République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et d'Informatique
Département Informatique

Mémoire de Licence

Filière : Informatique

Spécialité : Ingénierie des Systèmes d'Information et des Logiciels

---

Thème

Système de recommandation sensible au contexte dans un environnement social

---

Sujet Proposé par **:**

Dr BERKANI Lamia

Présenté par :

ABED Nada Fatima-Zohra

REBAI Mohamed Younes

Devant le jury composé de :

M. SELMOUNE Nazih

Mme. HACHEMI Asma

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

Binôme n° : 068 / 2021

Remerciements

Tout d’abord nous rendons grâce à Dieu, le miséricordieux, qui nous a donné la force, la volonté et le courage pour pouvoir accomplir ce modeste travail.

En second lieu, nous tenons à remercier notre encadreur Dr. Lamia BERKANI pour son encadrement, ses conseils et aide précieux et constants qu’elle nous a apporté tout au long de ce travail, ainsi que pour les remarques constructives qu’elle nous a donné qui nous ont permis de mener à bien ce modeste travail.

Nous tenons à remercier les membres du jury pour l’intérêt qu’ils ont porté à notre travail en acceptant de le lire et de l’examiner .

Nous voudrions exprimer notre gratitude envers nos amis et camarades pour leur soutien moral et intellectuel tout au long de cette démarche.

Enfin nous adressons notre plus sincère remerciement à nos chers parents pour leurs soutient moral inestimable ainsi que pour leurs affection inépuisable et leurs précieux conseil. Ils n'ont cessé de prier pour nous durant notre cursus scolaire et nous ont encouragés régulièrement.

Résumé :

Les systèmes de recommandation ont pour objectif de proposer automatiquement aux usagers des objets en relation avec leurs intérêts. Ces outils d'aide à l'accès à l'information sont de plus en plus présents sur les plateformes de contenus, mais les revues de littérature marquent l'absence des systèmes sensibles au contexte des utilisateurs dans un réseau social. Par conséquent le présent projet de licence est orienté vers la proposition d'une approche de recommandation hybride dans un contexte social, avec utilisation des techniques de **Deep Learning (DL)** et la prise en compte de l'information contextuelle selon une architecture basé sur les deux modèles suivant : (1) **HybMLP (Hybrid Multi Layer Perceptron – HybMLP)** permettant une hybridation du filtrage collaboratif et basé contenu ; et (2) **SocHybMLP (Social Hybrid Multi Layer Perceptron – HybMLP)** qui considère l'utilisation de l'information sociale. Les résultats des évaluations que nous avons effectuées sur trois bases de tests ont montré l'apport d'intégration de l'information sociale et contextuelle.

Mots clés : recommandation sociale ; contexte ; deep learning ; HybridMLP ; amitié ; confiance implicite ; confiance explicite

Abstract :

Recommendation systems aim to automatically suggest to users items that are related to their interests. These tools are increasingly present on content platforms, but literature reviews point out the absence of context aware recommender systems in social networks. Therefore, we focus in this work on this issue and we propose a hybrid recommendation approach in social networking context, using Deep Learning (DL) techniques and taking into account contextual information according to an architecture based on the following two models (1) HybMLP (Hybrid Multi-Layer Perceptron - HybMLP) which is the hybridization of collaborative and content-based filtering algorithms, and (2) SocHybMLP (Social Hybrid MultiLayer Perceptron - HybMLP) which takes into account the use of social information. The experiments results conducted on three different datasets demonstrated the added value of using social and contextual information.

Keywords : Social recommendation ; contextual information ; deep learning ; HybridMLP ; friendship ; implicit trust ; explicit trust.

TABLE DES MATIÈRES

CHAPITRE 3 : IMPLEMENTATION ET EXPERIMENTATION

INTRODUCTION

Actuellement, avec l'émergence des technologies de l'information et de communication et compte tenu de la grande quantité de données, la mise en œuvre de systèmes de filtrage d'informations est devenue cruciale pour les sites Web et les applications.

La nécessité de filtrage de données ne cesse d'accroitre, ce qui rend le domaine de recommandation un domaine d'étude d'actualité et important dans tous les contextes d'études tels que l'enseignement et la vente en ligne. Il ne s'agit pas seulement de recommander des articles (livres, produits, films, restaurants…) les plus pertinents à un utilisateur donné, mais parfois aussi il faudrait que cette recommandation tienne compte de l'information contextuelle (exemple : le genre de musique à écouter qui peut varier d'une période de la journée à une autre, la localisation géographique de restaurants…).

Différents algorithmes ont été proposés, dont les plus importants sont : le filtrage collaboratif, le filtrage basé contenu et le filtrage hybride. Avec le développement des médias sociaux, le filtrage social a connu un développement considérable ces dernières années. Notre projet de fin d'études de licence a pour objectif la proposition d'une approche hybride sensible au contexte, tout en utilisant les techniques d'apprentissage profond ou deep learning (DL) pour un apprentissage rapide et efficace tels que les réseaux de neurones profonds.

Après cette introduction, notre mémoire sera structuré autour de trois chapitres, comme suit :

Le chapitre 1 – Etat de l'art : présente les concepts de base liés à la recommandation de manière générale et contextuelle ou sociale de manière spécifique. Il présente également le DL, et des travaux liés à l'utilisation de ces techniques dans la recommandation.

Le chapitre 2 – Conception : propose une approche de recommandation hybride sensible au contexte selon une architecture basée sur HybMLP et MLP.

Le chapitre 3 – Implémentation et expérimentation : présente notre système de recommandation ainsi que les évaluations effectués pour évaluer notre approche en utilisant trois bases de données : MovieLens 1M, Yelp et FilmTrust.

Enfin, nous clôturons notre mémoire par une conclusion générale et quelques perspectives futures.

CHAPITRE 1 :
ETAT DE L'ART

1 - Introduction

Les systèmes de recommandation sont des outils logiciels qui sont utilisés pour fournir des suggestions aux utilisateurs en fonction de leurs besoins.

Ce chapitre sera organisé en plusieurs sections comme suit : recommandation personnalisée, recommandation contextuelle, approches de recommandation contextuelles ainsi que des travaux liés aux systèmes de recommandation sensible au contexte.

2 - Recommandation personnalisée

2.1 - Définition

Les Systèmes de recommandations personnalisés tiennent compte des préférences ou des intérêts de chaque utilisateur, ce qui permet de recommander plus efficacement des éléments particuliers à l'utilisateur ou à un groupe d'utilisateurs (communauté). Il existe principalement trois approches dans le système de recommandation personnalisée : le filtrage basé sur le contenu, le filtrage collaboratif et l'approche hybride. [2]

2.2 - Techniques de recommandation

2.2.1 - Filtrage à base de contenu

Le filtrage basé sur le contenu (FBC), également connu sous le nom de filtrage cognitif, recommande les articles en faisant le rapprochement (la comparaison) entre les profils d'utilisateurs et ceux des articles afin de recommander de nouveaux articles répondant à leurs intérêts.

2.2.2 - Filtrage Collaboratif

Le filtrage collaboratif (FC) a repris le principe de « bouche à oreille ». [4] Ce dernier exploite les « évaluations » faites par des utilisateurs sur certains items, afin de recommander ces mêmes items à d'autres utilisateurs. Nous pouvons distinguer deux approches utilisés pour le FC : basé mémoire et basé modèle.

a)-FC basé mémoire :

Les algorithmes de FC basés sur la mémoire utilisent la base de données de l'utilisateur pour générer une prédiction. Chaque utilisateur fait partie d'un groupe de personnes ayant des intérêts similaires. En identifiant les soi-disant voisins d'un nouvel utilisateur (ou utilisateur actif), il est possible de produire une prédiction des préférences sur de nouveaux articles susceptible de l'intéresser [6]. Ainsi la note potentielle que donnera un utilisateur à un article est calculée en fonction des notes données par son voisinage au même article.

La méthode de recommandation basée sur la mémoire peut être subdivisée en deux sous méthodes : FC basée sur l'utilisateur et FC basée sur les items [2].

- ***FC basé sur l'utilisateur (USER-BASED CF RECOMMENDATION) :*** l'idée du FC basée sur l'utilisateur est que les utilisateurs ayant des évaluations similaires devraient avoir des intérêts similaires, de sorte que la note potentielle d'un utilisateur sur un item sera prédite à partir des notes de ses voisins sur ce dernier. [2]

- ***FC collaboratif basé sur les items (ITEM-BASED CF RECOMMENDATION) :*** cette approche permet, plutôt que de trouver des utilisateurs similaires, de trouver des articles (items) similaires. [3]

b)-FC basé modèle :

Les systèmes basés sur la mémoire ne sont pas adapté aux applications lorsqu'il s'agit de manipuler une grande quantité d'utilisateurs et d'articles. Dans ce cas, les SR basés modèle interviennent par la suite, ce qui peut faire éviter l'important inconvénient. [2]

il s'agit d'un processus d'extraction de certaines informations de l'ensemble de données et l'utilisation de ces informations comme "modèle" pour faire des recommandations sans avoir à utiliser à chaque fois l'ensemble de données complet. [1]

2.2.3 - Filtrage Hybride

Le filtrage hybride comprends la combinaison de plusieurs techniques de recommandation ensemble afin d'améliorer la pertinence des recommandations et remédier aux différents problèmes rencontrées (d'évolutivité, démarrage à froid, manque d'information ou de rareté). Cela peut être fait de plusieurs manières, par exemple : implémenter chaque technique séparément et combiner les résultats, l'utilisation du filtrage basé sur le contenu dans l'approche collaborative et vice versa.

Souvent les algorithmes de recommandation sont combinés avec d'autres types d'information, telles que l'information sociale dans les réseaux sociaux ou encore l'information contextuelle. Nous présentons dans ce qui suit ces deux types d'algorithmes : le filtrage social et le filtrage sensible au contexte.

3 - Recommandation par filtrage social

Ce type de filtrage d'information utilise le contenu social des utilisateurs en fonction de leur parcours dans les différents réseaux sociaux. D'où plusieurs informations peuvent être retirées comme : les liens d'amitié, la confiance... etc. Afin d'enrichir le profil utilisateur et remédier au problème du démarrage à froid dû au manque de données dans la matrice d'usagers.

L'information sociale peut être considérée comme un ensemble de relations énumérées ci-dessous :

- **L'amitié :** L'amitié est une relation unissant deux utilisateurs ayant le même goût ou pas. D'où le problème rencontré par les systèmes de recommandation basé sur l'amitié est de chercher les amis de l'utilisateur qui partagent les mêmes centres d'intérêts que ce dernier.

- **La confiance :** La confiance est la croyance d'un individu nommé confiant, qu'un autre nommé crédible (ou digne de confiance), a la compétence et la volonté de coopérer, pour accomplir une tâche en faveur du confiant.

Dans les systèmes de recommandation la confiance entre les utilisateurs permet d'améliorer la qualité des items proposés aux utilisateurs.

Il existe deux types de confiance :

- **La confiance directe :** Elle représente l'existence d'une relation qui relie directement deux utilisateurs. Un ensemble de formule a été donné afin de calculer la confiance directe dont la formule de *Pearson*, la formule de *Tanimoto* …etc.
- **La confiance indirecte :** La confiance est défini comme étant transitive on calcul la confiance entre deux utilisateurs indirectement connectés. Parmi les formules qui nous permettent de la calculer on cite : *Tidal trust, Mole trust*. [8]

4 - Recommandation Contextuelle

4.1 - Notion de Contexte

4.1.1 - Définition du contexte

D'après Zimmerman et al : Le contexte est toute information qui peut être utilisée pour caractériser la situation d'une entité. Les éléments de description de ces informations de contexte se répartissent en cinq catégories : l'individualité, l'activité, l'emplacement, le temps et les relations. **[**7**]**

4.1.2 - Approches d'intégration du contexte dans les systèmes de recommandation

Il existe trois différents paradigmes algorithmiques pour l'intégration des informations contextuelles dans le processus de recommandation sont présentés :

1. **L'approche de Pré-filtrage Contextuel (contextual pre-filtering)** : Dans cette approche, l'information contextuelle est utilisée comme une étiquette permettant de filtrer les évaluations qui ne correspondent pas à l'information contextuelle spécifiée. Ceci est fait avant d'exécuter la méthode principale de recommandation sur les données sélectionnées restantes.
2. **L'approche de Post-filtrage Contextuel (contextual post-filtering)** : Selon la méthode de post-filtrage, les informations contextuelles sont utilisées après le démarrage de la méthode principale de recommandation 2D (bidimensionnelle)**.**
3. **L'approche de la Modélisation Contextuelle (contextual modeling)** : Dans cette méthode, les informations de contexte sont directement utilisées à l'intérieur de l'algorithme pour générer des recommandations. [8]

4.2 - Approches de recommandation contextuelles

4.2.1 - Approches bio-inspirées

Les algorithmes bio-inspirés sont des techniques d'intelligence artificielle inspirées par les phénomènes biologiques qui se produisent dans la nature. Ces derniers sont adaptés à la résolution des problèmes concernent un espace multidimensionnel dans lequel le problème et les contraintes spécifiées sont de nature dynamique. En raison de l'ajout de contextes, l'espace du problème pour les systèmes de recommandation devient multidimensionnel avec un dynamisme des préférences de l'utilisateur. Par conséquent, ces techniques sont adaptées au traitement de ce problème. [9] Parmi ces approches on cite : Réseau de neurones Artificielle, Algorithme génétique, Algorithme d'optimisation par colonies de fourmis.

4.2.2 - Approches statistiques

Les méthodes de calcul statistique sont une variété d'outils statistiques qui peuvent être appliqués aux données pour en tirer des informations et accéder à une description analytique débouchant sur une modélisation déterministe. L'utilisation de ces techniques aide à comprendre la nature, la distribution et les caractéristiques des données qui peuvent être utilisées dans différentes tâches d'apprentissage.

- ***Support Vector Machine SVM :***

Les SVMs sont des systèmes d'apprentissage qui utilisent un espace d'hypothèses de fonctions linéaires dans un espace de caractéristique à haute dimension. L'idée des SVMs est de rechercher un hyperplan (droite dans le cas de deux dimensions) qui sépare le mieux ces deux classe.

