**

Il existe différents algorithmes et techniques permettant la classification de texte grâce au *deep learning*, l'une des plus populaires est celle des réseaux de neurones convolutifs (*Convolutional Neural Networks CNN*), et cela, dû aux remarquables résultats qu'ils produisent dans le domaine de la classification.

Comme expliqué dans la section 1.3.2 un CNN prend en entrée une matrice qui, dans le cas de l'analyse de sentiments, la matrice qui est étudiée est celle représentant les vecteurs d'*embedding* de chaque mot qui constituent le texte à analyser. Cette matrice peut être traitée soit directement par la couche *embedding* du CNN ou alors passer par un prétraitement de « *word embedding* ».

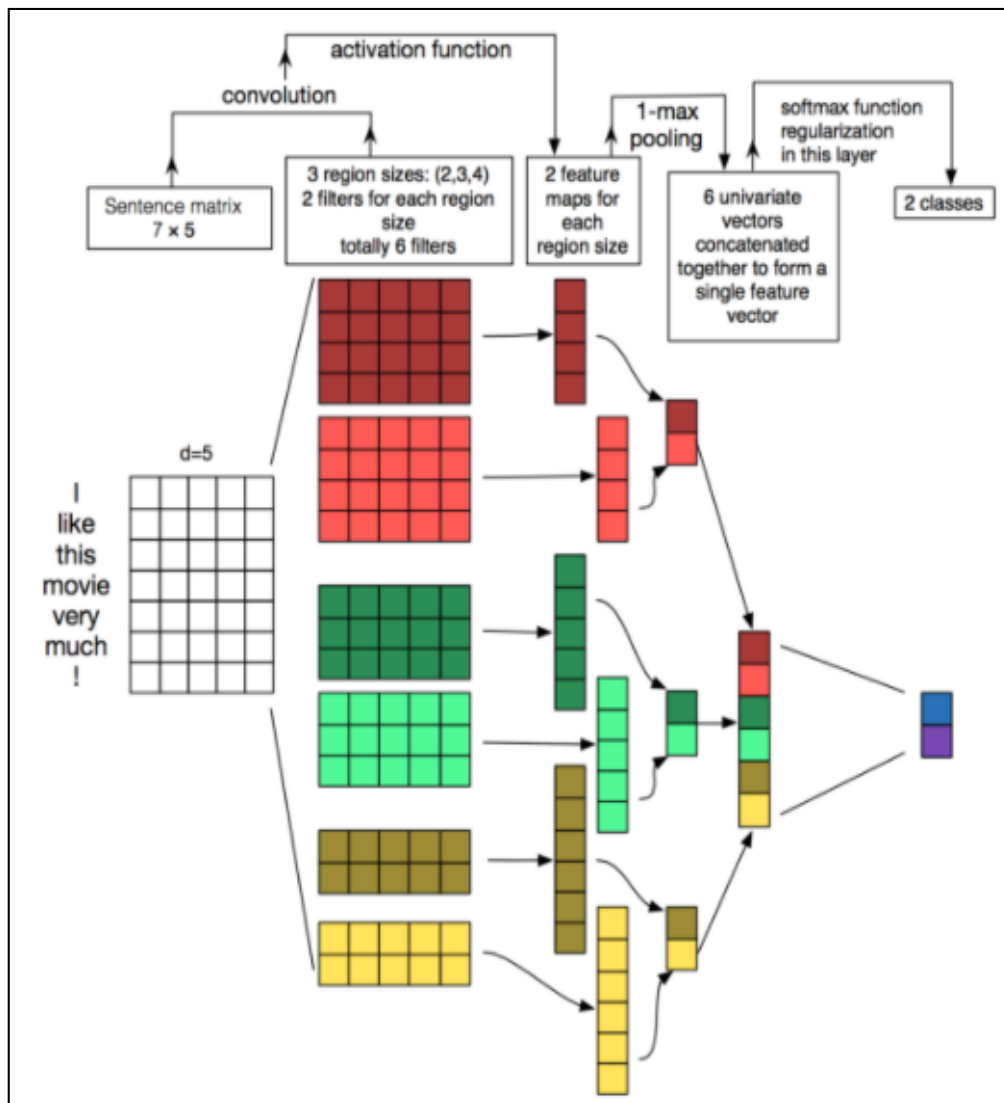


Figure 1.8: Classification d'une phrase en appliquant un réseau de neurone convolutif [14].

- *Word embedding* :

Le « *word embedding* » est une technique qui consiste à construire un espace vectoriel où chaque mot d'une phrase ou d'un texte est représenté par un vecteur multidimensionnel de nombres réels tel que calculer la distance entre deux mots nous permettrait de déduire leur similarité sémantique.

La logique derrière la construction de ces vecteurs repose sur le principe que chaque mot est défini par son contexte (les mots qui l'entourent), de ce fait, il devient possible de déduire la sémantique des mots en se basant sur les mots dont il est en relation avec.

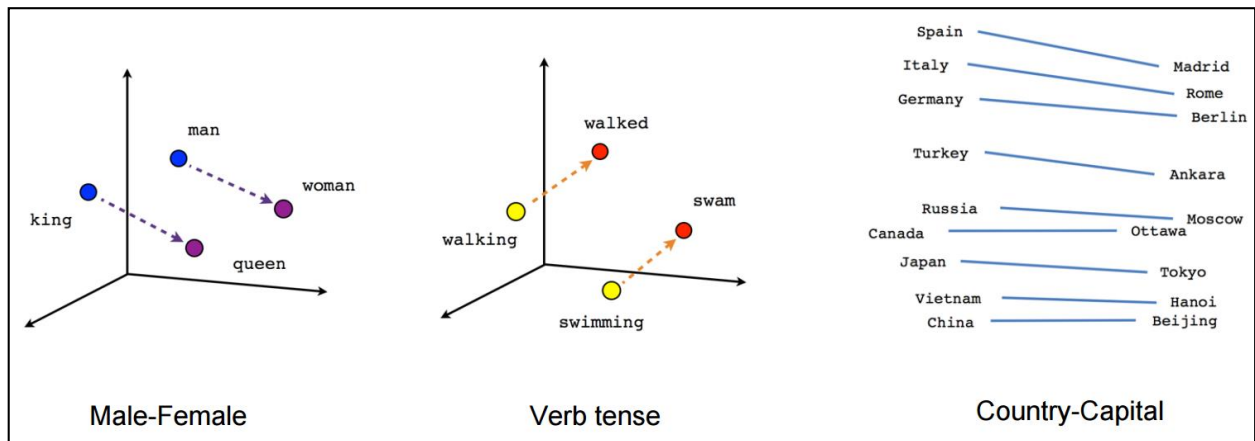


Figure 1.9 : Représentation de mots dans un espace vectoriel [15]

b. Approche à base de règles (lexique)

L'approche à base de règles (*Rule-Based*) aussi connu sous le nom d'approche à base de patrons linguistiques (*Lexicon-Based*) [16] se base principalement sur des règles grammaticales (éventuellement sémantiques) et des dictionnaires lexicaux pour étiqueter et catégoriser les expressions d'aspect dans le texte d'opinion. Le problème de ce type d'approches est qu'elles sont dépendantes du langage et du domaine, de plus, elles nécessitent une prédéfinition manuelle par des experts, ce qui est très coûteux.

c. Approche hybride

Dans cette approche, il suffit de combiner le meilleur des deux approches précédemment étudiées, les résultats démontrent souvent que les méthodes basées sur l'hybridation améliorent la précision de la recommandation.

1.4.3 Systèmes de recommandation et analyse des sentiments

L'analyse de sentiments est un domaine de recherche qui prend de plus en plus d'envergure au sein des systèmes de recommandations, cela, pour sa principale faculté d'analyser

l'opinion de l'utilisateur face aux *items* qui lui sont proposés, permettant ainsi de cibler au mieux les attentes et exigences de ses consommateurs.

Il existe plusieurs travaux de recommandation s'appuyant sur cette approche, parmi eux, **HRDR**²⁰ une étude [2] paru en 2019 dont la problématique vise à résoudre le problème de *data-sparsity* (i.e données disséminées) dont fait preuve les systèmes de recommandation lorsqu'il s'agit de représenter un utilisateur à partir des notes qu'il a attribué aux *items*, ainsi, ils proposent une approche combinant la représentation des notes et commentaires d'un *user* u à un *item* i le but étant de prédire la note $r_{u,i}$ qu'attribuera u à i .

L'architecture du modèle est résumée dans la figure ci-dessous :

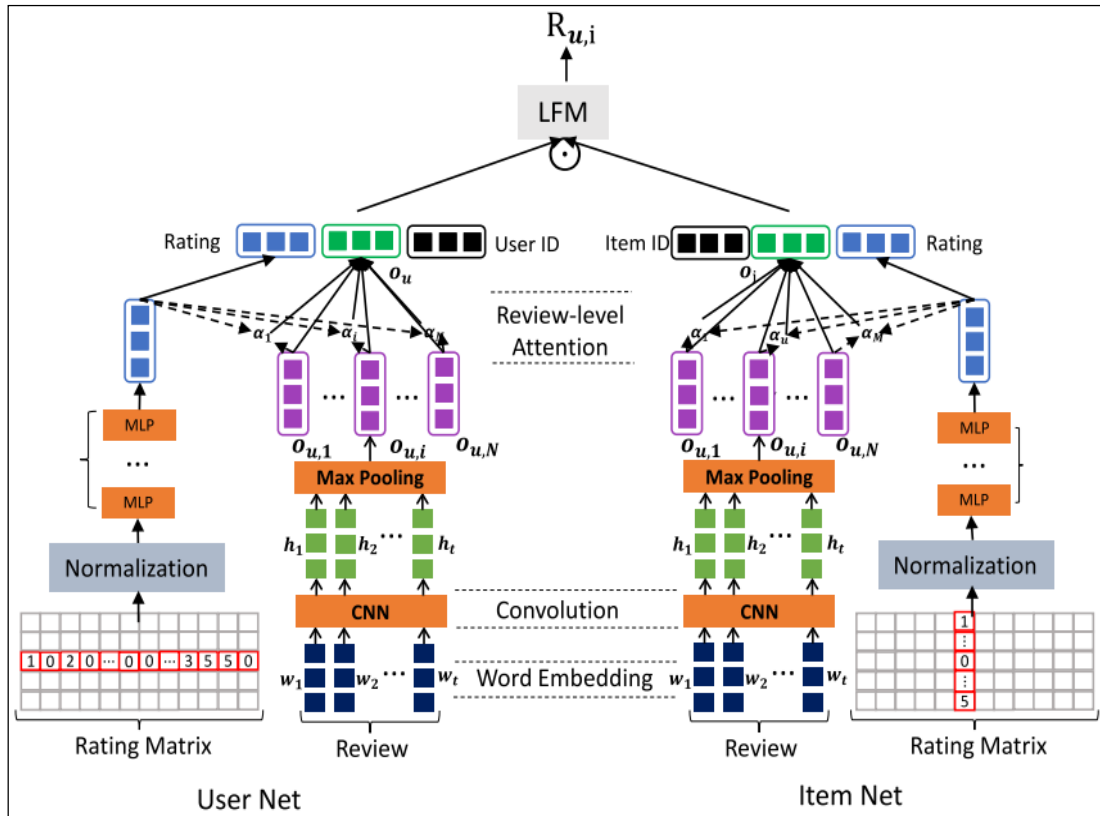


Figure 1.10: Architecture du HRDR : deux réseaux de neurones parallèles pour la représentation des users et des items [2].

L'architecture du HRDR se présente sous forme de deux réseaux de neurones parallèles l'un d'eux utilisé pour la représentation du *user* et l'autre de l'*item*. Les étapes sont les mêmes des deux côtés, nous détaillerons donc uniquement la partie de l'utilisateur. Celle-ci, étudie dans un premier temps les notes que l'utilisateur a attribué aux *items*, cela grâce à une matrice ayant pour lignes tous les *users* et pour colonnes tous les *items* du système, ainsi, en normalisant la ligne (vecteur) représentant l'utilisateur et ses précédentes notes ils obtiennent un vecteur dit normalisé (i.e. un modèle dont la complexité est réduite) qui sera passer par un perceptron multicouche (MLP) pour extraire les caractéristiques qui font que l'utilisateur étudié aime un certain article (cette étape est similaire du côté de l'*item* il suffira juste de prendre la colonne représentant les notes qui lui ont été attribuées).