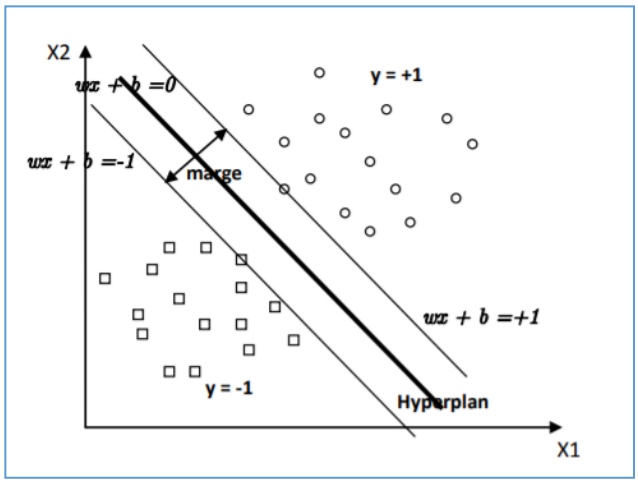

Figure 1.1 : Exemple de classification SVM

En réalité, un hyperplan séparateur n'existe pas toujours, et même s'il existe, il ne représente pas généralement la meilleure solution pour la classification (par exemple erreur d'étiquetage).Dans le cas où les données ne sont pas linéairement séparables, ou contiennent des données mal étiquetées. Ceci peut être fait en admettant une certaine erreur de classification des données, ce qui est appelé "SVM à marge souple (Soft Margin)". [10]

- ***K-Nearest-Nieghbor KNN :***

k-NN est un algorithme standard de classification qui repose exclusivement sur le choix de la métrique de classification. Il est “non paramétrique” et se base uniquement sur les données d’entraînement. à partir d’une base de données étiquetées, on peut estimer la classe d’une nouvelle donnée en regardant quelle est la classe majoritaire des k données voisines les plus proches (d’où le nom de l’algorithme). Le seul paramètre à fixer est k, le nombre de voisins à considérer.

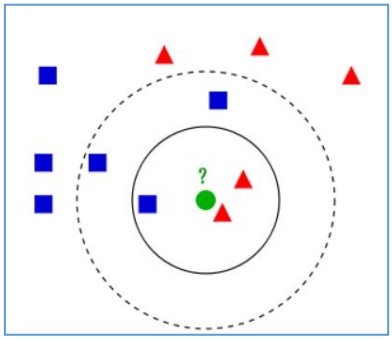

Figure 1.2 : Exemple de classification KNN.

On considère l’exemple suivant ou on cherche la classe du point vert, Si le nombre de plus proches voisins, k, est fixé à 3, la classe du point vert est celle des triangles rouges, car ces derniers sont au nombre de 2 contre un seul carré bleu. Si k vaut 5, la classe du point vert est celle des carrés bleus, au nombre de 3 contre 2 triangles rouges. [11]

4.3 - Techniques utilisées dans la recommandation contextuelle

4.3.1 - Apprentissage automatique

Apprentissage automatique ou « Machine Learning "ML" » en anglais est un sous-ensemble de l’intelligence artificielle (IA) qui est axé sur la création de systèmes qui apprennent ou améliorent les performances en fonction des données qu’ils traitent. [12]

Il existe plusieurs types d’algorithme d’apprentissage automatique les plus communs sont les suivants :

- **Apprentissage supervisé (Supervised Learning) :**

Les données d'entrée ou les données d’entrainement portent une étiquette prédéterminée, par exemple Vrai/Faux, Positif/Négatif, etc. Une fonction ou un classificateur est construit et formé (entrainé) pour prédire l'étiquette des données de test. Le classificateur est correctement réglé (les valeurs des paramètres sont ajustées) pour obtenir un niveau de précision adéquat. [14] Parmi les techniques supervisées on cite : Arbres décisionnels, régression logistique… .

- **Apprentissage non supervisé (Unsupervised Learning) :**

Les données d'entrée ou les données d'entraînement ne sont pas étiquetées. La technique d'apprentissage non supervisé tente de découvrir les similitudes entre les données d'entrée et,

sur la base de ces similitudes, cette technique regroupe les données. (clustering) [13] l'un des algorithmes les plus communs de l'apprentissage non supervisé est : k-means clustering, Association Rules.

- **Apprentissage par renforcement (Reinforcement learning) :**

L'algorithme apprend un comportement étant donné une observation. L'action de l'algorithme sur l'environnement produit une valeur de retour qui guide l'algorithme d'apprentissage. Par exemple : Q-Learning.

4.3.2 - Réseaux de neurones et réseaux de neurones profonds

L'idée de réseaux neuronaux est dérivée des réseaux neuronaux du cerveau humain. Le neurone va recevoir plusieurs entrées d'informations, plusieurs valeurs, qui vont être attaché à un poids qui peut être ajusté. Dans sa forme basique, il va effectuer une somme de l'ensemble de ces variables en fonction de leurs poids. Cette valeur passe ensuite par une fonction d'activation (Relu, sigmoïde…), qui en sera l'unique sortie. Le principe de ces réseaux va donc être d'assembler de grande quantité de neurone entre eux pour former des couches. [15]

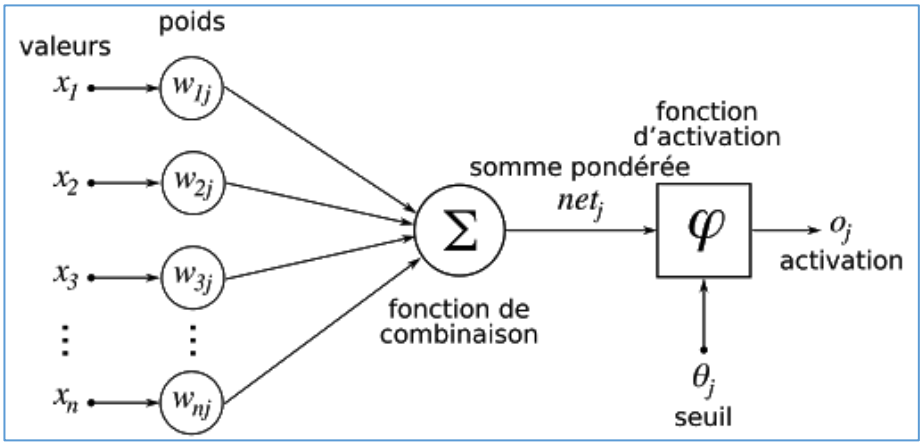

Figure 1.3 : Modèle mathématique d'un neurone.

Les réseaux en couches sont constitués de :

- ***Une couche d'entrée (Input layer) :*** composée de n neurones (un pour chaque entrée du réseau) qui permet de collecter des données du monde extérieur.
- ***Une couche de sortie (output layer) :*** constituée de p neurones (un pour chaque sortie du réseau) cette couche nous donne le résultat prédit par le réseau.
- ***Une couche cachée (Hidden layer) :*** c'est une ou plusieurs couches qui se situent entre les deux premières (la couche de sortie et la couche d'entée), constituées de m neurones. Chaque couche supplémentaire ajoute complexité supplémentaire dans la formation du réseau, mais donnerait de meilleurs résultats dans la plupart des situations.

S'il y a une ou deux couches cachées dans le réseau, celui-ci est alors classé comme un réseau neuronal. S'il y a plus de trois couches cachées dans un réseau, il est alors classé comme un réseau neuronal profond. [16]

Il existe plusieurs types de réseaux de neurones tels que le perceptron multicouche MLP.

- ***Perceptron multicouche (MLP) :***

Il est considéré comme le type le plus simple de réseau neuronal profond. Un MLP se compose d'au moins trois couches de nœuds : une couche d'entrée, une couche cachée et une couche de sortie. À l'exception des nœuds d'entrée, chaque nœud est un neurone qui utilise une fonction d'activation non linéaire. Les couches sont ensuite densément connectées, de sorte que chaque neurone d'une couche est connecté à chaque neurone de la couche suivante. [17]

5 - Travaux liées

Nous présentons, dans cette section, quelques travaux basés sur la recommandation sensible au contexte et/ ou basés sur l'apprentissage profond.

5.1 - Filtrage collaboratif neuronal (NCF)

Cette méthode développée par **He** et **al**. (2017) utilise des réseaux de neurones en utilisant des réseaux de neurones MLP au lieu de fonctions d'interaction linéaires sans générer de produits entre facteurs potentiels. MF a enrichi les modèles obtenus, créant ainsi un troisième modèle appelé NeuMf, dont l'architecture est résumée comme suit.

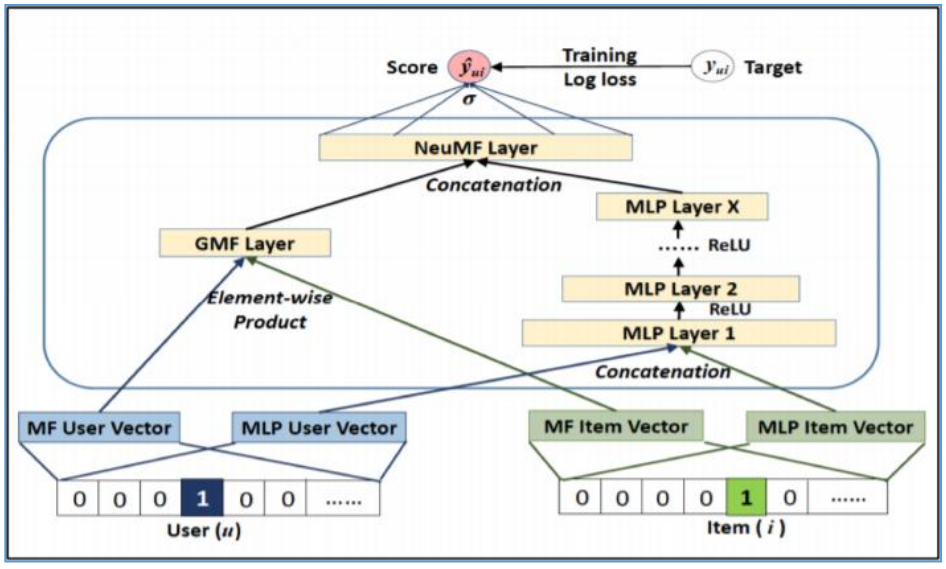

Figure 1.5 : Architecture du système FC neuronal **He** et **al**. (2017).

Dans ce modèle, les identifiants d'utilisateur et d'élément sont donnés en entrée. La couche suivante est embedding layer. Ensuite, il est divisé en deux parties, l'une est le modèle de factorisation matricielle générale et l'autre est la partie qui utilise MLP pour apprendre la fonction d'interaction. Enfin, les résultats de ces deux méthodes sont concaténés dans la

couche NeuMF, puis passent par la fonction d'activation de la couche de sortie qui nous donnera la prédiction du modèle.

5.2 - Context-Aware SVM C-SVM

Cette méthode développée par **Oku** et **al** (2006) prend en compte les contextes des utilisateurs lors de la recommandation.Ils ont pris le contexte de l'utilisateur comme des situations ou des conditions qui influencent les décisions de l'utilisateur par exemple, l'heure de la journée, les compagnons, la météo, la condition physique et ainsi de suite.

Le C-SVM est un SVM étendu en ajoutant des axes de contexte à son espace de caractéristiques. Le C-SVM apprend les préférences des utilisateurs par rapport à chaque contexte, ce qui lui permet de formuler des recommandations adaptées au contexte. Comme le désigne le schéma suivant :

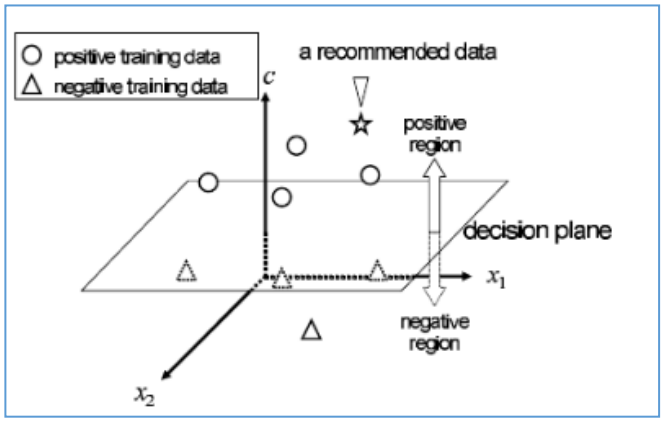

Figure 1.6 : schéma du système C-SVM **Oku** et **al** (2006).

6 - Conclusion :

Nous pouvons conclure de ce chapitre qu'il existe plusieurs méthodes de recommandations, chacune ayant ses avantages et ses inconvénients. Afin d'améliorer les performances des approches traditionnelles, des hybridations sont considérées ainsi que des techniques d'intelligence artificielles sont les algorithmes de machine ou deep learning. Aussi d'autres informations sont combinées avec les algorithmes de recommandation comme l'infromation sociale et l'information contextuelle.

Dans le chapitre suivant nous présenterons notre approche de recommandation d'items qui sera sensible au contexte et basée sur le filtrage social et le filtrage collaboratif tout en s'inspirant des travaux étudiés dans ce chapitre.

CHAPITRE 2 :
CONCEPTION

1 – Introduction

Après avoir défini les différentes méthodes de filtrage dans le chapitre précédent et introduit les différentes techniques de machines Learning et Deep Learning.

Dans ce chapitre, on va exploiter MLP afin d'implémenter une approche hybride basée sur le filtrage collaboratif, cognitif et social.

2 - Proposition d'une approche de recommandation contextuelle

2.1 – Motivation

Kulkarni et **Rodd** (2020) ont publié une revue de la littérature sur les techniques existantes pour la recommandation contextuelle qu'ils ont classé en : techniques bio-inspirées (exemple : les réseaux de neurones et les algorithmes) et techniques statistiques (exemple : Matrix factorization).Cette étude montre le manque de travaux de recommandation contextuelle qui utilisent les techniques de DL dans les réseaux sociaux.

Par conséquent, nous nous intéressons à la proposition d'une approche de recommandation hybride dans un contexte social, avec utilisation des techniques de Deep Learning (DL) et la prise en compte de l'information contextuelle. La dimension sociale se caractérise par les liens d'amitié et de confiance entre les membres du réseau social.

2.2 - Description générale du modèle proposé

La figure 2.1 décrit l'architecture globale de notre approche de recommandation. Cette approche propose une hybridation des algorithmes de FC et FBC avec l'information sociale, selon une architecture DL. Les identifiants des utilisateurs et des items sont donnés en entrée avec les caractéristiques de chacun afin de faire une prédiction d'interaction dont la valeur sera comprise entre 0 et 1. De plus, chaque utilisateur sera accompagné par sa liste d'amis et de personnes à qui il fait confiance ainsi que le degré de confiance qu'il accorde à chacun.

Notre modèle ne devra pas seulement prédire l'interaction d'un utilisateur donné avec les items mais aussi celle de ses amis de confiance.

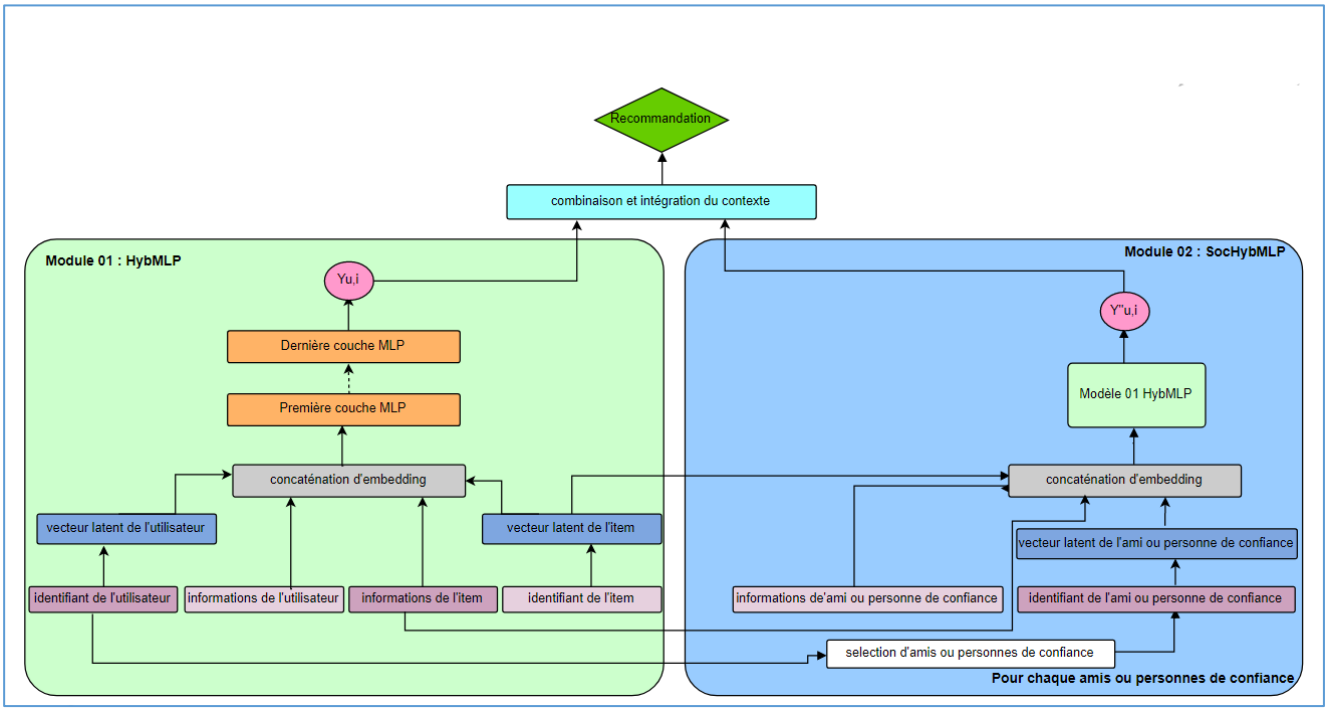

Figure 2.1 : Architecture du modèle proposé.

Dans ce qui suit, nous allons décrire les différentes couches du modèle proposé.

2.3 - Description des différents modules

2.3.1 - Perceptron Multicouches Hybride (Hybrid Multi Layer Perceptron – HybMLP)

Dans ce module nous prenons en entrée les identifiants des utilisateurs et des items, ainsi que les caractéristiques liées à chacun d'entre eux. Ces informations seront ensuite ajoutées aux facteurs appris par le modèle, de sorte qu'elles soient concaténées à *l'embedding*.

L'embedding est la représentation de variables discrètes dans un vecteur numérique continu. En ce qui concerne les réseaux de neurones, ce sont des vecteurs de représentation de variables discrètes appris.

Nous essayons de représenter les items et utilisateurs sous forme vectorielle. Cette représentation modélise les caractéristiques des utilisateurs (ou des items) qui sont automatiquement apprises, de sorte que l'algorithme identifiera les utilisateurs avec un taux de similarité élevé, de même pour les items.

Le module 1, HybMLP se charge de l'apprentissage de la fonction d'interaction, selon la formule suivante :

$$Y_{ui} = 0 \quad sinon\ 1\ s'il\ \text{y a une interaction entre l'utilisateur u et l'item i.} \qquad \textbf{(2.1)}$$

La figure 2.2 montre l'architecture de ce module :

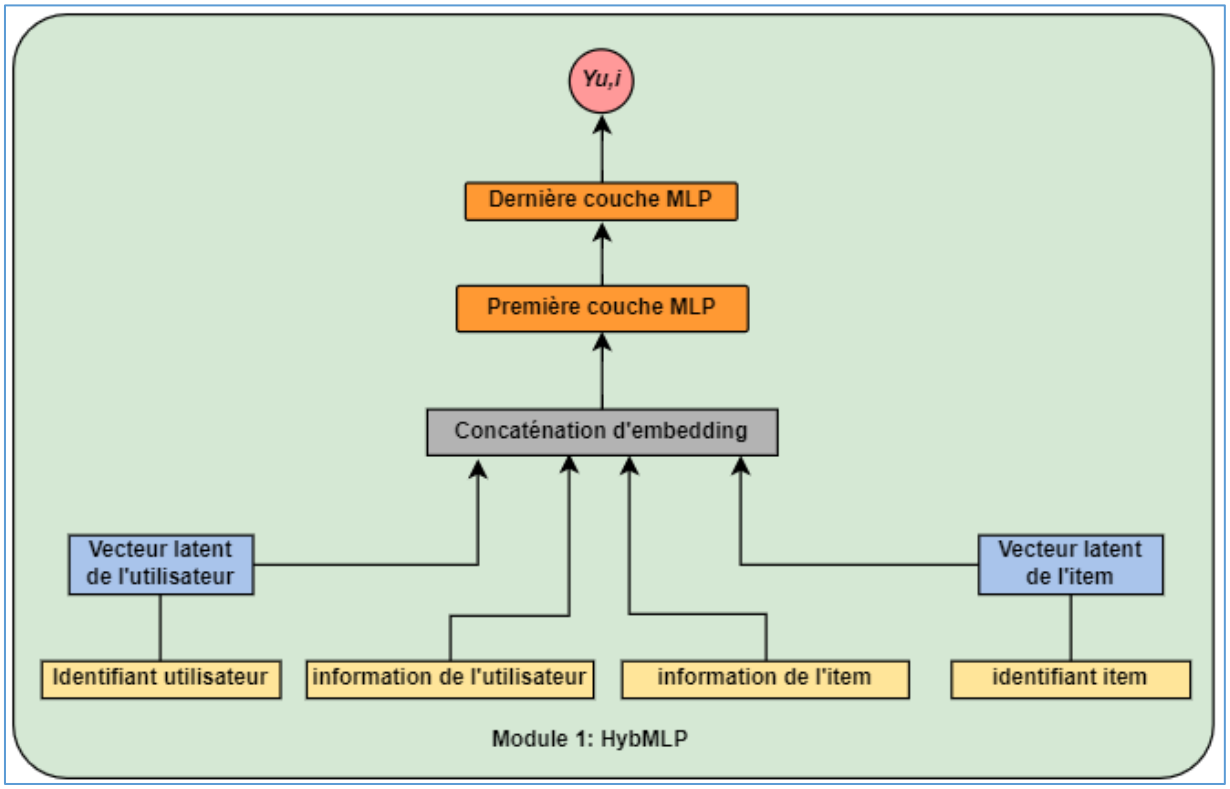

Figure2.2 : Architecture du module 1.

Description textuelle du modèle :

- **Vecteur latent de l'utilisateur** : représente l'embedding des utilisateurs.
- **Vecteur latent de l'item** : représente l'embedding des items.
- **Concaténation d'embedding** : cette couche constitue la première couche passée à notre modèle où se fait la concaténation des embeddings avec les caractéristiques non-apprises des utilisateurs et des items.
- **Autre(s) couche(s) cachée(s) (HybMLP Hidden Layers)) :** Ces couches sont ajoutées pour permettre l'apprentissage des interactions. La fonction d'activation utilisée dans chaque couche est ReLU (Rectified Linear Unit : $(x) = (0, x)$) car elle permet de réduire les risques de sur-apprentissage et de saturation des neurones. La formule du modèle est donnée ci-dessous :

$$\phi\ \mathrm{MLP} = a_L(W_L^T(a_{L-1}(\dots a_2(W_2^T\begin{bmatrix} p_u^M \\ f_u \\ q_i^M \\ f_i \end{bmatrix} + b_2\dots))\ + b_L) \qquad \textbf{(2.2)}$$

Où :

a_L: Fonction d'activation de la couche L.

b_L : Bias de la couche L (fonctionne comme le seuil).

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.

Il se peut qu'il n'y ait pas de couche de traitement alors les caractéristiques seront dirigées vers la couche de sortie. On parlera donc de HybMLP simple (HybMLP-0) qui sera testé dans le prochain chapitre.

On fixe un nombre de nœuds pour la dernière couche, la précédente aura le double du nombre de nœuds fixés et ainsi de suite jusqu'à la première couche Par exemple : si on fixe la dernière couche à 8 alors on a la répartition suivante des nœuds par couches [32→16→ 8].

2.3.2 - Module SocHybMLP

La figure 2.3 illustre l'architecture du module2, SocHybMLP. Nous avons considéré deux approches pour ce module :

La première approche considère que le goût de l'utilisateur *u* est proche de celui de ses amis proches (i.e. utilisateurs avec qui il a un degré social proche). Afin de calculer ce degré, nous allons montrer comment calculer le degré d'amitié et de confiance :

Par contre la deuxième approche considérera que l'utilisateur *u* ne possède pas forcément le même goût que ses amis, on va donc sélectionner à l'aide de la distance cosine avec l'algorithme de K plus proches voisins (KNN) les amis similaires de l'utilisateur *u*. Nous considérons donc que lorsque la distance diminue, la similarité augmente. Ainsi, avec un seuil optimal on sélectionne les amis qui ont une distance cosine inférieur au seuil choisi.

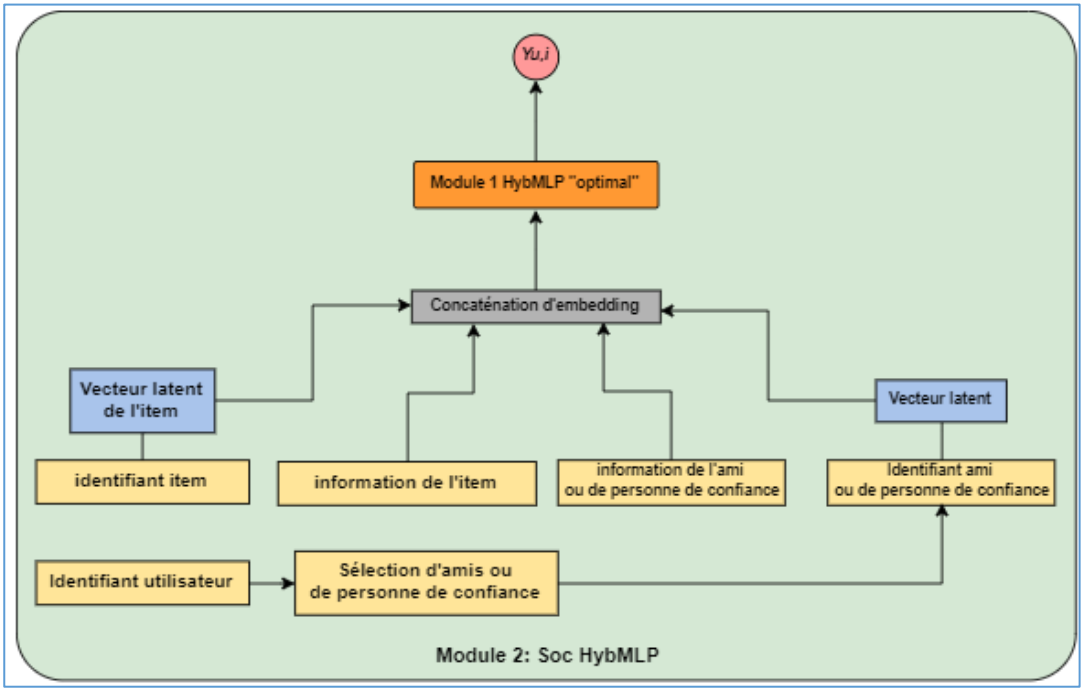

Figure2.3 : Architecture du module 02.

- **Calcul du degré d'amitié :**

Chaque utilisateur est accompagné de sa liste d'amis qui est extraite du graphe social qui modélise les liens d'amitié entre les utilisateurs. L'extraction du degré d'amitié se calcule à partir de la formule de Jaccard :

$$\mathrm{DAmitié(u,v)} = \frac{|\,\mathrm{Fu} \cap \mathrm{Fv}\,|}{|\,F\mathrm{u} \cup \mathrm{Fv}\,|} \qquad \textbf{(2.3)}$$

Où :

Fu : représente les amis de l'utilisateur u.

Fv : représente les amis de l'utilisateur v.

Pour le pseudo code correspondant voir l'algorithme A1.1 en annexe.

- **Calcul du degré de confiance :**

Le calcul du degré de confiance se fait en deux étapes : la première consiste à calculer la confiance directe entre utilisateur ; et la deuxième est la propagation de celle-ci.

- **Confiance directe (explicite) :**

Nous calculons le degré de confiance avec la formule de chan et al [19] comme suit :

$$\boldsymbol{et_{u,v}} = \sqrt{\frac{\boldsymbol{d^{-}(v_v)}}{\boldsymbol{d^{+}(v_v) + d^{+}(v_u)}}} \qquad \textbf{(2.4)}$$

Où :

- ✓ $\boldsymbol{d^{+}(v_v)}$ représente le nombre d'arcs sortants de l'utilisateur *v* dans le réseau de confiance.
- ✓ $\boldsymbol{d^{-}(v_v)}$ représente le nombre d'arcs entrants de l'utilisateur *v* dans le réseau de confiance.

Pseudo code correspondant (voir l'algorithme A1.2 en annexe).

- **Confiance indirecte (implicite) :**

La confiance implicite représente la similarité des évaluations entre utilisateurs d'après la matrice d'usage. De nombreuses métriques de confiance ont été proposées pour mesurer la confiance implicite à partir des évaluations des utilisateurs afin d'améliorer les performances du système de recommandation en réduisant les risques d'erreur. Nous avons donc tous d'abord opté pour la formule de **chen** et **al** [18] qui est la suivante d'où la valeur de la confiance implicite comprise entre 0 et 1 :

$$\frac{\sum_{\mathbf{1}}^{\mathrm{Iu,v}}(\mathrm{ru,i} - \mathrm{r^{-}u}) \times (\mathrm{rv,i} - \mathrm{r^{-}v})}{\sqrt{\sum_{1}^{Iu,v}(\mathrm{ru,i} - \mathrm{r^{-}u})^{2}} \times \sqrt{\sum_{1}^{Iu,v} (\mathrm{rv,i} - \mathrm{r^{-}v})^{2}}} \qquad \textbf{(2.5)}$$

Après l'avoir testé nous avons constaté des anomalies de calculs comme le montre l'exemple suivant :

Considérant, un utilisateur *u*, qui a évalué 100 items et un utilisateur *v*, qui a évalué 2 items. Ces deux utilisateurs ont évalué de la même manière un seul item. La formule précédente (Formule2.5) considère cette confiance comme étant maximale donc de valeur 1, ce qui n'est pas très logique. Nous avons donc utilisé une autre formule de **Hwang** et **al** [18] qui est la suivante :

$$it_{u,v} = \frac{1}{|\mathrm{I_{u,v}}|}\sum_{1}^{I_{u,v}}\left(1 - \frac{|p_{u,i} - r_{u,i}|}{r_{max}}\right) \quad \textbf{(2.6)}$$

Cette formule a estimé la confiance de l'exemple précédent à 0.7 et la valeur de confiance maximale (qui est 1) est fixée à l'utilisateur avec lui-même.