²⁰ **HRDR** : le nom de l'approche proposée étant la « Hybrid neural recommendation with joint deep representation learning of ratings and reviews »

Ensuite, pour traiter les avis de ce même utilisateur ils les regroupent en leurs appliquant un traitement de *word embedding*²¹ et font passer la matrice résultante par un réseau neuronal convolutif (CNN), puis, en appliquant le *max pooling*²² ils obtiennent une représentation des avis de l'utilisateur qu'ils vont par la suite combiné (grâce à une transformation linéaire) avec les caractéristiques résultantes lors de la première étape (basé sur les notes) afin de filtrer les avis pertinents²³ de l'utilisateur.

Enfin, pour prédire la note qu'attribuera l'utilisateur au *items* ils appliquent le *Latent feature model* (LFM) sur les résultats obtenus des deux étapes précédemment expliqués avec l'identifiant de l'utilisateur et de l'*item*.

Pour conclure, les résultats obtenus sont nettement supérieurs à ceux des modèles déjà existants (PMF [17], HFT [18], CTR [19], JMARS [20], ConvMF+ [21], DeepCoNN [22], NARRE [23], TARMF [24]) sur les différents jeux de données (deux à partir de Yelp Dataset Challenge²⁴ (Yelp 2013, Yelp 2014) et deux autres à partir d'Amazon²⁵ (Video Games, Gourmet Food)). Le modèle a été évalué avec la métrique de la racine de l'écart quadratique moyen (Root Mean Square Error RMSE). Il a aussi été conclu qu'un modèle qui inclut le filtrage des avis pertinents des utilisateurs donnés de meilleurs résultats qu'un modèle sans.

1.5 Conclusion

Nous avons abordé dans ce chapitre les quelques concepts liés aux systèmes de recommandation, en passant par les principes de base de l'apprentissage automatique et de l'apprentissage profond. Nous avons introduit par la suite la notion d'analyse de sentiments et ses différentes techniques, ainsi que quelques travaux liés à la recommandation et à l'analyse de sentiments.

Dans le chapitre suivant, nous proposerons une approche hybride de recommandation tout en s'inspirant des principes et travaux précédemment cités.

²¹ **Word embedding** : technique utilisée pour le traitement de texte qui permet d'extraire le contexte d'un mot dans une phrase et déduire sa sémantique en se basant sur les mots dont il est en relation avec.

²² **Max pooling** : technique utilisée pour gagner en temps de calcul en réduisant la dimension d'une matrice donnée.

²³ **Pertinents** : les avis cohérents avec ce qu'il aime et n'aime pas.

²⁴ <https://www.yelp.com/dataset/challenge>

²⁵ <http://jmcauley.ucsd.edu/data/amazon/>

Chapitre 2 :

Conception

2.1 Introduction

Tel que nous l'avons présenté dans le chapitre précédent, il existe différentes techniques de conception de systèmes de recommandations, dont certaines se basent sur l'utilisation de l'apprentissage automatique ou profond et faisant parfois appel à l'analyse de sentiments.

Dans ce chapitre, nous proposons notre propre approche de recommandation basée sur les techniques de filtrage d'information et l'analyse des sentiments des utilisateurs. Nous utiliserons des techniques d'apprentissage profond.

2.2 Approche proposée

2.2.1 Motivation

Le travail de Kerouba et Zeghoud [1] présenté dans la section travaux liés du chapitre 1, a proposé une approche de recommandation hybride basée sur l'apprentissage profond. A travers ces recherches les auteurs ont appliqué une recommandation selon la fonction d'interaction, en d'autres termes, ils ont étudié si l'utilisateur va interagir ou non avec un *item* (une valeur booléenne est ainsi prédite).

Cependant, bien que celle-ci ait amélioré les performances de l'approche de FC neuronale (NCF) proposée par He et al. [11], il est quand même important de noter que seuls les données et identifiants des utilisateurs et des *items* ont été pris en considération. Par conséquent, la recommandation pourrait vite se retrouver biaisée ou pas assez pertinente.

L'introduction des caractéristiques explicites (avis des utilisateurs) pour la prédiction des notes des utilisateurs pourrait alors enrichir cette étude et donc atténuer le manque de précision de la recommandation.

De ce fait, dans le but d'améliorer cette approche, nous introduisons le traitement des avis de l'utilisateur dans la recommandation. Cette nouvelle architecture permettra la prédiction de la note que va attribuer un utilisateur à un *item* en exploitant non seulement le filtrage collaboratif et le filtrage cognitif (i.e. à travers les identifiants et informations des utilisateurs et *items*), mais également les caractéristiques explicites (i.e. à travers les avis donnés par les utilisateurs sur ces *items*).

Nous notons que nos travaux ont également été inspirés par l'approche de recommandation proposée dans l'étude récente de Liu et al. [2]. Cette dernière, décrite dans le chapitre 1, traite

les avis et notes des utilisateurs afin d'effectuer une prédiction sur la note qu'attribuera un utilisateur u à un *item* i .

2.2.2 Prétraitements

Nous décrivons dans ce qui suit, les différents prétraitements appliqués à nos données afin de les rendre exploitables par notre modèle :

a. Prétraitements sur les avis

Afin que notre système de recommandation puisse analyser les avis (i.e. commentaires) des utilisateurs, nous effectuons les différents traitements ci-dessous :

Nous commençons d'abord par appliquer ce qu'on appelle la *Tokenisation* qui consiste à transformer du texte en une série de « *tokens* » où chaque *token* représente un mot. Ensuite, afin d'obtenir une liste de mots (ou *tokens*) finale, nous convertissons d'abord tous les caractères des mots de l'avis en minuscule, puis supprimons la ponctuation et tous les caractères indésirables, une fois ceci effectué, nous enregistrons le tout sous forme de liste de mots.

Par la suite, nous enlevons ce qu'on appelle les « *stop words* » ou « les mots les plus fréquents ». Ces mots représentent les mots les plus récurrents qui n'apportent aucune information pertinente. Ils sont alors établis comme une liste de mots à nettoyer. Nous pouvons prendre comme exemple en anglais les mots '*so*' ou '*the*'. La liste obtenue après cette opération est transformée par après en suite de mots représentant ainsi le texte « nettoyé ».

Enfin, le texte résultant est prêt à être traité avec les techniques du *word embedding* (expliquée dans l'approche automatique du titre 1.4.2), où nous générons pour chaque avis une matrice de vecteurs d'*embedding* propre à chaque mot qui constitue cet avis.

b. Prétraitements sur les notes

Le traitement effectué sur les notes consiste à obtenir un vecteur de notes normalisé avec le *One-Hot encoding*.

Le *One-Hot encoding* étant une technique de codage de données catégoriques²⁶ sous forme de vecteur binaire qui construit un vocabulaire de N mots où le i -ème mot est représenté soit par un 0, soit par un 1.

Ainsi, pour chaque note donnée, un vecteur *One-hot* correspondant, comme représenté dans ce qui suit (le 1 détermine la catégorie à laquelle appartient la note) :

²⁶ **Données catégoriques** : données ayant un nombre limité de valeurs (exemple : une note = {1, 2, 3, 4, 5}).

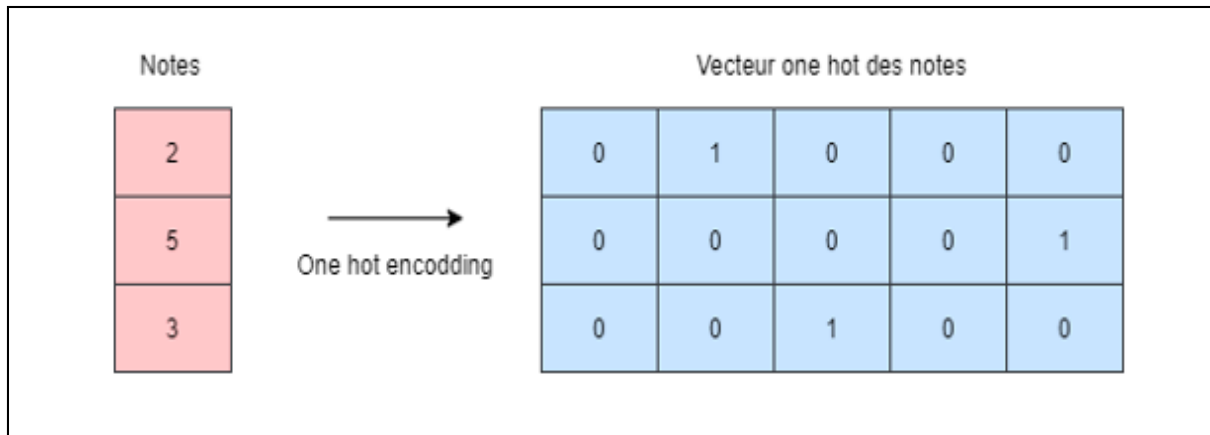


Figure 2.1 : Exemple de transformation des notes en vecteurs One Hot Encoding

2.2.3 Système de recommandation SC-NHF

a. Description générale du modèle

Le système de recommandation SC-NHF (*Sentiment Convolutional – Neural Hybrid Filtering*) se base sur un modèle d'apprentissage profond capable de prédire la note que l'utilisateur u donnera à un item i . L'approche que nous proposons repose sur une hybridation du filtrage cognitif avec le filtrage collaboratif, tout en introduisant l'analyse de sentiments supervisée par les notes afin de traiter les avis des utilisateurs sur les items. Nous exploitons les données explicites, identifiants et informations (données en entrée) des *users* et des *items* à travers les avis attribués par les utilisateurs ainsi que les vecteurs de facteurs latents²⁷.

Comme le décrit l'illustration ci-dessous, l'architecture du système de recommandation comporte quatre parties principales dont l'une réservée aux couches cachées. Celles-ci sont représentées par les modèles suivants : GMF (Factorisation Matricielle Généralisée) ; HybMLP (Perceptron Multicouches Hybride) ; CNN (Réseau Neuronale Convolutif).

Le schéma ci-dessous décrit l'architecture du modèle proposé :

²⁷ **Facteurs latents** : Caractéristiques cachées représentées par les identifiants des utilisateurs et des *items*.

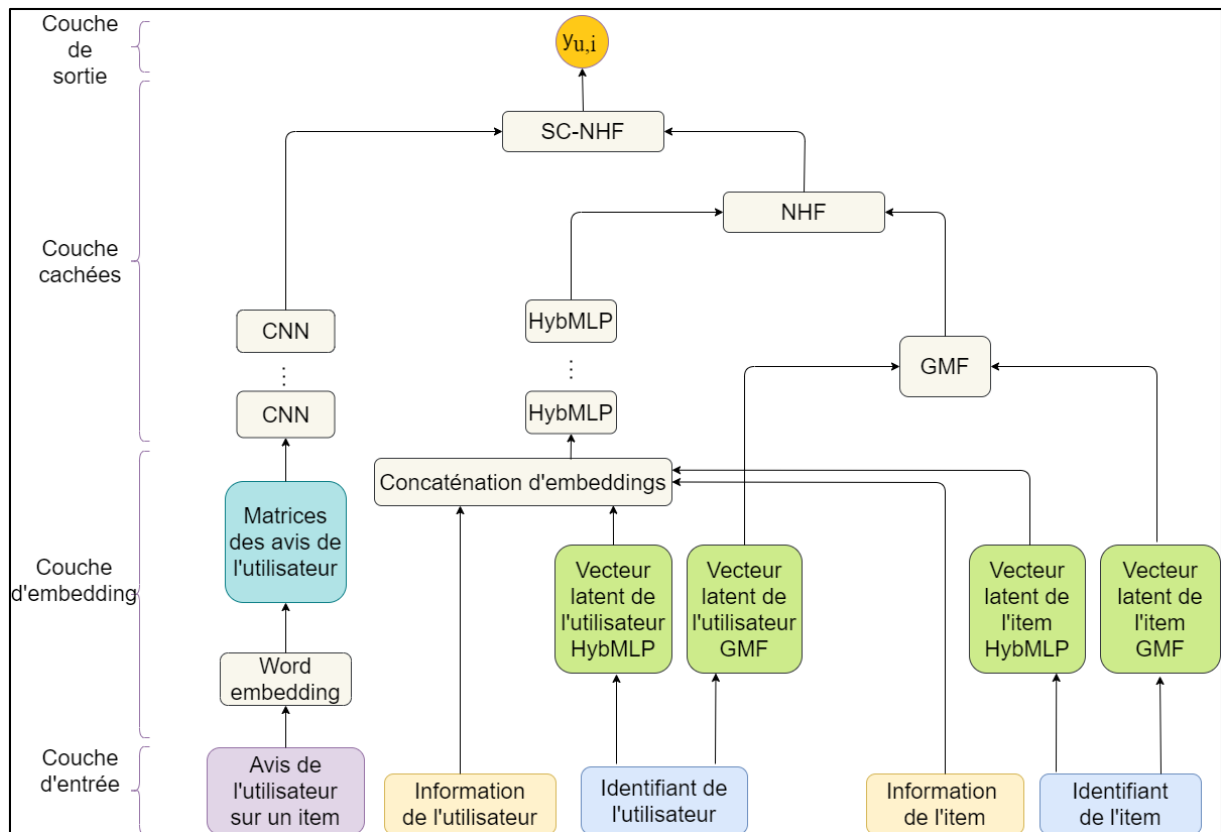


Figure 2.2: Architecture du SC-NHF

b. Description des couches du modèle

■ Couche d'entrée :

Dans cette section, il s'agit de présenter la première couche du modèle aussi appelée couche d'entrée. Celle-ci comporte les différentes informations et caractéristiques des utilisateurs et des *items*, ainsi que les différents avis d'un utilisateur sur les items avec lesquels il a préalablement interagi.