Où :

$I_{u,v}$: L'ensemble d'items commun entre l'utilisateur u et v.

$r_{u,i}$: La note moyenne de l'utilisateur u.

$\bar{r}u$: La note moyenne de l'utilisateur u.

$p_{u,i}$: Une évaluation prédite pour l'utilisateur u sur l'élément i.

r_{max} : La note maximale donnée sur une échelle de notation par exemple de 1 à 5.

Pour plus de détail sur l'implémentation de cet algorithme voir l'algorithme A1.3 en annexe.

Nous avons combiné les deux types de confiance, implicite et explicite, avec la formule suivante :

$$ct_{u,v} = \beta \times et_{u,v} + (1 - \beta) \times it_{uv} \qquad 0 < \beta < 1 \quad \textbf{(2.7)}$$

Dans le chapitre suivant on va évaluer cette formule afin de trouver la meilleure valeur de β.

Puis nous avons procédé à la propagation de la confiance explicite à l'aide de la formule de Chen et al. (2021). En s'inspirant de la formule de Mole Trust, on considère seulement les chemins de longueur inférieur à 3 afin de maximiser la confiance sinon le degré de confiance est de 0.

$$Itrust_{ui,uj} = \frac{\sum_{1}^{n}(W(k) \times W_{direct}^{k})}{\sum_{1}^{n} W(k)} \quad \textbf{(2.8)}$$

Où :

$Itrust$: La confiance indirecte entre l'utilisateur u_i et l'utilisateur u_j.

$\mathrm{W_{direct}^{k}}$: Représente la valeur de confiance avant que l'utilisateur u_i atteigne l'utilisateur u_j dans le chemin de confiance de k.

$W(k)$: Représente le poids du k nième chemin de confiance et calculé avec la formule suivante :

$$\prod_{i=1}^{l-1} Dtrust_i(x, y) \quad \textbf{(2.9)}$$

Où :

l : Représente la longueur du k-ième chemin de confiance.

$Dtrust$: Représente la confiance directe entre les utilisateurs u_x et u_y dans le k-ième chemin.

2.3.3 - Module de recommandation

La figure 3.1 illustre l'architecture du module de recommandation. Dans ce module on procédera à la combinaison des résultats des deux derniers modules avec la formule suivante :

$$Pr_i = \alpha \times PrH_i + (1 - \alpha) \times PrS_i \qquad 0 < \alpha < 1 \qquad \textbf{(2.10)}$$

Où :

Pr_i : La prédiction d'interaction finale de l'item *i* par le modèle.

PrH_i : La prédiction d'interaction de l'item *i* par HybMLP.

PrS_i : La prédiction d'interaction de l'item *i* par SocHybMLP.

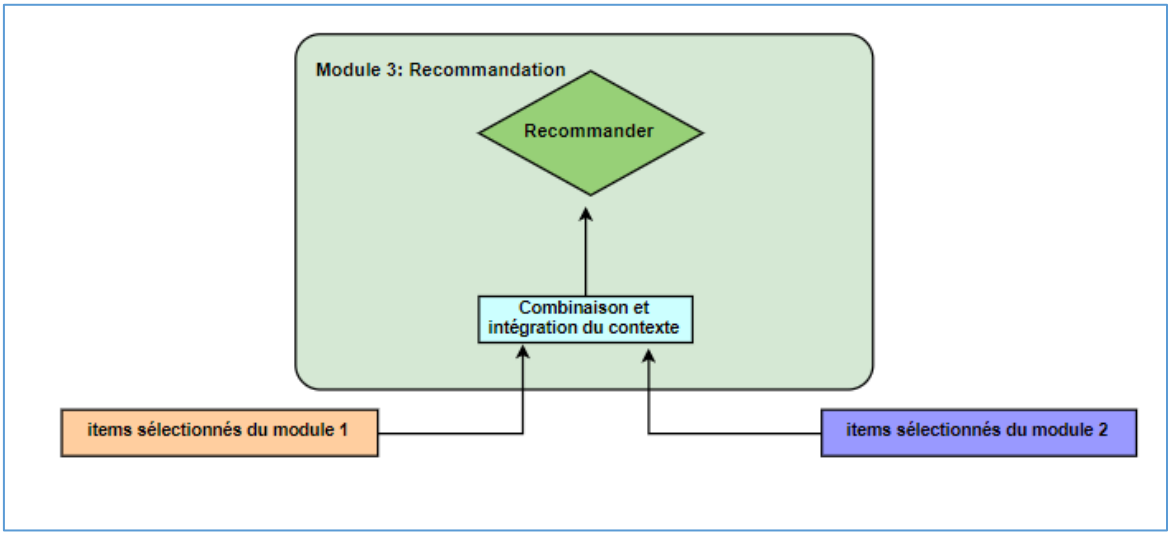

Figure 2.4 : architecture du module 03.

Dans le chapitre suivant on va évaluer cette formule afin de trouver la meilleure valeur de α.

Nous considérons les deux manières suivantes de prise en compte du contexte, selon la base utilisée :

- ***Base Yelp*** :

La base de données Yelp met à disposition la latitude et longitude de chaque restaurant exploité pour obtenir sa géo localisation correspondante. On a calculé la distance entre la localisation de l'utilisateur *u* et l'item *i* à l'aide des fonctions prédéfinies dans Python (voir annexe A.1). Nous avons pris en compte la modélisation réelle des problèmes de géo localisation dans laquelle on permet à l'utilisateur de choisir une distance limite à ne pas dépasser, par exemple :

L'utilisateur *u* souhaite se rendre à des restaurants qui ne se situent pas plus de 50 km loin de chez lui. Notre système vise alors à lui recommander des restaurants avec une distance inférieure ou égale à 50 km.

- ***Base Movie Lens 1m*** :

Quant à movie lens, on prend en considération la colonne Timestamp qui représente une codification de l'instant où l'utilisateur *u* a interagi avec litem *i*. A l'aide des fonctions déjà existantes dans Python (voir annexe A.1), nous avons pu extraire la saison dans laquelle l'item *i* a été visionné. Ensuite, nous avons pris la saison *s* dans laquelle l'item *i* a été le plus vu. Puis, nous considérons que l'item *i* est plus probable d'être visionné en saison *s*. Par exemple :

Prenons le film **«** *Holidate »* on considère qu'il a été vu 20 fois en hiver, 5 fois en été, 2 fois en printemps et automne donc ce film est un film qui a plus de chances d'être vu en hiver.

On intègre le contexte à la fin (post-filtering) pour recommander seulement les items sensibles au contexte choisi. (Pseudo code voir algorithme A1.4 en annexe).

3 – Entraînement

3.1 - Gestion du Dataset

3.1.1 - Exploitation des données

Nous avons exploité trois bases de données : MovieLens1M, yelp et FilmTrust. Les données extraites à partir de ces dernières doivent être traitées et transformées avant de passer par le modèle. Pour expliquer ce traitement, nous prenons la base Yelp qui contient un ensemble d'utilisateurs ainsi qu'un ensemble d'items qui représentent des restaurants et chaque utilisateur est accompagné de sa liste d'amis. Nous avons représenté ces données sous un format exploitable par le modèle. Nous avons aussi procédé au nettoyage des utilisateurs qui n'ont jamais interagi avec des items. Par exemple, il se pourrait qu'après le nettoyage d'utilisateurs, certains amis ne figureront plus dans le dataset, c'est pour cela qu'on a procédé aussi au filtrage des amis (et liens de confiances dans FilmTrust).

3.1.2 - Transformation des données explicites en données implicites

1) Définition des données implicites et explicites

Les données explicites représentent par exemple la note que peut donner un utilisateur à un item qui indique si oui ou non l'utilisateur a apprécié l'item…etc. Tandis que les données implicites sont des données où l'utilisateur n'exprime pas son avis envers l'item, cela pourrait être un clique, un achat ou bien un film visionné. Il est donc, dans ce cas, impossible de déterminer si l'utilisateur a apprécié l'article ou pas. Dans la vie quotidienne, les utilisateurs expriment rarement leurs avis sur les items avec lesquels ils interagissent. En conséquence, il devient très difficile de collecter des données claires, il est donc préférable de concevoir un modèle qui répond aux besoins des utilisateurs quotidiens que de concevoir un pour des utilisateurs «idéaux» qui partagent souvent leurs expériences. Ainsi, si nous considérons toutes les interactions utilisateur-item du dataset comme étant des instances positives, nous aurons plus de données à exploiter.

2) Construction du dataset

Nous ne pouvons pas permettre que le modèle soit formé uniquement d'instances positives, ce dernier n'apprendra qu'à donner des résultats positifs. Par conséquent, nous devons collecter des exemples négatifs, en considérant l'absence d'interaction entre l'utilisateur et l'item comme instance négatif. Le choix nous revient quant au nombre d'instances négatives que nous voulons ajouter pour déterminer l'équilibre entre les instances positives et négatives. Nous notons que par "instance positive", le terme "instance négative" ne signifie pas nécessairement que l'tem *i* n'est pas pertinent à l'utilisateur *u*. Il est juste plus probable qu'un utilisateur s'intéresse à un item avec lequel il a interagi plutôt qu'à un autre.

- **Exploitation de l'information sociale :**

L'utilisation de l'information sociale a été largement étudiée dans le cadre de nombreuses recherches. Il existe encore différentes manières d'intégrer cette dernière dans les systèmes de recommandation. Dans notre travail nous avons discuté deux approches différentes afin d'incorporer l'amitié. D'après **Ma** et **al** [19] le goût de l'utilisateur est proche du goût moyen de ses amis. Cependant, cela n'est pas toujours vrai car un utilisateur *u* peut avoir des centaines d'amis mais ne partage les mêmes centres d'intérêt qu'avec certains d'entre eux. Premièrement on a filtré les amis d'un utilisateur *u* en fonction de leur proximité vis-à-vis de ce dernier en calculant le degré d'amitié avec la formule de Jaccard. Cette valeur nous a permis de sélectionner les amis les plus proches de l'utilisateur *u* en définissant un certain seuil. Concernant la deuxième approche, nous avons sélectionné les amis de l'utilisateur *u* qui ont le même goût que ce dernier avec un seuil optimal en calculant la distance cosine avec KNN. De même pour la confiance, un degré a été calculé à l'aide de la formule de **Chen** et **al** [18]**.** Comme nous avons aussi opté pour le calcul de la confiance implicite qui prend en considération la similarité de notation entre utilisateurs. Afin de maximiser la confiance, nous avons combiné les deux valeurs calculées précédemment avec la formule présentée précédemment**.** Contrairement à l'amitié nous avons procédé à la propagation de la confiance tout au long du graphe social afin de prendre en compte l'avis des utilisateurs qui ne sont pas directement liés à l'utilisateur *u* à l'aide de l'algorithme proposé par **chen** et **al (2021)**. Par exemple, quand un utilisateur *A* fait confiance à l'utilisateur *B*, si *B* à son tour fait confiance à *C*, le modèle considère que *A* fait aussi confiance à *C*. Les valeurs de confiance et d'amitiés sont respectivement associées à chaque couple (utilisateur, personne de confiance) et (utilisateur, ami). Aussi, chaque item sera accompagné de la moyenne qui représente le goût moyen des amis (ou de personnes de confiance) de chaque utilisateur *u*. Après avoir sélectionné ces individus, nous les faisons passer dans le modèle HybMLP afin de prédire leurs interactions avec le même set d'items introduits à l'utilisateur *u*.

Remarque :

Dans FilmTrust il n'existe aucun lien d'amitié, donc on a considéré seulement les liens de confiance. Par contre avec Yelp, il n'existe aucun lien de confiance donc on a considéré seulement les liens d'amitié.

3) Répartition du dataset : nous avons réparti les données en données d'entrainements et de tests :

● ***Les données d'entraînements :*** Nous prenons un ensemble de données implicites qui ne contiennent que des instances positives jusqu'à présent et y ajoutons des instances négatives.

On va choisir seulement les pairs (utilisateur, item) où il n'y a pas eu d'interaction.

● ***Les données tests :*** Pour chaque utilisateur, nous choisissons au hasard un item avec lequel il a interagi (instance positive) et y ajoutons par la suite 99 items avec lesquels il n'a pas interagi (instances négatives). En effet, nous considérons que seulement 1% des données sont pertinentes pour l'utilisateur et essayons de voir si le modèle a considéré l'item le plus susceptible de plaire à l'utilisateur plus que ceux qui ne le sont pas.

Remarque :

Nous notons que les paires (utilisateurs, items) qui existent dans les données de tests n'existent pas dans les données d'entraînements.

3.2 - Entraînement du Modèle

3.2.1 - Fonction de coût (loss function)

Pour l'entrainement de notre modèle HybMLP, nous avons utilisé la fonction Binary cross entropy car il s'agit d'un problème de classification binaire et c'est la meilleure fonction adapté à ce genre de problèmes [20]**.** La formule de la fonction est la suivante :

$$L = -\sum_{(u,i)\in Y\cup Y^{-}} y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui}) \qquad \textbf{(2.11)}$$

Où :

y_{ui} : Valeur de l'interaction entre l'utilisateur u et l'item i.
$\hat{y}_{ui}$: Prédiction de l'interaction entre l'utilisateur u et l'item i.
Y : Ensemble d'instances positives.
Y^{-} : Ensemble d'instances négatives.

3.2.2 – Optimisation

Des algorithmes d'optimisation sont aussi utilisés pour mettre à jour les valeurs des poids du modèle. Nous utilisons Adam (Adaptive Moment Estimation) pour l'entraînement de HybMLP. Adam adapte le taux d'apprentissage à chaque paramètre et ajuste les poids pour un groupe de données (batch).

4 – Conclusion

Après avoir expliqué les détails du système de recommandation hybride basé sur le filtrage collaboratif et basé contenu avec l'information sociale et contextuelle. Nous présenterons dans le prochain chapitre les détails d'implémentation et d'évaluations.

CHAPITRE 3 :

Implémentation et Expérimentation

1 – Introduction

Dans ce chapitre, nous présentons les détails liés à l'implémentation et l'évaluation de notre système de recommandation hybride sensible au contexte combinant le FC et le FBC avec le filtrage social.

Ce chapitre est composé de trois sections : (1) une présentation de notre système et son implémentation ; (2) les expérimentations effectuées pour l'évaluation de nos différents algorithmes ; et (3) une comparaison entre les différents modèles proposés.

2 - Notre système de recommandation

2.1 - Environnement de développement

Pour développer l'approche présentée précédemment ci-dessus, nous avons utilisé le langage de programmation Python version 3.8. Le framework d'apprentissage profond (Deep Learning) utilisé pour développer le modèle est ***Keras***[1], avec ***Tensorflow***[2] comme Backend. Concernant la lecture et la conversion des données, nous avons utilisé la bibliothèque ***Pandas***[3], et pour l'affichage des graphes la bibliothèque ***Matplotlib***[4]. Par rapport au CPU, la vitesse du GPU (unité de traitement graphique) est plus adaptée aux modèles d'entraînement. Pour ce faire, nous avons opté pour les outils gratuits que Google met à disposition de tout programmeur ne disposant pas d'une carte graphique puissante sur son ordinateur. Il s'agit de ***Colaboratory***[5]**,** un bloc-notes où nous utilisons le processeur Google pour importer le code et l'exécuter, puis nous renvoyons les résultats sur la console du bloc-notes. Seules les versions Python 2.7 et 3.6 sont prises en charge et vous pouvez sélectionner le type d'accélérateur que vous souhaitez utiliser.

2.2 - Description des modules de notre système de recommandation

Afin de faciliter la manipulation de notre système de recommandation, nous avons conçu une application développée en Java, en utilisant Swing pour la création de l'interface graphique. Un environnement virtuel python est requis. Il doit contenir toutes les bibliothèques mentionnées dans la section précédente, à savoir ***Pandas***, ***NumPy***, ***Matplotlib***, ***Keras*** et ***Tensorflow***.

La figure ci-dessous (Figure 3.1) présente la fenêtre principale de du système de recommandation que nous avons implémenté.

[1] https://keras.io
[2] https://www.tensorflow.org
[3] https://pandas.pydata.org
[4] https://matplotlib.org/3.1.0
[5] https://colab.research.google.com

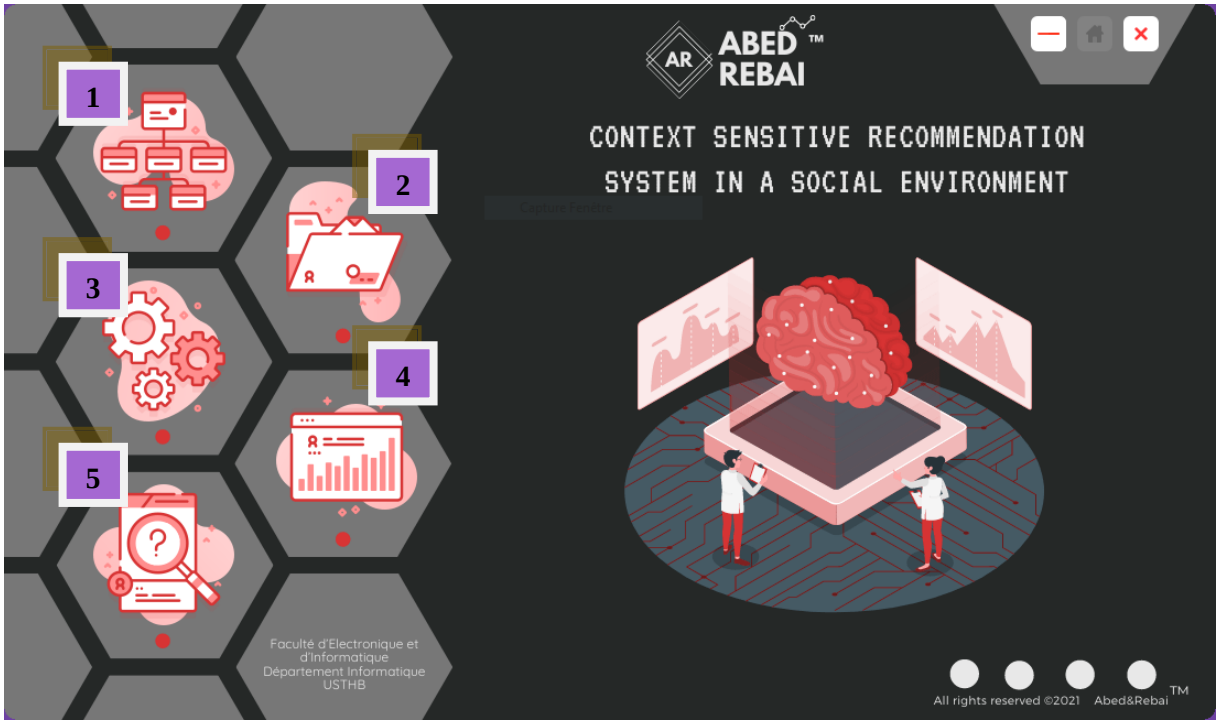

Figure 3.1 : Implémentation de notre système de recommandation – Fenêtre principale

Description des fonctionnalités décrites dans la figure 3.1 :

« 1 » : Page où nous avons présenté les modèles HybMLP et SocHybMLP et chaque modèle est accompagné avec sa description (voir la figure A2.1 en annexe).

« 2 » : Donne n à l'utilisateur la possibilité de choisir l'un des trois datasets (FilmTrust, MovieLens 1m et Yelp), créer un trainset et un testset et les afficher (voir la figure A2.2 en annexe).

« 3 » : Les paramètres d'apprentissage du modèle. On autorise les utilisateurs à personnaliser ces paramètres et à sélectionner la structure à former (voir la figure A2.3 en annexe).

« 4 » : Affichage des tests et résultats du modèle entraîné.

« 5 » : Informations à propos du développement du système de recommandation.

3 - Métriques d'évaluation

Comme mentionné dans le chapitre précédent nous allons évaluer tout d'abord le coté collaboratif basé contenu pour voir l'apport de l'information sociale dans notre approche. Afin de diminuer voir même régler le problème de démarrage à froid, nous allons aussi évaluer le coté sociale seul pour sauvegarder les meilleures conditions du modèle. Par la suite nous évaluerons la combinaison des deux modèles avec et sans contexte pour voir la contribution de celui-ci.

Ainsi expliqué dans la partie gestion des jeux de données, nous nous assurons la simulation des conditions réelles des bases de données (peu d'éléments pertinent à l'utilisateur) où on considère que seul un item sur 100 peut être pertinent pour l'utilisateur.

Dans ce qui suit nous allons présenter trois métriques que nous jugeons pertinentes pour notre expérimentation.

Remarque :

Avant de procéder à l'explication du fonctionnement des métriques on doit tout d'abord aborder la fonction d'activation sigmoïde.

Comme expliqué dans le chapitre précédent, la fonction sigmoïde utilisée pour calculer la probabilité d'interaction entre l'utilisateur u et l'item i retourne un nombre compris entre 1 et 0 ce qui est pertinent pour notre cas étant donné un problème de classification. On considère que l'interaction est forte si le nombre retourné est proche de 1 par exemple 0.9 à l'inverse l'interaction sera considérée comme étant faible par exemple si on obtient 0.02.

Par conséquent, pour chaque utilisateur, nous pouvons trier les items en fonction de la probabilité d'interaction prédite par ordre décroissant. L'article le plus recommandé est en haut de la liste. Là, nous pourrons voir si le modèle estime que l'item le plus susceptible d'être liés à l'utilisateur (la personne avec qui interagit) est meilleure que ceux qui ne sont pas liés (99 autres) ainsi qu'à quel degré le model a considéré l'item.

3.1 - Hit Ratio (HR@k)

Cette métrique permet de compter le nombre de "Hit". Ainsi, pour chaque utilisateur, on prend les k items qui lui sont les plus recommandés par le modèle et on cherche si l'item avec lequel il a interagi figure dans la liste. Si c'est le cas, le score est de 1 et 0 sinon. Puis, il s'agit de considérer tous les utilisateurs et calculer la moyenne.

3.2 - Normalized Discounted Cumulative Gain (NDCG@k)

On évalue la précision de prédiction en prenant en considération la position de l'item pertinent par rapport aux autres dans la liste des items recommandés c'est ce que fait la métrique NDCG, sa formule est la suivante :

$$\mathrm{NDCG@k} = \sum_{\mathrm{i}=1}^{\mathrm{k}} \frac{\mathrm{Pertinence(itemi\,)}}{\log 2(\mathrm{i}+1)} \qquad \textbf{(3.1)}$$

Où :

k : Taille de la liste d'items recommandés.

i : $Position\ de\ l'\ item \in [1, k]$.

$(x) \in \{0, 1\}$.

Nous sélectionnons les k items les plus pertinents prédits pour chaque utilisateur et calculons leur score NDCG. Etant donné que la pertinence d'un item est booléenne et qu'un seul item est pertinent pour chaque utilisateur nous pouvons en conclure que $max\ (NDCG) = 1$ quand l'item pertinent est en première position et $(NDCG) = 0$ quand il ne figure pas dans les k items recommandés ($NDCG \in [0, 1]$). Nous calculons les scores $NDCG$ pour les items recommandés à chaque utilisateur puis calculons leur moyenne que nous considérons comme étant le score $NDCG$ du modèle.

3.3 – Ecart (ET)

Avec cette métrique on calcule la différence entre la probabilité d'interaction maximale qui est à 1 et la valeur d'interaction prédite par le modèle pour l'item pertinent *i* afin d'évaluer la précision de prédiction d'interaction avec la formule suivante :

$$Et = y_i' - y_i \quad \textbf{(3.2)}$$

Où :

y_i' : La valeur maximale de prédiction $y_i' = 1$.

y_i : La valeur d'interaction prédite pour l'item i par le modèle.

On considère que le modèle donne de bon résultat lorsque l'écart diminue.

4 - Evaluation des modèles

Afin de procéder à l'évaluation de notre modèle, nous utilisons trois jeux de données :

- **MovieLens 1M (ML-1M)** [6]**:** contient 1 000 209 évaluations, effectuées par 6040 utilisateurs avec approximativement 3900 films. En plus des détails concernant ces utilisateurs et items, tels que l'âge, le sexe et l'occupation de la personne, le genre, le titre et la date de sortie des films ainsi que d'autres informations.

- **Yelp**[7]**:** nous prenons un échantillon de la base. Cet échantillon contient 110894 évaluations récoltées à partir d'interactions de 5436 utilisateurs sur 4733 restaurants. Des informations sur les utilisateurs et les restaurants sont aussi mis à disposition. Un prétraitement a été fait sur cette base pour transformer les données, comme expliqué dans le chapitre 2, en vecteurs exploitables.

- **FilmTrust**[8] **:** cette base est collectée à partir d'un site qui permet aux utilisateurs de partager des évaluations de films, les utilisateurs possèdent une liste de personnes de confiance. Nous adoptons l'ensemble de données qui contient 35 497 évaluations faite par 1508 utilisateurs sur 2071 items où les évaluations sont attribuées dans un intervalle de 0.5 à 4 avec un pas de 0.5. Cette base de données compte 2853 liens de confiance.

[6] https://grouplens.org/datasets/movielens/
[7] https://www.yelp.com/
[8] https://www.kaggle.com/abdelhakaissat/film-trust

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

Tableau 3.1 : Statistiques relatives aux jeux de données.

Dataset	Nombre d'utilisateurs	Nombre d'items	Evaluations	Densité
Yelp	5436	4733	110 894	0.43%
Movielens1M	6040	3952	1 000 209	4.19%
FilmTrust	1508	2071	35 497	1.14%

Notre objectif par rapport à l'évaluation est de discuter les trois aspects suivants :

- l'apport de l'information sociale sur le filtrage collaboratif.
- l'apport de l'information sociale sur le filtrage collaboratif et le filtrage basé contenu.
- l'apport du contexte dans la recommandation.

Par conséquent on va effectuer les évaluations suivantes :

- l'évaluation préliminaire pour sauvegarder les meilleures conditions des modèles et donc fixer les meilleurs paramètres à utiliser dans la suite de nos évaluations.
- l'apport de l'information sociale sur le filtrage collaboratif.
- l'apport de l'information sociale sur le filtrage collaboratif et le filtrage basé contenu.
- l'apport du contexte dans la recommandation.

Remarque :

Dans ce qui suit nous utilisons la fonction d'optimisation Adam car nous avons constaté qu'elle est largement meilleure que la fonction SGD comme le montre l'exemple suivant sur le même échantillon :

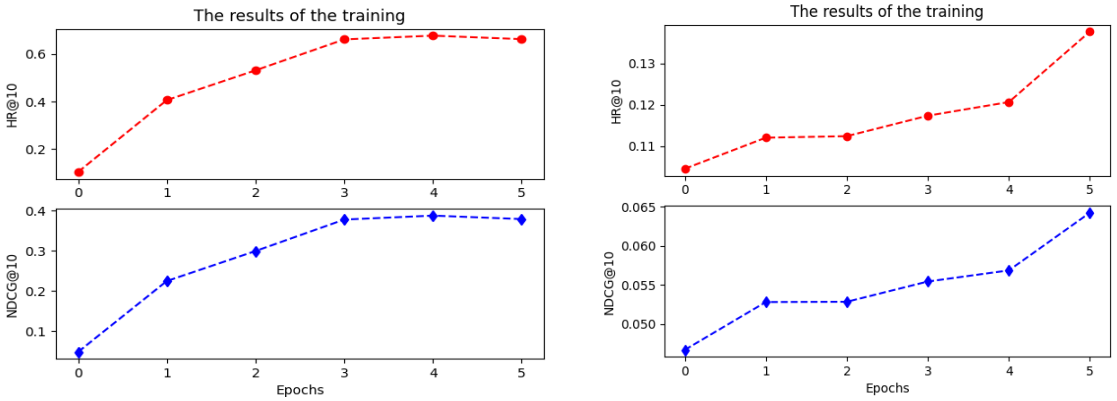

Figure 3.6 : résultats des deux fonctions d'optimisation sur le même échantillon (à gauche Adam- à droite SGD).

Le HR@10 de SGD est si bas qu'on ne peut pas se fier à ses résultats, de même pour NDCG@10,car ce dernier est calculé par rapport au Hit@10 (le hit@10 pour Adam est 6 fois supérieur).

4.1 - Evaluations préliminaires

Nous procédons à cette évaluation pour déterminer les meilleurs paramètres des modèles de notre approche afin d'étudier l'évolution de celle-ci.

Les tableaux ci-dessous contiennent les valeurs des métriques HR, NDCG et Ecart du modèle donné.

4.1.1 - Evaluation HybMLP

Nous évaluons le TOP 10 de chaque utilisateur après un certain nombre d'epochs puis on calcule la moyenne des HR et NDCG de tous les utilisateurs par itération. Pour cette évaluation les jeux d'entraînement sont équilibrés (le nombre d'instances négatives est égal à celui des instances positives).

Movie-lens 1M												
NC	**HybMLP-0**		**HybMLP-1**		**HybMLP-2**		**HybMLP-3**		**HybMLP-4**		**HybMLP-5**	
TE	HR @10	NDC G@10	HR @10	NDC G@10	HR @10	NDC G@10	HR @10	NDC G@10	HR @10	NDC G@10	HR @10	NDC G@10
8	0.560	0.334	0.716	0.443	0.733	0.462	0.746	0.475	0.756	0.484	0.761	0.494
16	0.566	0.334	0.719	0.449	0.750	0.484	0.759	0.492	0.761	0.502	0.771	0.513
32	0.568	0.335	0.743	0.472	0.753	0.487	0.771	0.499	0.779	0.505	0.787	0.522
64	0.561	0.334	0.750	0.479	0.759	0.489	0.771	0.504	0.779	0.508	**0.789**	**0.524**
128	0.569	0.334	0.752	0.484	0.761	0.577	0.774	0.508	0.775	0.510	0.779	0.520

Tableau 3.2 : Evaluation du modèle HybMLP sur ML-1m après 10 epochs avec différents nombres de facteurs (taille de l'embedding) – Predictive factors = 16.