Trois entrées principales sont considérées :

- Les informations de l'utilisateur (exemple : occupation, âge, sexe, etc.) et *item* (exemple : genre, titre etc.).
- Les identifiants de l'utilisateur / *item*.
- L'avis de l'utilisateur donné aux *items*.

■ Couche d'embedding :

La deuxième couche du modèle (couche *embedding*), exploite les identifiants, avis des utilisateurs/*items* récoltés en entrée. Son objectif est dans un premier temps, d'obtenir les vecteurs de facteurs latents des identifiants de l'utilisateur/*item*, et dans un second temps, représenter sous forme matricielle les avis de l'utilisateur donnés à un certain item.

- **Traitement effectué sur l'identifiant de l'utilisateur/ item (les vecteurs latents de la couche GMF et HybMLP) :**

Cette partie a pour objectif la représentation vectorielle des différents identifiants de l'utilisateur (et parallèlement de l'*item*) de façon à reconnaître ceux qui ont un taux de similarité élevé.

Pour ce faire, le vecteur latent de l'utilisateur et celui de l'*item* des deux couches GMF et HybMLP sont obtenus en appliquant des traitements similaires à ceux du *Word embedding* (précédemment expliqué dans l'approche automatique du titre 1.4.2). Il suffit dans notre approche de créer à partir des identifiants des utilisateurs et *items* donnés en entrée deux vecteurs d'*embeddings* en multipliant l'index codé en *one hot* (technique expliquée dans le sous-titre b de 2.2.2) de chaque identifiant par les poids de la matrice d'*embedding* de sorte à ce que les deux vecteurs d'*embeddings* en sortie soient par la suite envoyés à la couche suivante (couche cachée) pour effectuer leurs concaténations dans le cas de HybMLP et multiplication dans le cas de GMF.

La transformation en *embeddings* est illustrée par la figure ci-dessous [25]:

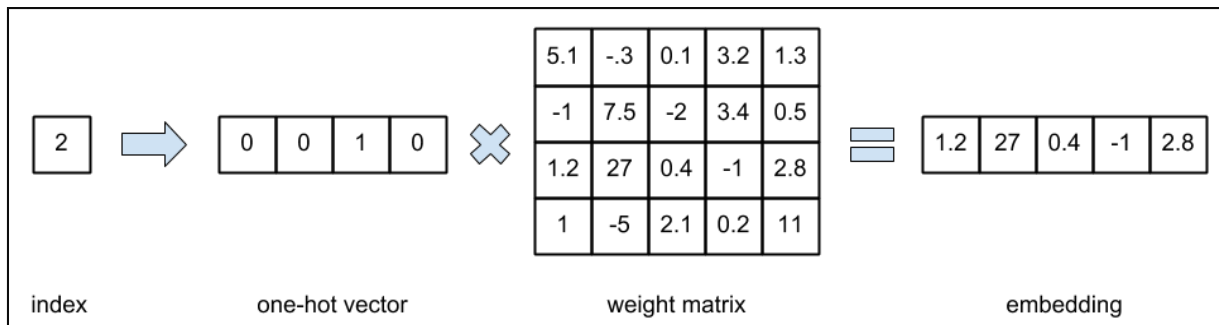


Figure 2.3 : Exemple de transformation en embedding

Il est à noter que les deux *embeddings* pour les différents couches GMF et HybMLP sont entraînés séparément afin de gagner en flexibilité.

▪ Couche cachée :

Dans la troisième couche de ce modèle, le but est d'effectuer les différents traitements nécessaires afin de prédire la note d'un utilisateur u sur un item i , et cela, tout en exploitant les différents résultats obtenus. Pour se faire, cette couche sera divisée en cinq sous-couches principales : une sous-couche représentant un modèle non-linéaire notée HybMLP, une autre représentant un modèle linéaire notée GMF, une représentant le modèle qui étudie les avis CNN et les deux dernières sous-couches pour la concaténation des résultats des dernières couches citées précédemment (HybMLP, GMF et CNN) notées NHF et SC-NHF.

- **Sous-Couche Perceptron Multicouches Hybride (Hybrid Multi Layer Perceptron - HybMLP) : Modèle non-linéaire :**

Dans cette partie, nous représentons le modèle non-linéaire obtenu à partir d'un réseau de neurones Hybride MLP (Hybrid Multi Layer Perceptron) qui aura pour objectif de donner une première prédiction de la note qu'attribuera l'utilisateur à l'item. Ainsi, la sortie de ce modèle sera sa dernière couche entraînée.

Pour cela, le modèle passe par deux étapes principales comme le montre la figure ci-dessous. Dans un premier lieu, les entrées seront combinées dans une couche de concaténation d'*embedding*. Ensuite les données obtenues seront passées par plusieurs couches cachées entraînées pour effectuer une prédiction sur les notes des utilisateurs.

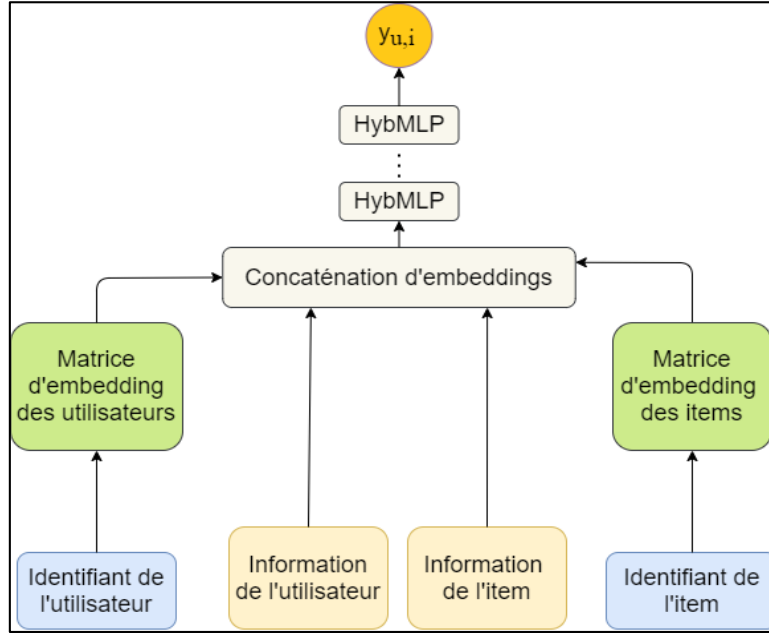


Figure 2.4 : Architecture du sous modèle HybMLP

Le modèle pourra alors être divisé en deux comme suit :

- **Couche de concaténation d'*embedding***

Cette couche possède comme entrée les vecteurs de facteurs latents et informations des utilisateurs (ou *items*) et les concatène dans l'objectif de générer la représentation en *embedding* des différents vecteurs obtenus à partir de la couche précédente (couche *embedding*).

- **Couche cachée du réseau de neurones**

Les couches cachées permettent de ressortir les informations pertinentes des données en entrées et introduisent dans notre cas une non-linéarité grâce à la fonction d'activation « *rectified linear unit* ReLU » (ReLU = max[0, x]) appliquée à chacune des couches du modèle.

Le modèle est représenté par la formule suivante [1]:

$$\phi^{MLP} = a_L(W_L^T(a_{L-1}(\dots a_2(W_2^T \begin{bmatrix} p_u^M \\ f_u \\ q_i^M \\ f_i \end{bmatrix} + b_2)\dots)) + b_L) \quad (2.1)$$

Avec :

- a_L : Fonction d'activation « ReLU » de la couche L.
- W_L^T : Poids de la couche L.
- p_u^M : Vecteur des facteurs latents de l'utilisateur u de HybMLP.

f_u : Vecteur d'informations de l'utilisateur u .
 q_i^M : Vecteur des facteurs latents de l'item i de HybMLP.
 f_i : Vecteur d'informations de l'item i .
 b_L : Bias de la couche L (fonctionne comme le seuil).

Le nombre de nœuds par couche est défini par une formule mathématique impliquant une valeur nommée *predictive factors* donnée en entrée. L'objectif de cette dernière est de garantir un nombre de nœuds par couche dans un ordre décroissant, en d'autres termes, plus il y a de couches et plus le nombre de nœuds diminue.

Ceci peut être donné par la formule mathématique suivante :

$$N_L = PF \times [2^{L-1}] \quad (2.2)$$

Où :

L : Numéro de la couche (initialisé au nombre de couches données en entrée et se décrémente d'une valeur après chaque couche).

PF : Valeur de *predictive factors* définie en entrée.

N_L : Nombre de neurones de la couche L .

- Sous-Couche Factorisation Matricielle Généralisée (Generalized Matrix Factorization - GMF) - Modèle linéaire :

La couche représentant la factorisation matricielle généralisée (GMF) est obtenue d'une extension de la factorisation matricielle classique (MF).

Elle exploite les deux vecteurs de facteurs latents des utilisateurs et *items* obtenus dans la deuxième couche de ce modèle (couche *embedding*) de sorte à ce que celle-ci produise un vecteur qui sera égal au produit du vecteur latent de l'utilisateur et de celui de l'item, comme le montre la figure ci-dessous.

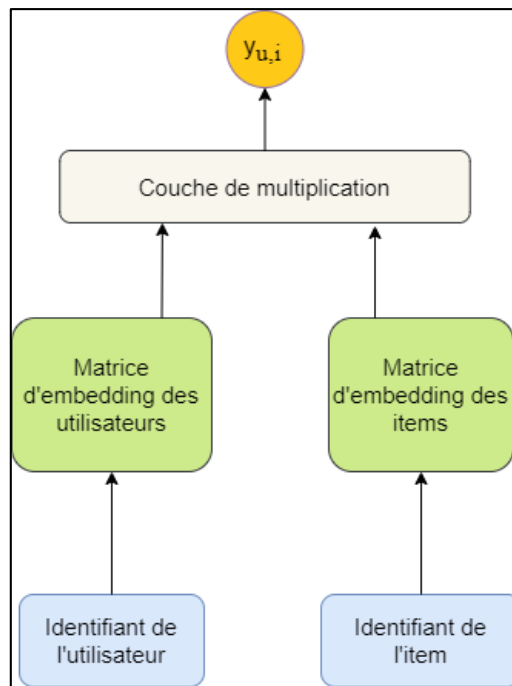


Figure 2.5 : Architecture du sous modèle GMF

Ainsi, la fonction de cette couche pourra être représentée selon la formule suivante [1] :

$$\phi^{GMF} = p_u^G \odot q_i^G \quad (2.3)$$

Où :

p_u^G : Vecteur des facteurs latents de l'utilisateur u de GMF.

q_i^G : Vecteur des facteurs latents de l'item i de GMF.

\odot : Opération de multiplication élément par élément (*element-wise product*).

Après avoir effectué ce traitement, le résultat de cette couche sera combiné avec le résultat de la couche HybMLP au niveau de la prochaine sous-couche à savoir NeuHybMF (aussi appelée NHF).

- Sous-Couche Réseau Neuronale Convolutif (Convolutional Neural Network CNN) :

Dans cette partie, nous étudions les avis des utilisateurs sur les *items* commentés dans le but d'extraire les caractéristiques des avis de ces utilisateurs en se référant aux notes qu'ils ont attribués.

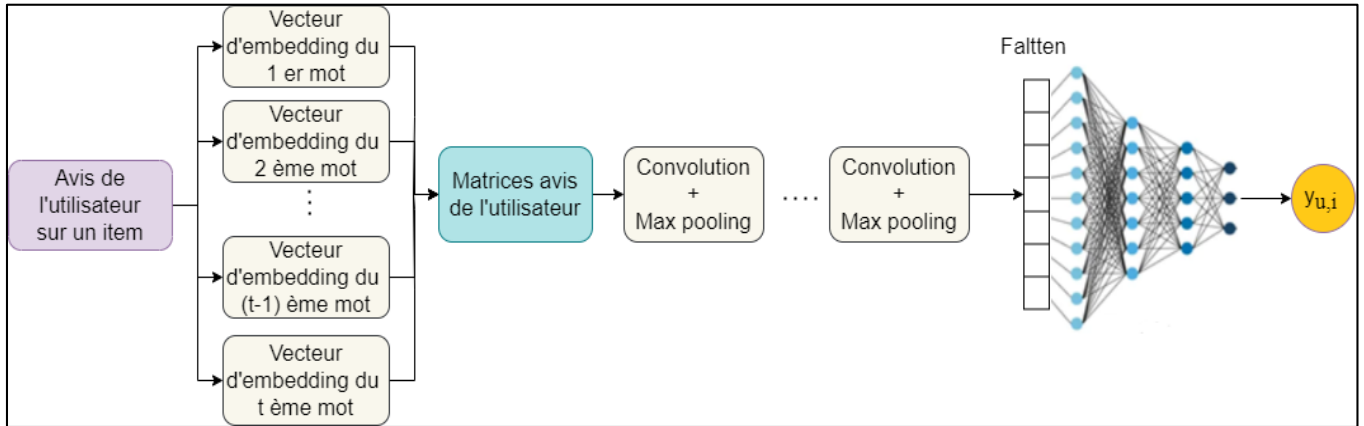


Figure 2.6 : Architecture du sous modèle CNN

Comme le montre l'architecture ci-dessus, nous commençons par le traitement des avis r_u (avis de l'utilisateur u) émis grâce à la technique du *word embedding* (expliquée dans l'approche automatique du titre 1.4.2). Ce dernier génère pour chaque avis $r_{u,i}$ une matrice W de vecteurs propre à chaque mot qui constitue l'avis noté, tel que, $W = [w_1, \dots, w_t] \in R^{t \times d_w}$ avec pour t le nombre de mots, d_w la dimension du vecteur et w_t le t -ième vecteur de la matrice (*word embedding vector*). La matrice W correspondante à l'avis $r_{u,i}$ sera par la suite analysée par un réseau de neurone convolutif (CNN) à k filtres $F = \{f_1, f_2, \dots, f_k\}$ où chaque filtre f_j est une fenêtre (matrice) avec une taille et des paramètres spécifiques.

Le résultat de cette convolution peut être interprété par la formule suivante [2]:

$$c_j = ReLU(W_{1:t} * f_j + b_j) \quad (2.4)$$

Avec :

c_j : Le résultat obtenu pour le j -ième filtre.

ReLU : *Rectified Linear Unit* est une fonction d'activation (non linéaire) dont la formule est la suivante $g(x) = \max[0, x]$.

$W_{1:t}$: Le vecteur obtenu pour chaque mot de l'avis (la matrice W contient de 1 à t mots)

$*$: opération de convolution.

f_j : Le j -ième filtre.

b_j : le biais ajouté au j -ième filtre.

Le résultat obtenu par les filtres est représenté par $C = \{c_1, c_2, \dots, c_k\}$. Nous effectuerons sur chaque c_j (avec j allant de 1 à k) un traitement de *max-pooling* afin d'extraire au final $O_{u,i} = [o_1, o_2, \dots, o_k]$ correspondant au traitements effectué sur l'avis $r_{u,i}$ de l'utilisateur u sur le item i .

La matrice $O_{u,i}$ passe par autant d'opérations de convolution et *max pooling* que nécessaire. Le résultat est par la suite empilé en un vecteur à une dimension grâce à la couche *flatten* (mis-à-plat) et envoyé la couche d'entrée du réseau de neurones qui aura pour sortie le résultat du modèle.

Ainsi, les données produites de cette dernière couche sont envoyées à la couche de concaténation SC-NHF pour obtenir le résultat final de la prédiction.

Il est à noter que le CNN de notre approche suit un apprentissage supervisé par les notes (étant la valeur à prédire (*target*) de notre système).

Nous décrivons dans ce qui suit, l'algorithme appliqué pour les transformations des avis en matrice d'*embedding* exploitable par le modèle.

Algorithm 1: Matrice word embedding des avis

Entrée : Texte pré-traité

Sortie : Matrice d'embedding du texte en entrée

Début

-Créer un dictionnaire de données (nommé wordIndex) pour chaque mot du texte en entrée. //Chaque mot correspond à une clé à laquelle on attribue un index (valeur numérique unique)

-Remplacer tous les mots du texte par leur index.

wordEmbFile : Fichier contenant tous des mots avec leurs vecteurs d'embedding pré-entraînés.

wordEmbDico : Dictionnaire de données initialisé à vide.

for ligne dans wordEmbFile **do**

 mot = le mot de la ligne;

 vecteur = tableau du vecteur que contient la ligne;

 wordEmbDico[mot] = vecteur;

end

EmbMatrice : Matrice initialisée à zero.

for mot, index dans wordIndex **do**

 vecteur = chercher mot dans wordEmbDico;

if vecteur \neq null **then**

 EmbMatrice[index] = wordEmbDico[vecteur];

end

end

return EmbMatrice

Fin

- **Sous-Couches de concaténation « Neural Hybrid Matrix Factorization – NeuHybMF » et « Convolutional Neural Hybrid Matrix Factorization – ConvNeuHybMF » :**

L'objectif de cette couche est la concaténation des couches des modèles précédents (GMF, HybMLP et CNN) puis l'envoi du résultat de cette dernière à la couche de sortie.

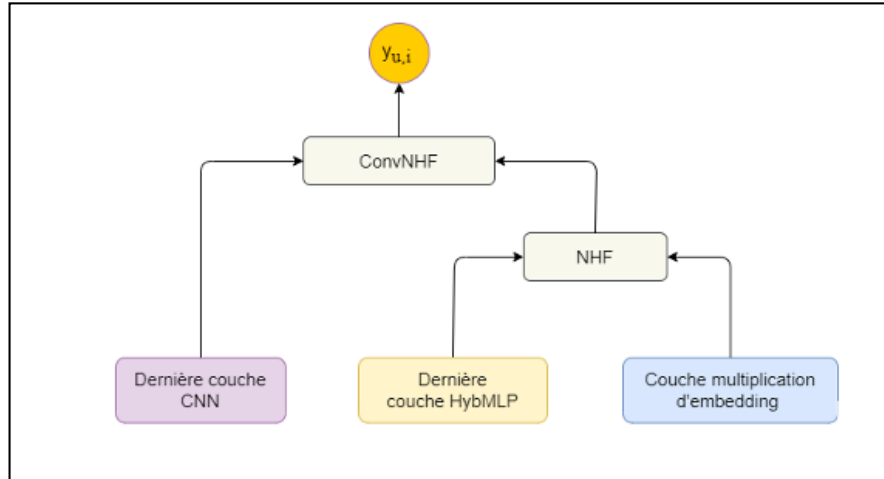


Figure 2.7 : Couches de concaténation

Étant donné que pour concaténer trois couches, il suffit de passer par une concaténation intermédiaire. Nous allons dans un premier temps concaténer les deux couches résultantes des modèles HybMLP et GMF que nous appellerons « NeuHybMF » (qu'on note NHF). Puis, afin d'exploiter les avis de l'utilisateur, nous concaténons le résultat de cette dernière (NeuHybMF) avec le résultat de la couche CNN, pour enfin avoir comme sortie une couche de concaténation finale appelée « SC-NeuHybMF » (qu'on note SC-NHF).

Il est à souligner que le résultat de CNN est concaténé avec celui de GMF et HybMLP afin de garantir une meilleure précision lors de la prédiction finale, à savoir celle donnée dans la couche de sortie.

▪ Couche de sortie :

Dans cette partie nous présentons la quatrième et dernière couche de ce me modèle aussi appelée couche de sortie. Celle-ci n'est autre que le résultat de la couche SC-NHF qui prend en entrée la concaténation des deux couches résultantes des modèles NHF et CNN, cela, afin d'obtenir la prédiction de la note de l'utilisateur u sur l'item i suivant la loi de probabilité définie par la fonction d'activation softmax ($s(x_i) = e^{x_i} / \sum_{j=1}^n e^{x_j}$).

2.3 Processus de recommandation

Il est possible de définir le problème de la recommandation de deux manières différentes, la génération de listes d'items ordonnées ou la prédiction de notes.

Dans le cas de l'ordonnancement, la recommandation est une liste d'items classée selon l'ordre d'intérêt de l'utilisateur. Seul les K items les plus pertinents sont renvoyés vers celui-ci [26]. Dans le second cas, la qualité est mesurée selon la différence entre la note prédite et la note réellement attribuée. Ce dernier cas est le paradigme utilisé aussi bien par le moteur GroupLens [27] mais également dans le cadre du challenge Netflix [28] et c'est aussi l'approche que nous avons retenue.

Afin de tester l'efficacité de cette approche, nous prenons un utilisateur u ayant recommandé n items i . Nous sélectionnons ensuite, une partie p_1 de l'ensemble de ces *items* recommandés par l'utilisateur u pour effectuer l'entraînement du modèle, et une autre partie p_2 afin de le tester. Les résultats des tests permettront d'évaluer la qualité de la prédiction des notes $r_{u,i}$ (notes attribuées par u sur chaque *item* de p_2).

Les étapes par lesquelles passe notre modèle pour effectuer cette prédiction sont décrites dans l'algorithme ci-dessous.

Algorithme 2: Algorithme de prédiction

Entrée : Identifiants et informations des utilisateurs/items, avis des utilisateurs.

Sortie : Note prédite.

Début

- (1) Traiter les identifiants des utilisateurs et items puis les transformer en vecteurs de facteurs latents pour les modèles GMF et HybMLP.
- (2) Traiter les avis des utilisateurs pour obtenir une représentation matricielle de ces avis.
- (3) Entraîner le modèle GMF avec les vecteurs de facteurs latents GMF.
- (4) Entraîner le modèle HybMLP avec les vecteurs de facteurs latents HybMLP et informations des utilisateurs et items.
- (5) Entraîner le modèle CNN avec la matrice de l'ensemble des avis des utilisateurs.
- (6) Concaténer le résultat des dernières couches du modèle GMF et HybMLP dans la couche NHF.
- (7) Concaténer la couche NHF avec le résultat de la couche CNN et obtenir une prédiction finale de la note.

return note

Fin

2.4 Entraînement

2.4.1 Gestion du jeu de données (*Dataset*)

a. Exploitation des données

Les données explicites représentent les données que l'utilisateur fournit explicitement au système afin d'indiquer son intérêt pour un *item* précis. Nous pouvons prendre comme exemple un système de notation allant de 1 à 5 ou encore les avis (commentaires) émis par l'utilisateur sur un certain *item*. Les données implicites sont représentées par une observation du comportement de l'utilisateur, comme par exemple le clique d'une souris, le temps de

consultation passé etc. Les données explicites ont donc souvent l'avantage par rapport aux données implicites d'exprimer de manière plus précise les préférences de l'utilisateur face aux *items*. De ce fait, notre choix s'est porté sur le traitement des données dites explicites de nos *datasets*.