Les résultats obtenus avec ML-1m confirment que les modèles de deep learning ont de meilleures performances.

Les figures ci-dessous (figure 3.7 et 3.8) illustrent l'évolution de la prédiction selon le nombre de couche des deux modèles MLP et HybMLP (pour plus de détails sur les valeurs des métriques voir les tableaux en annexe A4.1 et A4.6).

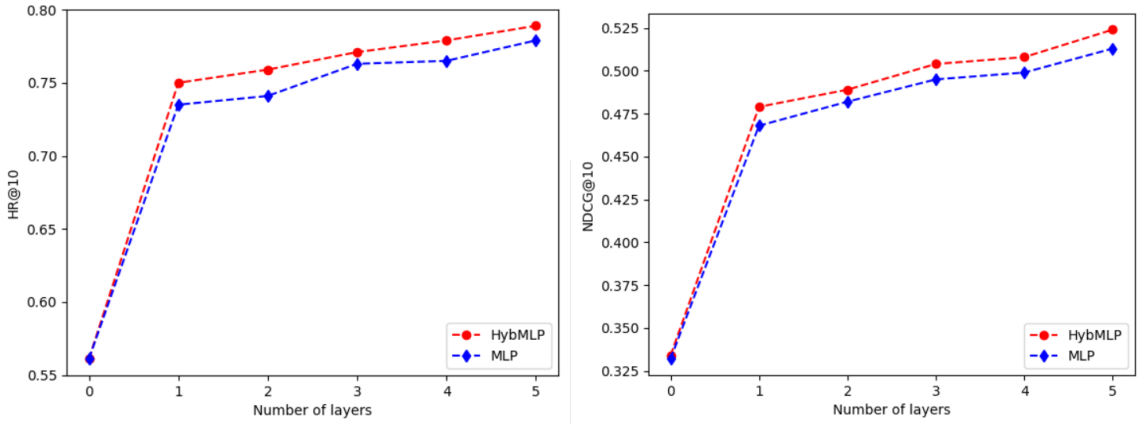

Figure 3.7 : Evolution de HybMLP selon le nombre de couches – ML-1m.

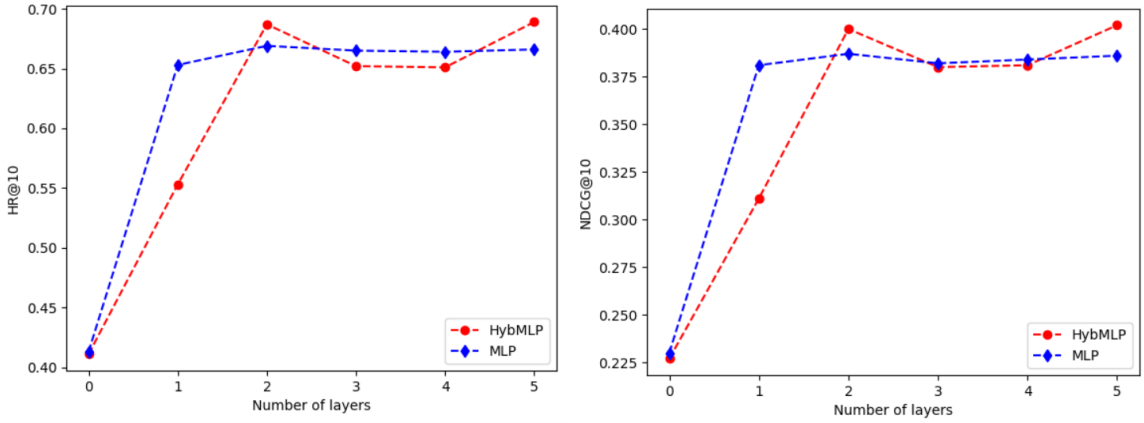

Figure 3.8 : Evolution de HybMLP selon le nombre de couches - Yelp.

Les résultats obtenus avec 5 couches cachées justifient l'utilisation d'un réseau de neurones profond au lieu d'un réseau simple (HybMLP 0) : nous avons obtenu une précision HR = 0.413 et HR= 0.561 respectivement pour Yelp et ML-1m lorsqu'il n'y avait pas de couches cachées pour les traitements. Mais lorsque ce nombre augmente les prédictions et valeurs des métriques s'améliorent.

Les figures ci-dessous (figure 3.10 et 3.9) illustrent les valeurs des métriques HR et NDCG lorsque la taille de l'embedding est variée (pour plus de détails sur les valeurs des métriques voir les tableaux A4.2 et A4.7 en annexe).

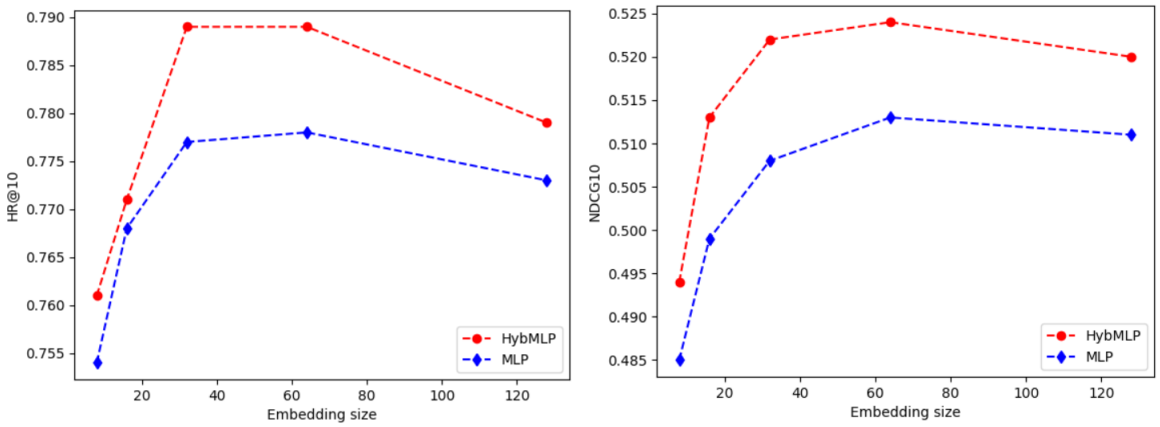

Figure 3.9 : Evaluation selon la taille des embeddings de différents modèles - ML-1m

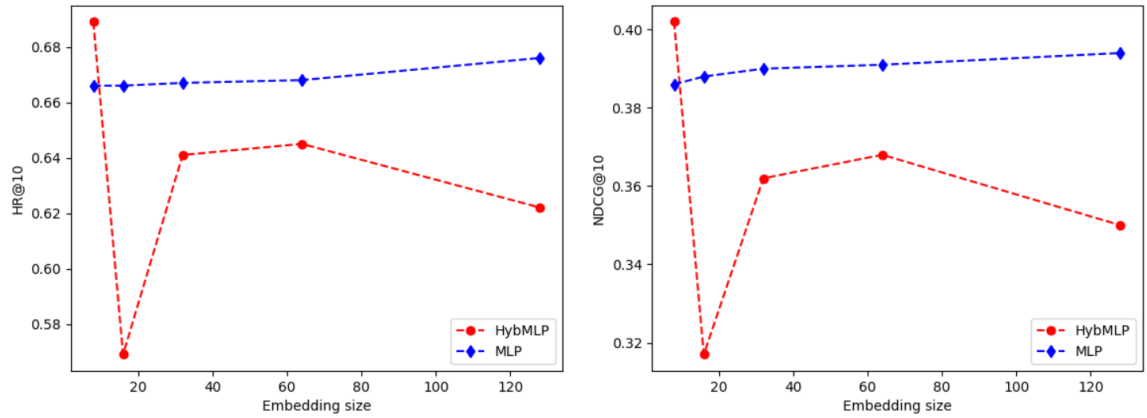

Figure 3.10 : Evaluation selon la taille des embeddings de différents modèles - Yelp.

Nous avons procédé à cette évaluation pour trouver la taille optimale de l'embedding (le nombre de facteurs latents optimal) pour chaque modèle et chaque dataset. Le nombre de facteurs latents le plus optimal pour ML-1m est de 32 facteurs. Par contre, pour Yelp la taille la plus optimale est de 8 facteurs car il existe plus de données relatives aux items et utilisateurs à exploiter par rapport à Movie-Lens.

Afin de voir l'efficacité du modèle, nous avons procédé au brouillage des données d'apprentissage, c'est-à-dire augmenter le nombre d'instances négatives par rapport aux positives. L'objectif est de trouver le nombre d'instances négatives adéquats pour améliorer les performances du modèle.

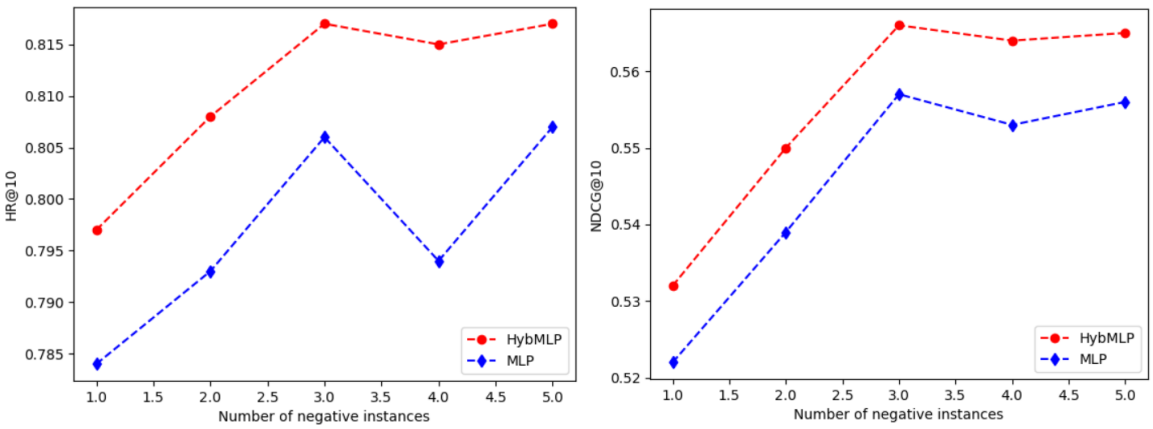

Figure 3.11 : Performance des modèles selon le nombre d'instances négatives par instance positive - ML-1m.

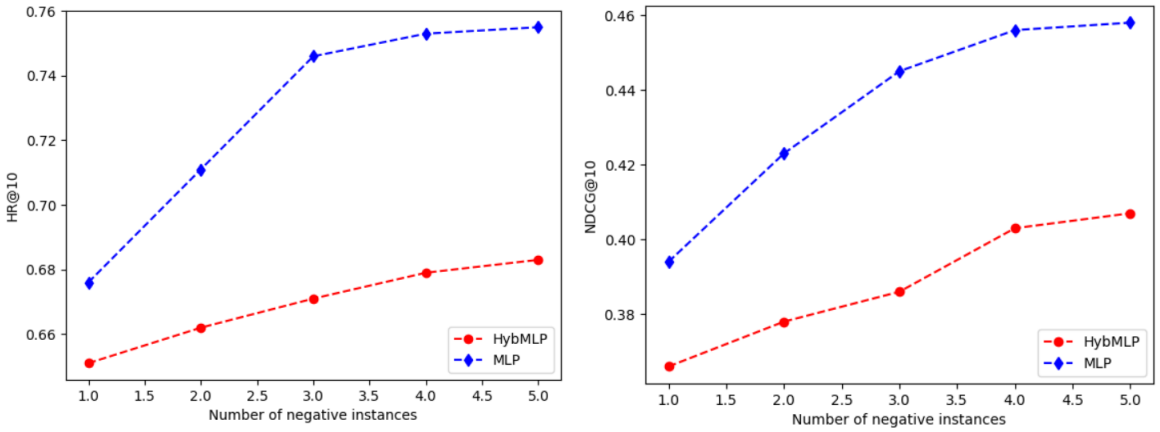

Figure 3.12 : Performance des modèles selon le nombre d'instances négatives par instance positive - Yelp.

Des figures ci-dessus (figure 3.12 et 3.11), nous pouvons déduire que notre modèle fonctionne mieux lorsque nous avons un dataset composé de trois instances négatives par instance positive pour ML-1M, et avec cinq instances négatives pour Yelp (pour plus de détails sur les valeurs des métriques voir les tableaux A4.4 et A4.8 en annexe).

Les figures ci-dessous illustrent l'évaluation des Top-K listes d'items recommandés, avec K allant de 1 à 10 (pour plus de détails sur les valeurs des métriques voir les tableaux A4.5 et A4.9 en annexe).

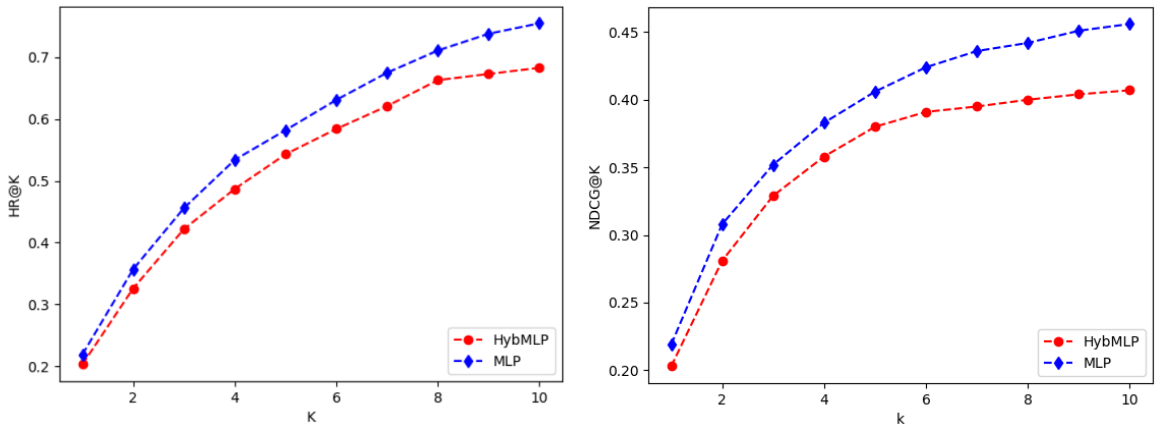

Figure 3.13 : Evaluation du Top-K de recommandation d'items avec K variant de 1 à 10 - Yelp.