Le prétraitement général appliqué aux données se fait comme expliqué dans la section 2.2.2 de ce même chapitre, néanmoins chaque *dataset* mets à disposition des données de catégories et types différents, chaque type est traité de sorte à pouvoir être exploitée correctement par le modèle. Nous prenons à titre d'exemple le traitement effectué sur les données de « Yelp *dataset* challenge 2013²⁸ », qui offre la possibilité d'étudier les notes et avis des utilisateurs dont le traitement a été expliqué dans la section 2.2.2 mais aussi, des informations de type booléen ou catégorique qui seront respectivement transformés en code binaire et labélisés.

b. Type d'évaluation et répartition des données du dataset

Nous entraînons notre model selon les deux types d'évaluations suivants :

- **Étude des données sur *dataset* avec utilisateurs connus :** L'approche classique qui consiste à répartir (*split*) le *dataset* de manière aléatoire entre les données d'entraînement et de test.
- **Étude des données du *dataset* avec utilisateurs inconnus :** A travers cette approche nous voulons tester le comportement de notre modèle face au problème du démarrage à froid et vérifier ainsi si l'hybridation du filtrage cognitif et filtrage collaboratif appliquée avec l'exploitation des avis des utilisateurs réussis à pallier à ce dernier. Pour ce faire nous répartissons les données de sorte à ce que les utilisateurs avec lesquels a été entraîné le modèle et ceux avec lesquels il est testé soit différents, en d'autres termes tester le modèle sur des utilisateurs inconnus et voir ses résultats.

Le jeu de données a été réparti comme suit :

- 80% pour l'entraînement.
- 20% pour le test.

2.4.2 Entraînement

a. Fonction de coût (*loss function*)

Dans le but d'améliorer la performance de notre modèle, nous exploitons la fonction de coût, aussi appelée fonction-objectif, MAE (*Mean Absolute Error*) ou erreur absolue moyenne qui calcule la différence entre la prédiction du modèle et la réelle valeur et dont la formule est donnée ci-dessous [29] :

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |e_i| \quad (2.5)$$

²⁸ <https://www.yelp.com/dataset/>

Où :

$e_i = y_i - x_i$ où y_i représente la prédiction et x_i la vraie valeur.

Par définition, plus la valeur de la fonction objectif est petite, plus nos modèles seront précis. Notre objectif principal est d'arriver à des paramètres qui minimiseraient cette fonction.

De ce fait, pour minimiser au mieux notre fonction de coût, nous utilisons ce qu'on appelle un algorithme d'optimisation expliqué dans ce qui suit.

b. Algorithme d'optimisation (*optimizer*)

Les algorithmes d'optimisation représentent des méthodes utilisées afin de modifier les paramètres d'un réseau de neurones (poids, taux d'apprentissage, etc.), ceci, dans le but de réduire sa fonction coût expliquée précédemment. Nous utiliserons dans notre modèle l'un des algorithmes d'optimisation les plus connus, l'algorithme Adam (*Adaptive Moment Estimation*) [1] qui représente une méthode d'optimisation stochastique (aléatoire).

2.5 Conclusion

Dans ce chapitre, nous avons présenté notre approche de recommandation, qui enrichi l'approche hybride neuronale (NHF) Kerouba et Zeghoud [1] par l'étude des avis des utilisateurs sur les items. Notre objectif étant de montrer l'apport de l'analyse de sentiments aux algorithmes de recommandation.

Afin d'expérimenter cet apport, nous allons présenter dans le chapitre suivant les aspects techniques liés à l'implémentation et à l'évaluation de notre système de recommandation.

Chapitre 3 :

Implémentation et expérimentations

3.1 Introduction

Dans ce chapitre, nous décrivons dans un premier lieu les aspects techniques liés à l'implémentation de notre système de recommandation, qui sera basé sur le modèle SC-NHF. Nous présentons ensuite les différents résultats des expérimentations et évaluations effectuées.

3.2 Environnement de développement

Cette section décrit les outils et ressources utilisés dans notre implémentation.

3.2.1 Environnement logiciel

a. Langage de programmation

Pour la réalisation de notre système, nous avons choisi Python²⁹ (version 3.6.9) qui est l'un langage de programmation les plus utilisé pour l'apprentissage automatique et l'apprentissage profond, cela, dû à sa polyvalence, la simplicité de ses syntaxes et aux nombreuses bibliothèques qu'il présente.

b. Bibliothèques

Ci-dessous les bibliothèques qui nous ont aidés à la réalisation des modèles et au traitement des données.

- Pandas ³⁰ permet d'exploiter et créer différents formats de fichiers (csv, json, xls ...). Elle permet également de filtrer, fragmenter et combiner les données grâce aux méthodes prédéfinies de ses classes.
- Numpy ³¹ facilite le calcul vectoriel en offrant la possibilité de manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
- Keras ³² avec Tensorflow ³³ comme *backend* permettent d'appliquer les notions d'apprentissage automatique comme la création et l'entraînement des modèles de différentes architectures de réseaux de neurones (légers ou profonds).

²⁹ <https://www.python.org/>

³⁰ <https://pandas.pydata.org/>

³¹ <https://numpy.org/>

- Gensim utile pour traitement du langage naturel (NLP), elle permet de modéliser des thèmes, indexer des documents et opérer une recherche de similarité d'informations dans de larges corpus.
- Nltk ³⁴ est souvent utilisé pour traitement du langage naturel (NLP), elle permet d'effectuer divers traitements sur les données textuelles grâce à ses différents modules (*tokenization, stemming, parsing...*).
- Matplotlib ³⁵ facilite la visualisation statique, animée et interactive des graphiques en Python.
- Surprise ³⁶ donne la possibilité de créer, analyser et comparer des systèmes de recommandation prédéfinis qui travaillent explicitement avec des notes.

c. Google Colaboratory

Colaboratory³⁷, ou plus communément appelé « Colab », est un produit de Google Research qui permet de partager des *notebooks* (projets) entre plusieurs utilisateurs, il est possible à partir de ces derniers d'écrire et d'exécuter du code Python à partir d'un navigateur sans avoir à télécharger ou installer quelconque logiciel au préalable. C'est un environnement particulièrement adapté au *machine learning* et à l'analyse de données. En termes plus techniques, Colab est un service hébergé de *notebooks* Jupyter qui ne nécessite aucune configuration et permet d'accéder gratuitement à des ressources informatiques, dont des GPU (Graphic Processing Unit). L'utilisation de ces derniers est préférable pour l'entraînement des modèles compte tenu de sa rapidité comparée au CPU.

3.2.2 Description de notre système de recommandation

Afin faciliter la présentation de notre système de recommandation, nous avons développé une application Java, en utilisant le *framework* JavaFX ainsi que son outil interactif SceneBuilder pour la création de la GUI (l'interface graphique).

Ci-dessous la fenêtre principale de l'application qui comporte une représentation des deux modèles développés : SC-NHF et SC-NCF (une variante de notre modèle qui considère seulement le FC avec l'analyse de sentiment).

³² <https://keras.io/>

³³ <https://www.tensorflow.org/>

³⁴ <https://www.nltk.org/>

³⁵ <https://matplotlib.org/>

³⁶ <http://surpriselib.com/>

³⁷ <https://colab.research.google.com/>

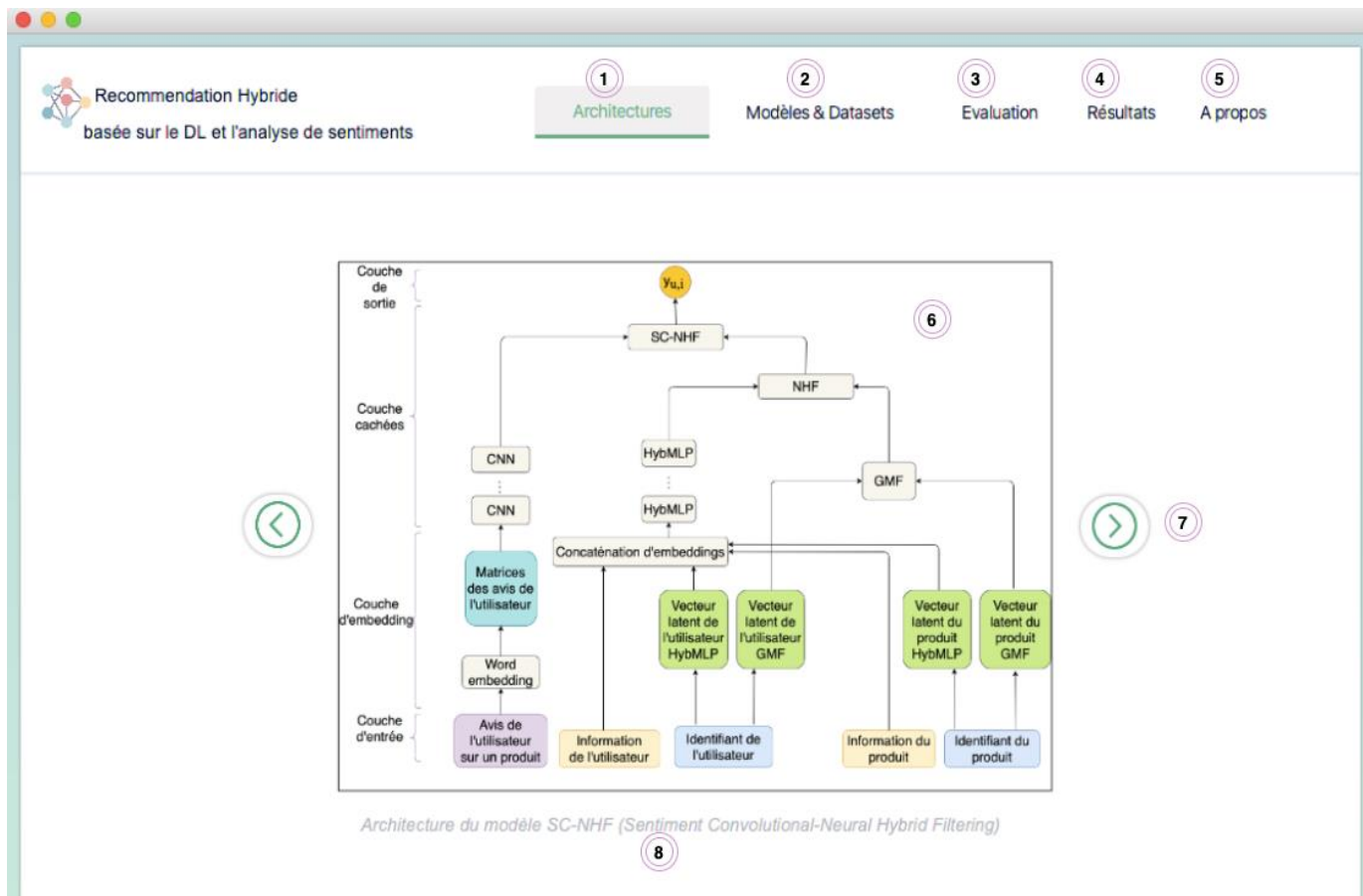


Figure 3.1 : Interface de l'application - Fenêtre principale

Description des fonctionnalités décrites dans la figure 3.1 :

- 1 : Présentation de l'architecture des modèles SC-NHF et SC-NCF.
- 2 : Sélection du modèle, type d'évaluation et *datasets* à évaluer.
- 3 : Sélection des options d'évaluation et l'affichage du score obtenu.
- 4 : Affichage des résultats de tests sur les hyper-paramètres.
- 5 : Informations concernant l'application.
- 6 : Architecture d'un des modèles affichés.
- 7 : Affichage du modèle suivant.
- 8 : Nom du modèle dont l'architecture est affichée.

Afin de pouvoir afficher le résultat des scores il faut tout d'abord commencer par définir le modèle, le type d'évaluation ainsi que le *dataset* souhaité.

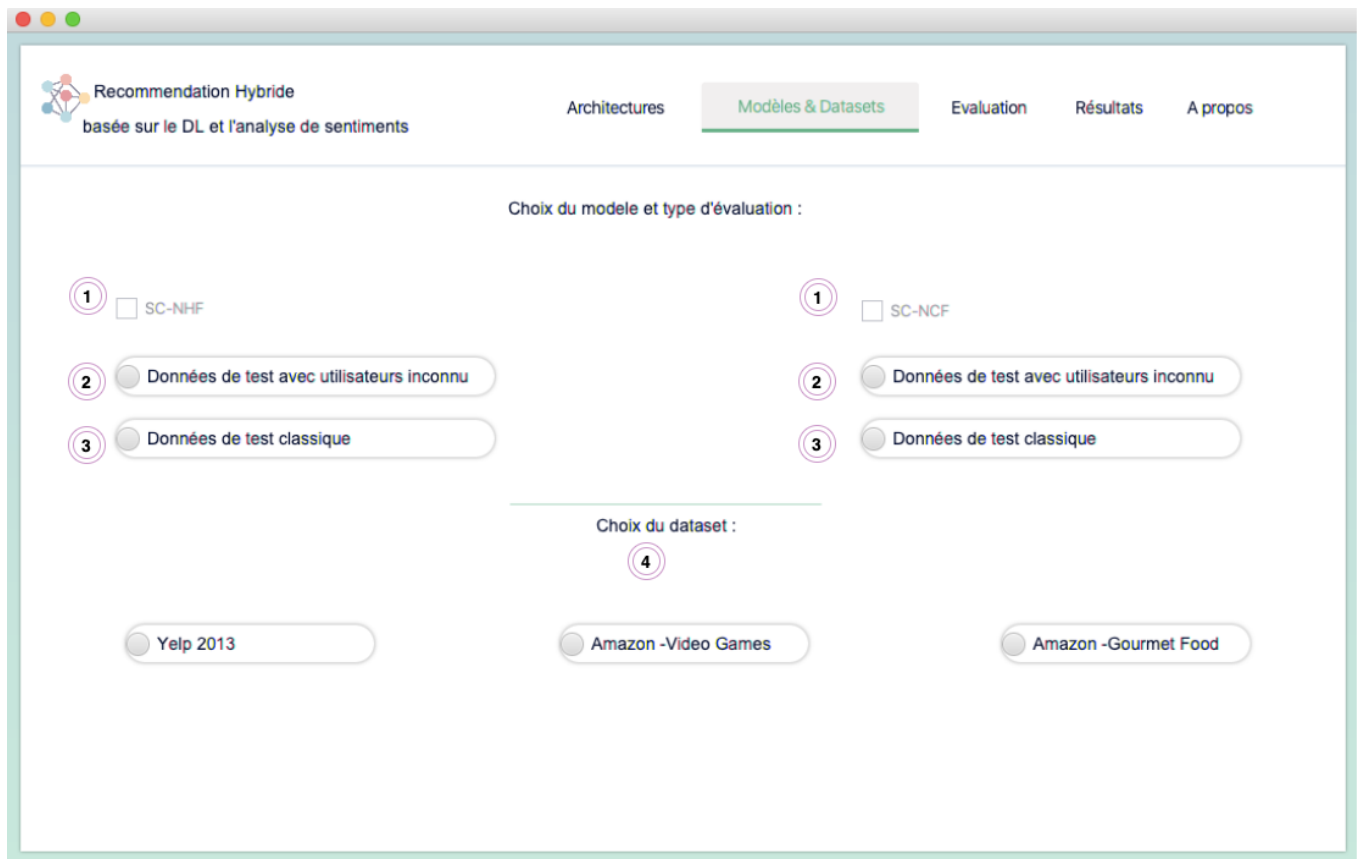


Figure 3.2 : Interface de l'application - Sélection du modèle et du dataset

Description des fonctionnalités décrites dans la figure 3.2 :

- 1** : Sélection du modèle à évaluer.
- 2** : Sélection du type d'évaluation qui détermine le découpage du *dataset* selon le critère des utilisateurs inconnus (voir sous-titre b de 2.4.1).
- 3** : Sélection du type d'évaluation qui détermine un découpage classique du *dataset*.
- 4** : Sélection du jeu de données.

Une fois que les informations relatives au modèle et jeux de données ont été sélectionnés, vient le choix des informations propre à l'entraînement du système. Il est possible, par exemple, de faire varier le nombre de couches du sous modèle MLP de SC-NHF (ou de SC-NCF) et voir selon chaque couche les résultats obtenus.

Recommandation Hybride
basée sur le DL et l'analyse de sentiments

Architectures Modèles & Datasets **Evaluation** Résultats A propos

Options d'évaluation :

1

Optimizer :	Adam	Nombre d'épochs :	20
Learning rate :	1e-8	Batch size :	256
Métrique :	RMSE	Patience :	2
Taille de l'embedding GMF :	8	Dropout :	▼
Taille de l'embedding HybMLP :	8	Nombre de couches HybMLP :	▼
Predictive factors :	16	Score de l'évaluation:	2

3 Evaluer

Figure 3.3 : Interface de l'application - Evaluation du modèle

Description des fonctionnalités décrites dans la figure 3.3 :

- 1 : Sélection des options d'évaluation comme la valeur du *dropout*³⁸souhaitée.
- 2 : Affichage du score obtenu.
- 3 : Evaluation selon les paramètres donnés.

3.3 Métrique d'évaluation

Afin d'évaluer la qualité de nos prédictions, notre choix s'est porté sur l'Erreur Quadratique Moyenne aussi appelée **RMSE** (Root Mean Squared Error) étant également adoptée par l'article de Ye et al [30]. Elle est donnée par la formule suivante :

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (\hat{r}_j - r_j)^2} \quad (3.1)$$

Avec : \hat{r}_j : le résultat prédit.

³⁸ **Dropout** : Aussi appelée « régularisation par abandon » est une méthode de régularisation utilisée dans l'apprentissage profond qui consiste à sélectionner de manière aléatoire un nombre défini de neurones pour les ignorer ou « abandonner » afin d'éviter un sur-apprentissage.

r_j : le résultat attendu.

n : le nombre total d'évaluations.

Cette formule consiste à calculer la différence entre les résultats prédits (de l'échantillon de test) et les résultats attendus. De ce fait, plus la valeur de cette mesure diminue plus notre modèle est efficace de par ses prédictions.

3.4 Modèles de comparaison

Pour évaluer notre système et observer l'apport de l'hybridation combinée à l'analyse de sentiments, nous avons comparé nos résultats avec sept autres modèles chacun traitant une approche spécifique. La première approche étant celle basée sentiments à travers les modèles **LSTM** [31] et **CNN** [32], la seconde est basée contenu à travers le sous modèle **NHF** de notre architecture (qui est un modèle adapté du système de Kerouba et Zeghoud [1]), et enfin les modèles suivant l'approche collaborative à travers le sous modèle **NCF** de notre architecture (qui est un modèle adapté de l'article de He et al [11]) ainsi que les modèles **SVD** [33], **KNN** [34] et **Co-Clustering** [35] proposés par la bibliothèque Surprise (décrite dans 3.2.1).

Il est à noter que les modèles NHF et NCF ont été adaptés en considérant les notes des utilisateurs au lieu de leurs interactions.

3.5 Évaluations des modèles

3.5.1 Description des bases de données

Les données exploitées pour l'entraînement et l'évaluation des modèles proviennent des trois *datasets* suivants :

- **Yelp** : Cette base de données provient du Yelp Dataset Challenge³⁹ (2013) et met à disposition les notes, avis ainsi que les informations relatives aux utilisateurs et restaurants (*items*).
- **Video games & Gourmet food** : Ils représentent deux catégories différentes de *datasets* proposés par Amazon⁴⁰ avec un filtrage prédéfini et contiennent respectivement des données relatives aux commerce des jeux vidéo et produits alimentaires. Le concept de filtrage proposé par cette enseigne garanti une base de K-cores qui veut dire K évaluations par utilisateur et article. Dans notre projet nous utilisons une base de 5-cores.

Nous appliquons sur ces données les différents traitements précédemment cités dans le chapitre 2 afin d'obtenir des vecteurs exploitables à savoir : les transformations en vecteurs numériques, traitements sur le texte (*word embedding*), etc.

³⁹ <https://www.yelp.com/dataset/challenge>

⁴⁰ <http://jmcauley.ucsd.edu/data/amazon/>

Les statistiques relatives aux jeux de données utilisés sont présentées dans le tableau ci-dessous :

<i>Datasets</i>	<i>#utilisateurs</i>	<i>#items</i>	<i>#notes/avis</i>	<i>Densité %</i>	<i>#longueur avis</i>
Yelp	5436	4733	118709	0.46	3738
Video games	13927	10644	131248	0.09	3000
Gourmet food	14681	8713	151 230	0.11	2840

Tableau 3.1 : Statistiques relatives aux jeux de données

Remarque : Pour chaque note attribuée par un utilisateur sur un *item* donné il existe un avis (commentaire) décrivant cette interaction.

La densité d'un *dataset* peut être calculée comme suit :

$$D = \frac{\text{nombre d'évaluations contenues dans le dataset}}{\text{nombre d'utilisateurs} \times \text{nombre d'items}} \quad (3.2)$$

3.5.2 Évaluations préliminaires

Dans cette section, nous allons présenter les résultats obtenus par le modèle SC-NHF à travers une série d'expérimentations dans le but déterminer ses paramètres optimaux. Ainsi, nous allons observer l'impact qu'a eu la variation des hyper paramètres⁴¹ de même que l'utilisation du *word embedding* pré-entraîné sur les performances du système.

Le tableau 3.2 illustre les résultats de la métrique RMSE sur les différentes versions du modèle (testé sur le *dataset* Yelp 2013). Ces résultats sont interprétés par ce qui suit :

- La variation du nombre d'epochs et du taux d'apprentissage (*Learning Rate - LR*) :

Il est important de préciser dans un premier lieu que le taux d'apprentissage (LR) fait partie des paramètres de l'*optimizer* Adam. Ce paramètre permet de contrôler la vitesse à laquelle les pondérations sont ajustées. Le meilleur taux d'apprentissage dépend de l'architecture du modèle à optimiser ainsi que les *dataset* utilisé.

Du tableau 3.2, nous pouvons observer qu'un nombre trop réduit d'epochs et une valeur trop élevée du *LR* affecte négativement les résultats du modèle, ne permettant pas à ce dernier de s'entraîner assez et de capturer les caractéristiques nécessaires à la prédiction.

Dans notre modèle, il est clair qu'un taux d'apprentissage de 10^{-8} combiné a un nombre d'epochs égal à 20 avec un arrêt précoce (ou *EarlyStopping*⁴²) effectuent des progrès notables comparé aux résultats du premier modèle figurant sur le tableau.

⁴¹ **Hyper-paramètres** : les hyper paramètres représentent les paramètres relatifs aux modèles, dont la valeur est fixée avant l'entraînement de celui-ci. La variation de ces valeurs influence le processus d'entraînement.

⁴² **EarlyStopping** : une méthode de régularisation qui permet d'arrêter l'apprentissage avant d'atteindre le nombre total d'epochs afin d'éviter le sur-apprentissage du modèle.

- *L'utilisation d'un word embedding pré-entraîné :*

Après avoir ajusté le nombre d'*epochs* et le taux d'apprentissage du modèle SC-NHF, nous expérimentons l'utilisation d'un *word embedding* pré-entraîné de Google⁴³. Comme le montre le tableau 3.2, cet apport satisfait la convergence du 3ème modèle(SC-NHF avec un *word embedding* pré-entraîné) et donne un résultat d'RMSE plus optimal, soit, 0.4021 au lieu de 0.4186 (résultat donné par le 2ème modèle SC-NHF sans le *word embedding* pré-entraîné, présent sur le tableau).