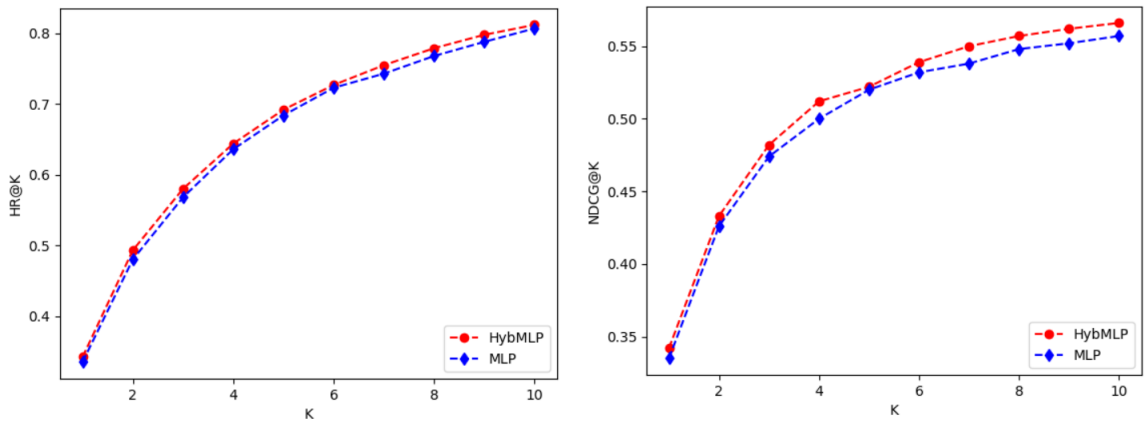

Figure 3.14 : Evaluation du Top-K de recommandation d'items avec K variant de 1 à 10 - ML-1m.

4.1.2 - Evaluation MLP-Film Trust

Nous procédons de la même manière que dans HybMLP.

Film trust												
NC	MLP-0		MLP-1		MLP-2		MLP-3		MLP-4		MLP-5	
TE	HR @10	NDC G@10	HR @10	NDC G@10	HR @10	NDC G@10	HR @10	NDC G@10	HR @10	NDC G@10	HR @10	NDC G@10
8	**0.9065**	**0.7976**	0.8999	0.7934	0.8979	0.7954	0.9019	0.7903	0.9012	0.7851	0.9012	0.7765
16	**0.9038**	**0.7997**	0.9025	0.7959	0.9065	0.7969	0.9038	0.7889	0.8999	0.7926	0.8919	0.7897
32	**0.9065**	**0.8005**	0.9019	0.7930	0.9012	0.7926	0.9038	0.7896	0.8979	0.7915	0.8966	0.7862
64	**0.9045**	**0.7985**	0.9045	0.7975	0.9005	0.7950	0.9038	0.7885	0.8959	0.7889	0.8992	0.7937
128	**0.9065**	**0.8015**	0.9019	0.7925	0.9045	0.7893	0.9032	0.7902	0.8912	0.7814	0.9005	0.7809

Tableau 3.3 : Evaluation du modèle MLP sur FilmTrust avec différentes tailles d'embedding différents nombre de couches cachées – Predictive factors = 128

Du tableau 3.3, il est clair qu'on a obtenu de bon résultats mais les valeurs sont classés de manière qu'on remarque parfois une diminution puis une augmentation des valeurs de métriques car le jeu de données Film Trust n'est pas volumineux d'où un réseau de neurones simple (MLP-0) suffit pour donner de bons résultats contrairement à Yelp et Movie Lens 1M.

Les figures ci-dessous (figure 3.15 et 3.16) présentent respectivement l'évolution de la prédiction selon le nombre de couches et différentes tailles d'embeddings du modèle MLP (pour plus de détails sur les valeurs des métriques voir les tableaux A4.14 et A4.15 en annexe).

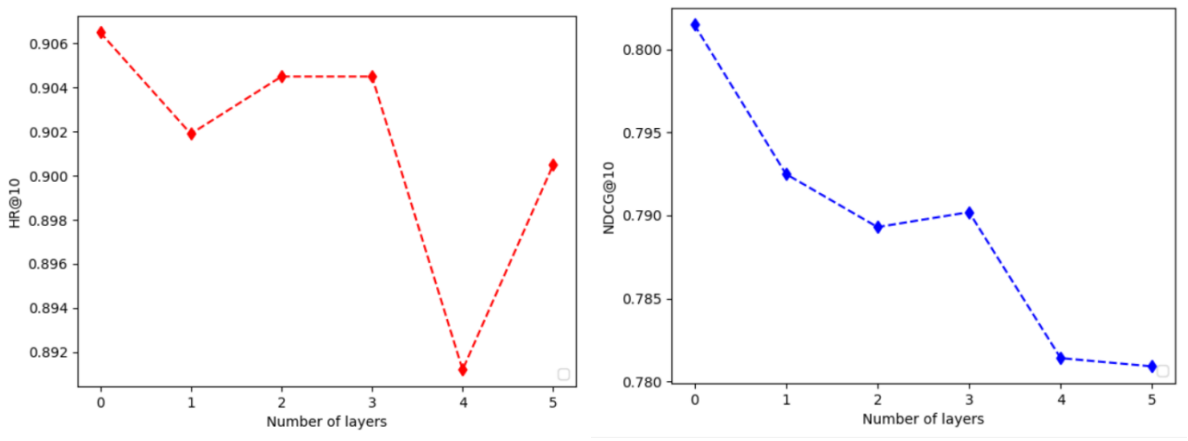

Figure 3.15 : Evaluation du modèle MLP sur Film trust selon le nombre de couche.

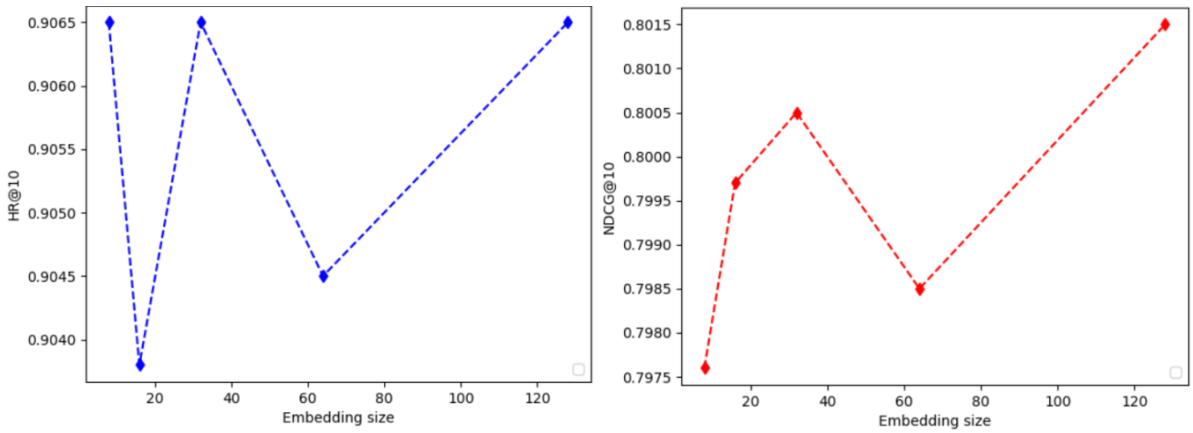

Figure 3.16 : Evaluation du modèle MLP sur Film trust selon les tailles d'embeddings.

D'après la figure 3.16 on note que le modèle donne de meilleurs résultats lorsque la taille d'embeddings est égale à 128.

Par la suite, nous avons procédé au brouilllage du dataset comme c'est illustré dans l'évaluation de HybMLP. La figure 3.17 montre les résultats de cette évaluation. (Voir l'annexe A4.17)

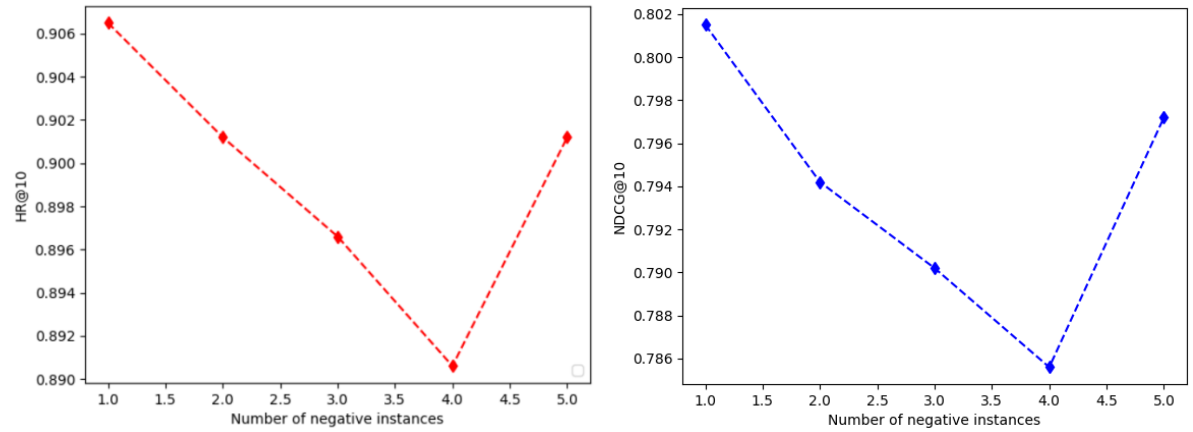

Figure 3.17 : Evaluation du modèle MLP sur Film trust selon le nombre d'instances négatives.

D'après les résultats illustrés par la figure 3.17, nous constatons que le modèle est plus performant quand le nombre d'instance négative est égal à 1 avec un NDCG de 0.8015 et un HR de 0.9065.

Les figures ci-dessous illustrent l'évaluation des Top-K listes d'items recommandés, avec K allant de 1 à 10 (pour plus de détails sur les valeurs des métriques voir les tableaux de l'annexe A4.18).

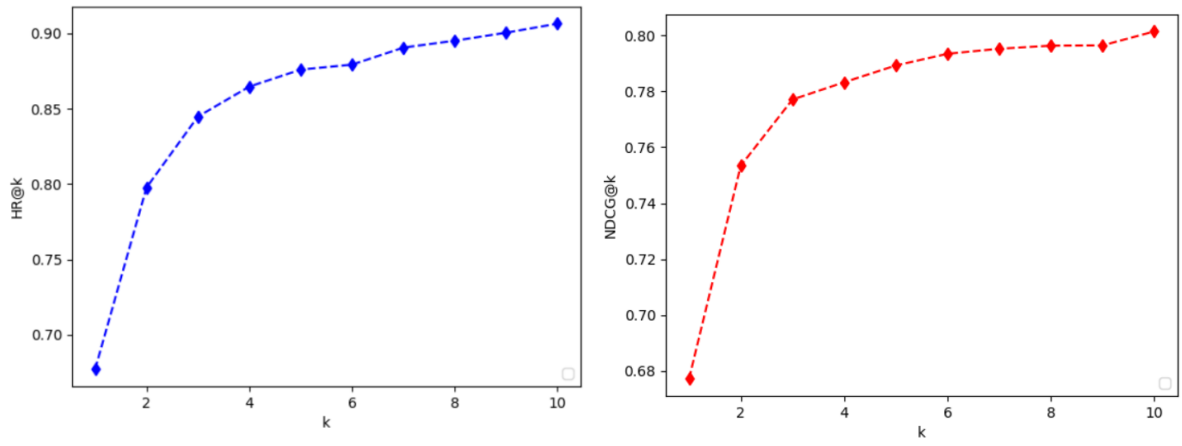

Figure 3.18 : Evaluation du Top-K de recommandation d'items avec K variant de 1 à 10 – Film Trust.

4.1.3 - Evaluation du filtrage social Soc

Ces évaluations ont été effectuées sur la meilleure structure de chaque modèle.

1) Film Trust

Nous avons d'abord procédé à l'évaluation du coté social dans Film Trust avec NDCG mais les valeurs de cette dernière ne changeait pas, alors on a donc opté pour l'écart afin de choisir le meilleur ratio entre la confiance implicite et explicite.

Les figures ci-dessous (figure 3.20 et 3.19) montrent que le meilleur ratio de la confiance implicite et explicite est 0.4. Nous allons donc donner plus d'importance à la confiance implicite afin de recommander des items pertinents à l'utilisateur à travers ses amis de confiance. (Voir tableau de l'annexe A4.20 et A4.19)

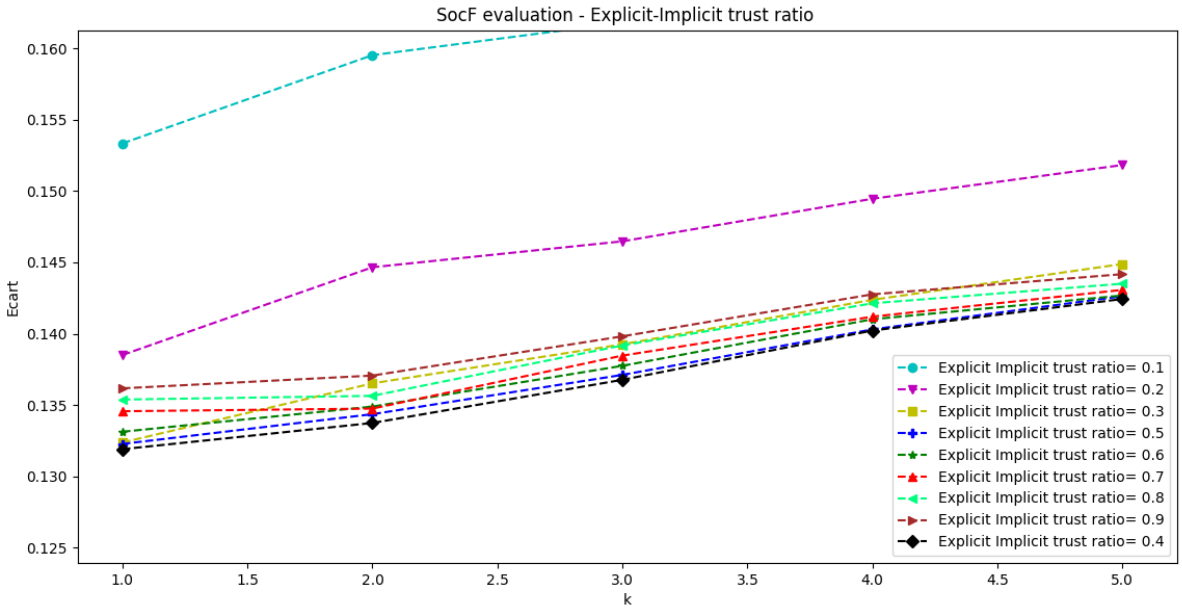

Figure 3.19 : Performances du modèle selon le nombre d'amis de confiance et ratio alpha - FilmTrust.