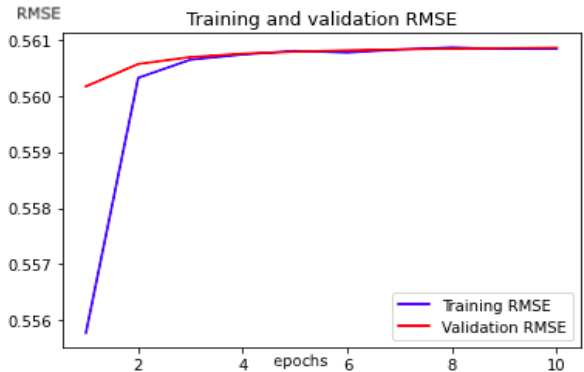
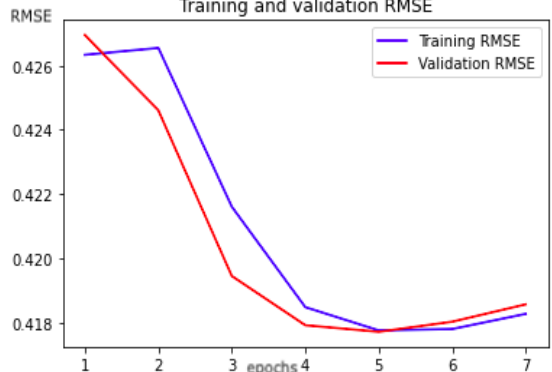
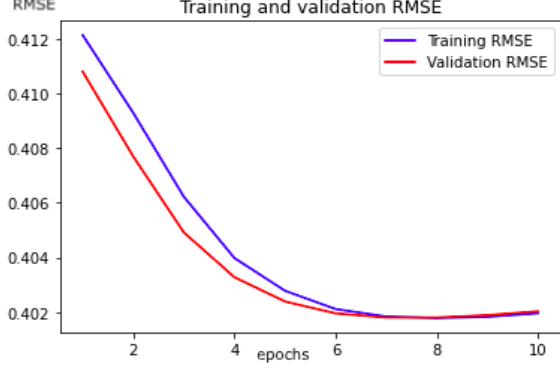
<i>Variantes de l'architecture du modèle SC-NHF</i>	<i>Hyper-Paramètres</i>	<i>RMSE</i>	<i>Graphes d'évaluation par nombre d'epochs</i>
SC-NHF	$LR=10^{-3}$ $Epochs=10$	0.4936	
SC-NHF sans le Word embedding pré-entraîné	$LR=10^{-8}$ $Epochs=20$ (avec <i>EarlyStopping</i>)	0.4186	
SC-NHF	$LR=10^{-8}$ $Epochs=20$ (avec <i>EarlyStopping</i>)	0.4021	

Tableau 3.2 : Tableau comparatif des variantes du modèle SC-NHF

⁴³ <https://code.google.com/archive/p/word2vec/>

Dans cette deuxième partie, nous étudions comme illustré par les figures 3.4 et 3.5, les résultats du modèle selon la variation du *Dropout* ainsi que du nombre de couches de MLP. Pour chaque variation, un modèle est créé et son score est affiché par un point du graphe.

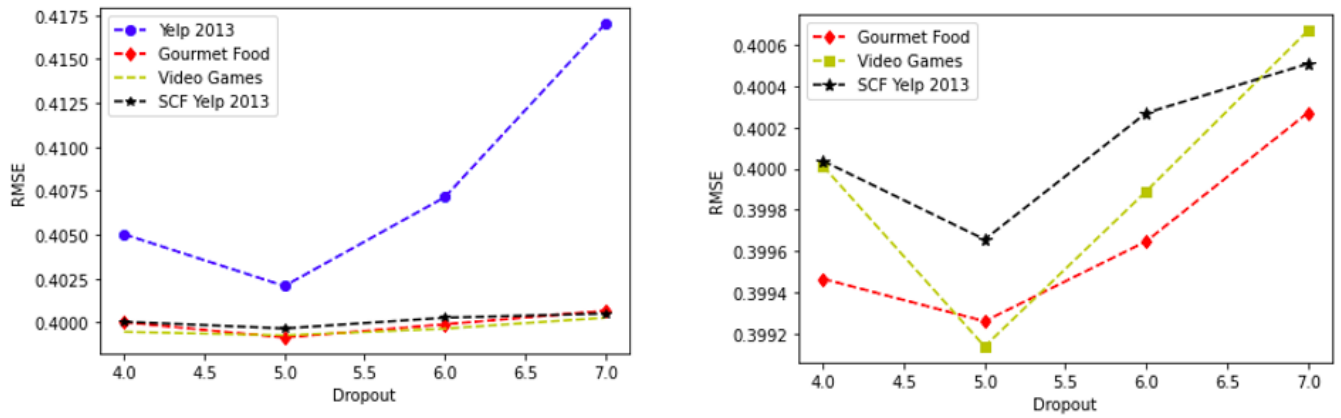


Figure 3.4 : Figure représentant l'évaluation du modèle SC-NHF sur les différents datasets en fonction de l'hyper-paramètre Dropout

Le tableau 3.3 présente les résultats obtenus sur les différents *datasets* avec une variation de la valeur du *dropout*. Nous notons que les datasets Video games et Gourmet food ne peuvent pas être utilisées avec le modèle hybride vu qu'ils n'incluent pas les informations relatives aux utilisateurs.

		Dataset											
		Yelp				Video games				Gourmet food			
Dropout ($\times 10^{-1}$)		4	5	6	7	4	5	6	7	4	5	6	7
Modèles	SC-NCF	0.400 0	0.39 97	0.40 02	0.40 05	0.40 00	0.39 91	0.39 99	0.40 07	0.39 95	0.39 92	0.39 96	0.40 02
	SC-NHF	0.405 0	0.40 20	0.40 71	0.41 70	/	/	/	/	/	/	/	/

Tableau 3.3 : Tableau représentant l'évaluation de nos approches selon la métrique RMSE sur les différents datasets en fonction de la variation du Dropout

Les résultats d'expérimentation illustrés dans la figure 3.4 démontrent que le modèle SC-NHF avec un *dropout* fixé à 0.5 parvient à réaliser les meilleurs résultats sur tous les *datasets*.

Comme illustré par cette même figure, la métrique RMSE commence d'abord par se décrémenter puis s'incrémente parallèlement avec la valeur du *dropout*. Le meilleur résultat est atteint avec un *dropout* de 0.5. Cette valeur est donc sélectionnée pour l'entraînement du modèle final.

Ces résultats démontrent, l'apport de cet hyper-paramètre et son impact sur l'atténuation du sur-apprentissage du modèle.

La figure 3.5 ci-dessous illustre les valeurs de la métrique RMSE selon la variation du nombre de couches cachées du modèle MLP.

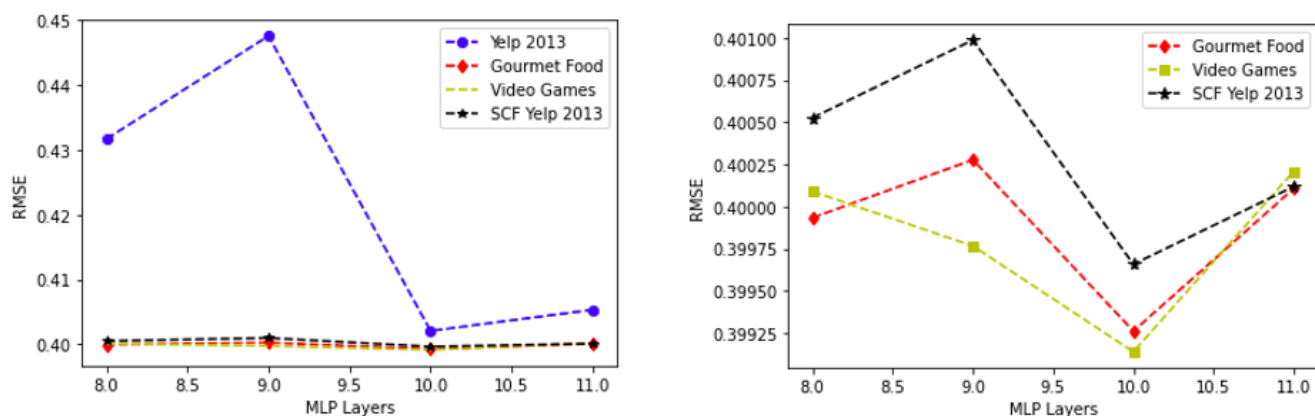


Figure 3.5 : Figure représentant l'évaluation du modèle SC-NHF sur les différents datasets selon le nombre de couches MLP.

Le tableau 3.4 présente les résultats obtenus sur les différents *datasets* avec une variation du nombre de couches de MLP.

		Dataset											
		Yelp				Video games				Gourmet food			
Nombre de couches MLP		8	9	10	11	8	9	10	11	8	9	10	11
Modèles	SC-NCF	0.4005	0.4009	0.3997	0.4001	0.4000	0.3997	0.3991	0.4002	0.3999	0.4002	0.3992	0.4001
	SC-NHF	0.4316	0.4475	0.4020	0.4053	/	/	/	/	/	/	/	/

Tableau 3.4 : Tableau représentant l'évaluation de nos approches selon la métrique RMSE sur les différents datasets en fonction de la variation du nombre de couches de MLP.

Nous pouvons observer que le nombre optimal de couches cachées du modèle MLP est égale à 10, avec une valeur minimale de RMSE. Nous pouvons également remarquer qu'un plus grand nombre de couches augmente la valeur de cette mesure. Cela peut être expliqué par un éventuel sur-apprentissage (*overfitting*) du modèle dû au nombre trop élevé de ces couches cachées. Parallèlement, un nombre de couches trop réduit, conduirait le modèle à un sous-apprentissage (*underfitting*).

3.5.3 Impact de l'analyse de sentiments sur la recommandation collaborative

Afin de tester l'apport de l'analyse de sentiments sur le filtrage collaboratif, nous avons comparé les résultats des évaluations de notre modèle SC-NCF avec les différents modèles de comparaisons précédemment cités (voir section 3.4).

		<i>Datasets</i>		
		<i>Yelp</i>	<i>Video games</i>	<i>Gourmet food</i>
<i>Modèles</i>	SC-NCF	0.3997	0.3991	0.3993
	NCF	0.4936	0.4000	0.4000
	SVD	1.0136	1.0891	0.9946
	KNN	1.0765	1.2843	1.1199
	Co-Clustering	1.0809	1.1738	1.0680
	LSTM	0.4947	0.4368	0.4096
	CNN	0.4947	0.4368	0.4097

Tableau 3.5 : Comparaison des résultats du filtrage collaboratif et basé sentiments

Comme illustré par le tableau 3.5, l'approche SC-NCF réussit à surpasser les modèles de comparaisons sur les trois *datasets*, avec pour meilleur résultat, une valeur de RMSE égale à 0.3991 sur Video games. Nous retrouvons en deuxième position le modèle NCF qui, de par son filtrage collaboratif combinant les modèles GMF et MLP, réussit à donner des résultats très respectables. Puis, sont classés juste après, les modèles basés exclusivement sur l'analyse de sentiments, à savoir : LSTM et CNN. Enfin on retrouve les modèles de filtrage collaboratif classique représentés par SVD, KNN et Co-Clustering.

3.5.4 Impact de l'analyse de sentiments sur la recommandation hybride

a. Évaluation avec utilisateurs connus (classique)

Nous présentons dans cette partie l'évaluation sur le *dataset* Yelp 2013 du modèle hybride SC-NHF. L'hybridation réside dans le fait que le modèle SC-NHF étudie en plus les informations relatives aux utilisateurs et aux *items*.

<i>Modèle</i>	<i>RMSE</i>
SC-NHF	0.4021
NHF	0.4118
SC-NCF	0.3997
NCF	0.4936
SVD	1.0136
KNN	1.0765
Co-Clustering	1.0809
LSTM	0.4947
CNN	0.4947

Tableau 3.6: Comparaison des modèles avec des utilisateurs connus

Bien que le modèle SC-NHF ait été légèrement surpassé par sa variante SC-NCF, il donne néanmoins des résultats satisfaisants se positionnant en second lieu du classement. Nous pouvons déduire de ceci que l’approche collaborative est plus performante que l’approche hybride pour la prédiction des notes sur le *dataset Yelp 2013*.

b. Évaluation avec utilisateurs inconnus

Il est important d’observer l’apport de l’hybridation en considérant le problème de démarrage à froid. Pour cela, nous présenterons dans cette partie les performances du modèle SC-NHF face à des utilisateurs inconnus.

<i>Modèle</i>	<i>RMSE</i>
SC-NHF	0.4196
NHF	0.4193
SC-NCF	0.4004
NCF	0.4937
SVD	0.4405
KNN	0.5221
Co-Clustering	0.9365
LSTM	0.4908
CNN	0.4910

Tableau 3.7: Comparaison des des modèles suivant l’évaluation avec des utilisateurs inconnus

Du tableau 3.7, nous pouvons observer que nos modèles ont généralement surpassé les résultats des autres modèles, avec comme meilleure performance obtenue avec le modèle SC-NCF. Nous pouvons en conclure que l’approche proposée rencontre moins de difficultés face au démarrage à froid que les algorithmes classiques de FC ou basé exclusivement sur l’analyse de sentiment.

3.5.5 Discussion générale

L’évaluation des deux modèles SC-NCF et SC-NHF a pour objectif de voir l’impact de l’analyse de sentiments sur le FC et hybride respectivement, tout en considérant le cas de démarrage à froid.

Les tests précédents sur les trois *datasets*, démontrent que nos modèles surpassent plusieurs modèles de l’état de l’art. Ceci peut témoigner, entre autres, de l’apport de l’analyse de sentiments aux systèmes de recommandation.

Nous constatons également que le *deep learning* a encore une fois fait ses preuves en présentant de meilleurs résultats avec l’accroissement du nombre de couches cachées des modèles comme le démontre les évaluations de la figure 3.5 et le tableau 3.4.

Aussi, comme le montre le tableau 3.2, nous pouvons noter l’apport positif du *word embedding* dans l’analyse de sentiments que réalise notre modèle.

De plus, le choix des paramètres adéquats reste un critère essentiel à une bonne prédiction, nous avons donc procédé à une étude des hyper-paramètres (voir section 3.5.2) en faisant varier par exemple le nombre d'*epochs*, le *learning rate*, le *dropout* ...etc.

Néanmoins, l'étude présente quelques manques, tel que, l'évaluation des modèles sur un plus grand nombre de *datasets* et dont la densité serait plus élevée. Aussi, développer l'approche choisie de sorte à ce qu'elle puisse opérer une analyse de sentiments sans avoir à être supervisée par les notes des utilisateurs. En traitant ce cas, elle pourrait être appliquée sur des *datasets* où les notes ne sont pas renseignées par les utilisateurs.

3.6 Conclusion

Dans ce chapitre, nous avons présenté notre système ainsi que les différents outils et plateformes utilisés pour sa réalisation. Ensuite, des évaluations ont été présentées en utilisant les différents *datasets*. Enfin, la comparaison avec des modèles déjà existants a montré que notre modèle a donné de meilleures performances, notamment dans le cas du démarrage à froid.

Conclusion

En l'espace de quelques années seulement, les systèmes de recommandation ont réussi à se faire une place au sein de plusieurs grandes enseignes, telle qu'Amazon, en devenant de véritables acteurs commerciaux de l'écosystème des ventes en ligne.

Ainsi, la réalisation de ce projet de licence nous a offert l'opportunité d'étudier en détails plusieurs aspects de ces systèmes et de comprendre leur réel fonctionnement.

Pour ce faire, nous nous sommes inspirées principalement de l'article de Liu et al [2] ainsi que des travaux de Zeghoud et Kerboua [1]. Ces derniers ont respectivement proposé une architecture dédiée à la prédiction des notes qui utilise des réseaux de neurones convolutifs pour l'analyse des avis des utilisateurs, et une architecture hybride combinant le filtrage collaboratif au le filtrage basé contenu selon une architecture neuronale multicouches.

A partir de ces travaux, nous avons proposé un modèle hybride basé sur l'analyse de sentiment en utilisant les techniques de *deep learning*, appelé SC-NHF et une variante de ce modèle en considérant seulement le filtrage collaboratif et l'analyse de sentiment appelée SC-NCF. Ces deux modèles sont conçus pour la prédiction des notes des utilisateurs en se basant sur des Réseaux de Neurones Convolutifs (*Convolutional Neural Network* – CNN), sur un modèle de Perceptrons Multicouches (*Multi Layer Perceptron* - MLP) ainsi que sur le modèle de Factorisation Matricielle Généralisée (*Generalized Matrix Factorization* - GMF).

Pour aboutir à ces résultats, nous sommes passées par différentes étapes qui incluent le prétraitement effectué sur les données exploitées, la conception architecturale de chaque système ainsi que, l'évaluation de la qualité de leurs prédictions (selon la métrique RMSE) sur différents jeux de données : Yelp, Amazon - Vidéo games et Amazon - Gourmet food. Nos modèles ainsi testés ont montré des résultats satisfaisants, particulièrement le SC-NCF qui de par ses prédictions a surpassé tous les modèles auxquels il a été comparé.

Comme perspectives futures, nous envisageons dans un premier temps d'évaluer nos différents modèles en utilisant d'autres bases de données avec une densité plus importante, et dans un second temps, nous souhaitons tester notre architecture avec d'autres types de modèles tels que les réseaux de neurones récurrents (*Recurrent Neural Network* – RNN). D'autre part, il serait intéressant de faire évoluer notre approche en lui ajoutant un module qui permettrait de cerner les préférences de l'utilisateur en se basant sur l'analyse des images des articles avec lesquels il a positivement interagi.

Liste des tableaux

Tableau 3.1 : Statistiques relatives aux jeux de données	38
Tableau 3.2 : Tableau comparatif des variantes du modèle SC-NHF	39
Tableau 3.3 : Tableau représentant l'évaluation de nos approches selon la métrique RMSE sur les différents datasets en fonction de la variation du Dropout	40
Tableau 3.4 : Tableau représentant l'évaluation de nos approches selon la métrique RMSE sur les différents datasets en fonction de la variation du nombre de couches de MLP.....	41
Tableau 3.5 : Comparaison des résultats du filtrage collaboratif et basé sentiments	42
Tableau 3.6: Comparaison des modèles avec des utilisateurs connus	42
Tableau 3.7: Comparaison des des modèles suivant l'évaluation avec des utilisateurs inconnus	43

Table des figures

Figure 1.1:Factorisation de la matrice d'évaluation [5].....	5
Figure 1.2 : Schématisation d'un neurone biologique par un neurone artificiel [7].	8
Figure 1.3: Schéma d'un réseau de neurones [8].	8
Figure 1.4: Architecture d'un réseau de neurones convolutif ayant pour entrée une image. [9]	9
Figure 1.5 : Schéma représentant l'architecture de l'approche RM-DL [10].....	11
Figure 1.6 : Architecture du modèle avec l'hybridation [1]	12
Figure 1.7 : Techniques d'analyse de sentiments [13]	14
Figure 1.8: Classification d'une phrase en appliquant un réseau de neurone convolutif [14]..	15
Figure 1.9 : Représentation de mots dans un espace vectoriel [15]	16
Figure 1.10: Architecture du HRDR : deux réseaux de neurones parallèles pour la représentation des users et des items [2].	17
Figure 2.1 : Exemple de transformation des notes en vecteurs One Hot Encodding	21
Figure 2.2: Architecture du SC-NHF	22
Figure 2.3 : Exemple de transformation en embedding	23
Figure 2.4 : Architecture du sous modèle HybMLP	24
Figure 2.5 : Architecture du sous modèle GMF.....	25
Figure 2.6 : Architecture du sous modèle CNN	26
Figure 2.7 : Couches de concaténation.....	28
Figure 3.1 : Interface de l'application - Fenêtre principale.....	34
Figure 3.2 : Interface de l'application - Sélection du modèle et du dataset	35
Figure 3.3 : Interface de l'application - Evaluation du modèle.....	36
Figure 3.4 : Figure représentant l'évaluation du modèle SC-NHF sur les différents datasets en fonction de l'hyper-paramètre Dropout	40
Figure 3.5 : Figure représentant l'évaluation du modèle SC-NHF sur les différents datasets selon le nombre de couches MLP.	41

Références

- [1]. Zeghoud, S., Kerouba, I. *Système de recommandation hybride basé sur l'apprentissage profond*. Mémoire de fin d'études de licence, option ISIL, Dép. Informatique, USTHB, encadré par L. Berkani, 2019.
- [2]. H. Liu, Y. Wang and Q. Peng et al. *Hybrid neural recommendation with joint deep representation learning of ratings and reviews*. Neurocomputing, 2019.
- [3]. Gasmi, W. *Le filtrage basé sur le contenu pour la recommandation de cours (FCRC)*. Ecole Polytechnique, Montreal (Canada), 2012.
- [4]. F.O. Isinkaye, Y.O. Folajimi b, B.A. Ojokoh. *Recommendation systems: Principles, methods and evaluation*, 2015.
- [5]. https://www.researchgate.net/figure/Diagram-of-matrix-factorization_fig1_321344494, site consulté en Mai, 2021.
- [6]. Balabanović, M., & Shoham, Y. *Fab: content-based, collaborative recommendation*. *Communications of the ACM*, 1997.
- [7]. <https://deeplylearning.fr/cours-theoriques-deep-learning/fonctionnement-du-neurone-artificiel>, site consulté en Mars, 2021.
- [8]. <https://vitalflux.com/deep-learning-neural-network-explained-simply-layman-terms>, site consulté en Avril, 2021 .
- [9]. <https://developersbreach.com/convolution-neural-network-deep-learning>, site consulté en Avril, 2021.
- [10]. Juan, N., Zhenhua H., JiuJun, C., Shangce, G. An effective recommendation model based on deep representation learning, 2021.
- [11]. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T-S.: Neural Collaborative Filtering. (2017). International World Wide Web Conference Committee (IW3C2), WWW 2017, April 3-7, Perth, Australia (2017), <http://dx.doi.org/10.1145/3038912.3052569>.
- [12]. Pak, A. *Automatic, adaptive, and applicative sentiment analysis* (Doctoral dissertation, Université Paris Sud-Paris XI), 2012.
- [13]. <https://www.mdpi.com/2079-9292/9/3/483/xml>, site consulté en Juin, 2021.
- [14]. <https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f>, site consulté en Avril, 2021.
- [15]. <https://towardsdatascience.com/creating-word-embeddings-coding-the-word2vec-algorithm-in-python-using-deep-learning-b337d0ba17a8>, site consulté en Mai, 2021.
- [16]. Kooli, N., Pigneul, E. *Analyse de sentiments à base d'aspects par combinaison de réseaux profonds : application à des avis en français*, 2018.

- [17]. Mnih, A., Salakhutdinov, R.R., *Probabilistic matrix factorization*, in: *Proceedings of the NIPS*, 2008.
- [18]. McAuley, J., Leskovec, J. *Hidden factors and hidden topics: understanding rating dimensions with review text*, 2013.
- [19]. Wang, C., Ble D.M. *Collaborative topic modeling for recommending scientific Knowledge discovery and data mining*. ACM, 2011 .
- [20]. Q. Diao, M. Qiu, C.-Y. Wu, A.J. Smola, J. Jiang, C. Wang, *Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)*, in: *Proceedings of the KDD*, 2014.
- [21]. Kim, D., Park, C., Oh, J., Lee, S., & Yu, H. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM conference on recommender systems*, 2016.
- [22]. Zheng, L., Noroozi, V., & Yu, P. S. *Joint deep modeling of users and items using reviews for recommendation*. In *Proceedings of the tenth ACM international conference on web search and data mining* , 2017.
- [23]. Chen, C., Zhang, M., Liu, Y., & Ma, S. *Neural attentional rating regression with review-level explanations*. In *Proceedings of the 2018 World Wide Web Conference*, 2018.
- [24]. Lu, Y., Dong, R., & Smyth, B. *Coevolutionary recommendation model: Mutual learning between ratings and reviews*. In *Proceedings of the 2018 World Wide Web Conference*, 2018.
- [25]. <https://neuro.cs.ut.ee/the-use-of-embeddings-in-openai-five>, site consulté en Juin, 2021.
- [26]. McLaughlin, M. R., & Herlocker, J. L. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004.
- [27]. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 1994.
- [28]. Bennett, J., & Lanning, S. The netflix prize. In *Proceedings of KDD cup and workshop* , 2007.
- [29]. Chai, T., & Draxler, R. R.. Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature. *Geoscientific model development*, 2014.
- [30]. Ye, W., Zhang, Y., Zhao, W. X., Chen, X., & Qin, Z. A collaborative neural model for rating prediction by leveraging user reviews and product images. In *Asia Information Retrieval Symposium* . Springer, Cham, 2017.
- [31]. https://medium.com/@zhiwei_zhang/yelp-review-classification-b2816d990429, site consulté en Juin, 2021.
- [32]. <https://github.com/diegoschapira/CNN-Text-Classifier-using-Keras.git>, site consulté en Juin, 2021.

- [33]. Stewart, G. W. On the early history of the singular value decomposition. *SIAM review*, 1993.
- [34]. Silverman, B. W., & Jones, M. C. (1989). E. fix and jl hodes : An important contribution to nonparametric discriminant analysis and density estimation: Commentary on fix and hodes . *International Statistical Review/Revue Internationale de Statistique*, 1995.
- [35]. George, T., & Merugu, S. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)* (pp. 4-pp). IEEE, 2005