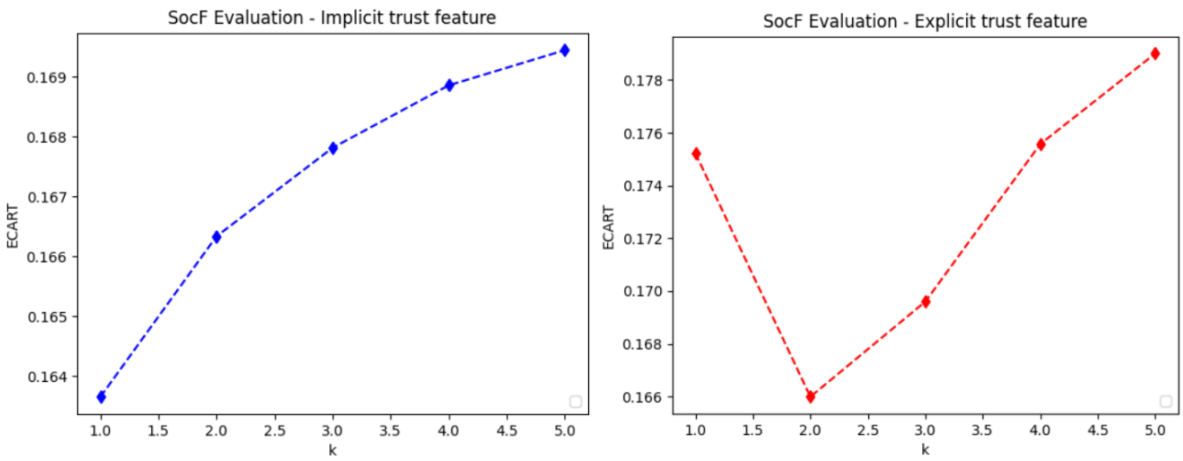

Figure 3.20 : Performances du modèle selon le nombre d'amis de confiance.

Remarque :

Nous remarquons que la confiance implicite donne de meilleurs résultats que la confiance explicite. Film Trust met à disposition des liens de confiances directes entre 522 utilisateurs seulement, ce qui prouve l'hypothèse que les utilisateurs ne partagent pas toujours leurs avis, ceci nous amènent à exploiter les données implicites

2) Yelp

Comme cité dans le chapitre précédent nous évaluons l'amitié en considérant deux variantes : la **distance cosine** et le **degré de Jaccard**. La figure 3.21 démontre que la **distance cosine** est plus performante que le **degré de Jaccard** ce qui prouve que le gout de l'utilisateur ne ressemble pas forcement à celui de ses amis.

Pour plus de détail sur les valeurs de métrique voir les tableaux sur l'annexe A4.10 et A4.11

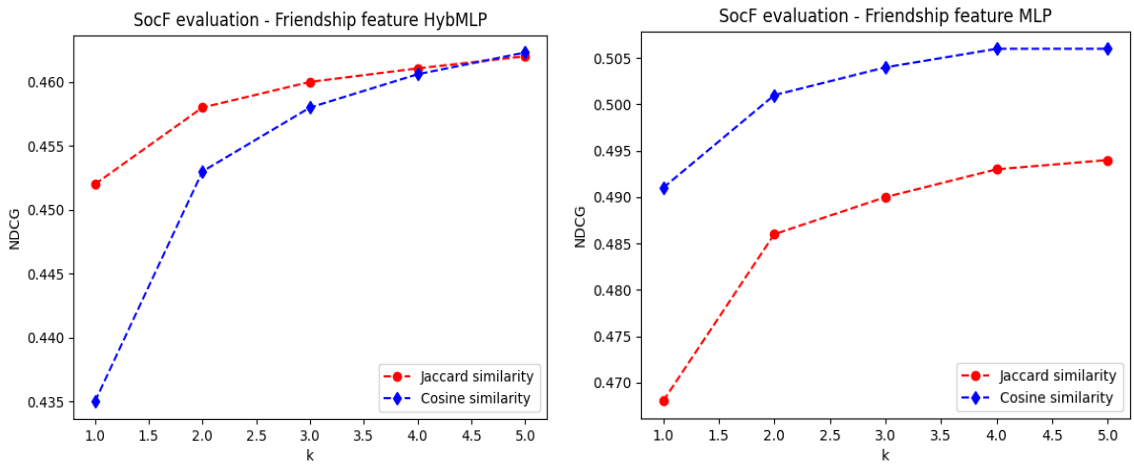

Figure 3.21 : Performances du modèle selon le nombre d'amis K.

Comme déjà constaté dans l'évaluation de HybMLP, MLP est plus performant dans Yelp donc donne de meilleurs résultats même avec l'aspect social avec un NDCG de 0.505 pour la distance cosine.

4.2 - l'apport de l'information sociale

1. Film Trust

La figure 3.22 montre que le coté social améliore les performances du modèle, où l'écart augmente si on donne plus d'importance au filtrage collaboratif. On note qu'Alpha est associé au filtrage collaboratif (Alpha =1 représente l'écart du FC).

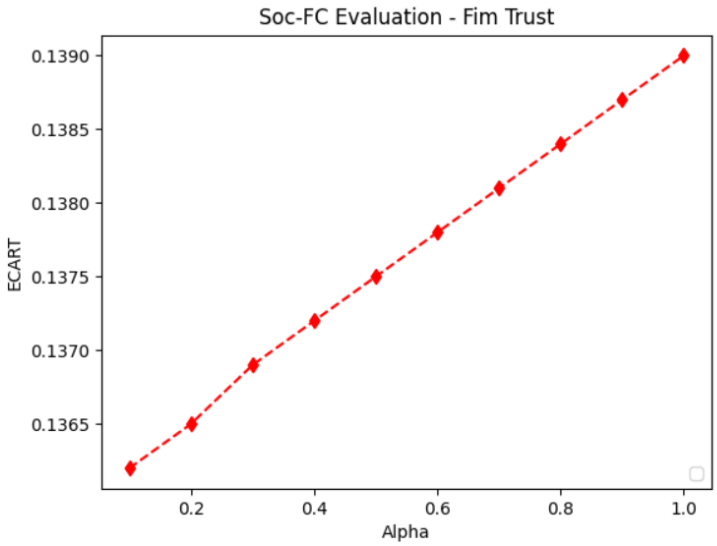

Figure 3.22 : Performances du modèle selon le ratio Alpha- FilmTrust.

D'après le graphe la meilleure valeur de Alpha est 0.1 d'où 0.1 pour le collaboratif et 0.9 pour le coté social ce qui montre l'efficacité de l'ajout de l'information social. Pour plus de détail sur les valeurs des métriques voir A4.21.

2. Yelp

Dans la base de données Yelp, le filtrage collaboratif seul donne de très bons résultats et même dans le cas où on ajoute l'information social, le filtrage collaboratif reste le plus performant, de même pour le filtrage collaboratif basé contenu.

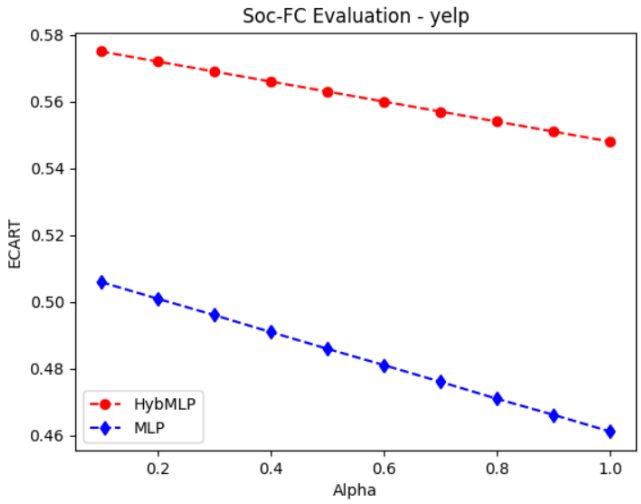

Figure 3.23 : Performances du modèle Soc-FC selon le ratio Alpha.

De même avec Film Trust, Alpha est associé au filtrage collaboratif (Alpha =1 représente l'écart du FC).

Nous présentons les détails des résultats obtenus dans l'annexe A4.12 et A4.13.

Remarque :

Après avoir évalué le coté social, on remarque qu'on peut diminuer le problème de démarrage à froid en utilisant Jaccard et la confiance explicite car ces deux dernières se base sur les liens directes exprimés par l'utilisateur et ne nécessite pas l'interaction de celui-ci avec les items de la base de données.

4.3 - Apport du contexte dans la recommandation

Afin d'étudier l'apport du contexte dans la recommandation, nous avons sélectionné le contexte selon l'item pertinent choisis de l'utilisateur.

- **Yelp**

Comme déjà expliqué dans le chapitre précédent, le contexte dans Yelp est la localisation de l'utilisateur où il doit spécifier la distance souhaité. Le tableau 3.5 présente les différentes valeurs de HIT@10 et NDCG@10 en changeant la valeur du contexte.

Yelp												
Distance	**20km**				**40km**				**60Km**			
Métrique	**HIT @10**	**NDCG @10**	**nbr pos**	**nbr neg**	**HIT @10**	**NDCG @10**	**nbr pos**	**nbr neg**	**HIT @10**	**NDCG @10**	**nbr pos**	**nbr neg**
MLP	**0.851**	**0.523**	23	77	0.785	0.479	32.43	67.57	0.782	0.4773	33.74	66.26
HybMLP	**0.848**	**0.512**	23	77	0.742	0.449	32.44	67.56	0.737	0.447	33.74	66.26

Tableau 3.4 : Evaluation des différents modèles sur Yelp avec différentes Distances.

Nous remarquons une amélioration de valeurs de Hit@10 et NDCG@10 par rapport au filtrage collaboratif qui était de 0.458 (HIT@10) et 0.755 (NDCG@10) sans contexte, ainsi que l'hybridation de celui-ci avec le filtrage basé contenu 0.407 comme valeur de HIT@10 et 0.683 pour le NDCG@10.

Remarque :

Dans le tableau 3.4, on remarque que quand le nombre d'items positifs au contexte augmente les valeurs de HIT et NDCG diminue. Cette diminution se justifie par l'ajout d'items qui seront peut-être plus proches que l'item pertinent.

- **Movie Lens 1M**

Comme le montre le tableau 3.5 en ajoutant le contexte on constate une amélioration des valeurs de HIT@10 et NDCG@10.

ML-1M					
Métrique @10		**HIT @10**	**NDCG @10**	**nbr pos**	**nbr neg**
Avec contexte	**MLP**	0.851	0.602	73	27
	HybMLP	0.846	0.605	73	27
sans contexte	**MLP**	0.807	0.557	/	/
	HybMLP	0.812	0.566	/	/

Tableau 3.5 : Evaluation des différents modèles sur ML-1M avec et sans contexte.

4.4 - Discussion générale

L'objectif principal des modèles proposés était d'évaluer l'apport de l'information sociale dans la recommandation. Les tests précédents ont démontré l'importance de ce type d'information et sa contribution à l'amélioration des prédictions, ce qui prouve qu'on peut diminué voir même corriger le problème de démarrage à froid en utilisant seulement l'information sociale.

Nous avons également étudié l'influence du contexte sur la recommandation, et les résultats d'évaluation obtenus montrent que l'ajout du contexte peut améliorer la précision de la prédiction. Avec l'augmentation du nombre de couches cachées du modèle, le Deep Learning a également prouvé son efficacité. Nous avons également remarqué l'importance de choisir les paramètres du modèle, qui contribuent beaucoup à l'amélioration ou au contraire à la réduction des prédictions, tel que la taille d'embeddings, le nombre d'instances négatives sur lesquelles le modèle doit s'entrainer.

5 - Conclusion

Dans ce chapitre, nous avons évalué notre approche proposé et prouvé son efficacité dans la prédiction en intégrant le contexte, comme on a prouvé l'importance de l'information sociale ainsi que sa contribution dans la correction du problème de démarrage à froid.

CONCLUSION GENERALE

Nous nous sommes intéressés dans ce travail à la proposition d'une approche de recommandation hybride sensible au contexte en s'inspirant du travail de Zeghoud et Kerboua [5] qui ont proposé une approche hybride basée sur une architecture avec les deux modèles : (1) la Factorisation Matricielle Généralisée, (Generalized Matrix Factorization - GMF) et le Perceptron Multicouches hybride (Hybrid Multi Layer Perceptron – HybMLP). Nous avons proposé une hybridation des algorithmes de filtrage collaboratif et basé contenu avec l'information sociale, selon une architecture deep learning avec les deux modèles : HybMLP (hybridation des algorithmes collaboratif et basé contenu) et SocHybMLP (complémente le modèle HybMLP par l'information sociale).

Afin d'évaluer notre approche, nous avons effectué plusieurs tests en utilisant trois bases de données : Film Trust, MoviLens 1M et Yelp. Les résultats ont montré une amélioration des performances de l'algorithme hybride SocHybmlp comparé à l'algorithme de filtrage collaboratif sous la même architecture avec les métriques HR et NDCG ainsi qu'une amélioration de prédiction en intégrant l'information contextuelle.

Comme perspectives à ce travail, nous envisageons d'abord d'évaluer nos méthodes en utilisant d'autres bases de données ; évaluer notre système en conditions réelles pour recueillir les commentaires des utilisateurs sur les recommandations effectuées ; et enfin, utiliser d'autres architectures d'apprentissage profond comme un réseau de neurones convolutifs pour la combinaison des deux modèles présentés précédemment.

REFERENCES

[1] Poonam, B., Thorat, R.,Goudar, M., Sunita, B. Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System, International Journal of Computer Applications (0975 – 8887) Volume 110 – No. 4, January 2015

[2] Rui, C., Qingyi, H., Yan-Shuo, C., Bo, W., Lei, Z., Xiangjie, K. A Survey of Collaborative Filtering-Based Recommender Systems : From Traditional Methods to Hybrid Methods Based on Social Networks.IEEE November 19, 2018.

[3] Lipi, S., Hetal, G., Prem, B. Survey on Recommendation System. International Journal of Computer Applications (0975 – 8887) Volume 137 – No.7, March 2016.

[4] Gasmi.W. Le filtrage basé sur le contenu pour la recommandation de cours (FCRC). Université de montréal décembre 2011.

[5] Zeghoud, S., Kerboua, I.L. Système de recommandation hybride basé sur l'apprentissage profond Mémoire-de fin d'études de licence, Option ISIL, Dép. Informatique USTHB encadré par Dr. L.Berkani, Juillet 2019

[6] Xiaoyua, S., Taghi, M.K. Review Article A Survey of Collaborative Filtering Techniques .Department of Computer Science and Engineering, Florida Atlantic University, USA. 3 August 2009.

[7] Dahmani, Rahal. Proposition d'un système de recommandation par filtrage collaboratif basé sur les folksonomies. Université saad dahleb octobre 2019.

[8] Mesroua, N.A., Seghiri, D. Application et exploration de l'approche de découpage des systèmes de recommandations pré-contextuelle. Présenté le 27 Juin 2019.

[9] Surasbh, K., Sunil, F.R, Context Aware Recommendation Systems : A review of the state of the art techniques. Computer sience Review (2020).

[10] Djeffal, A. Utilisation des méthodes Support Vector Machine (SVM) dans l'analyse des bases de données. Thèse, encadré par : Dr. Mohamed Chaouki babahenini Co-Encadrant : Pr. Abdelmalik taleb ahmed. Université Mohamed Khider - Biskra (2012).

[11] https://dataanalyticspost.com/Lexique/k-nearest-neighbours/

[12]Boutaba, R., Salahuddin, M.A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano F., Caicedo, O.M. A comprehensive survey on machine learning for networking : evolution, applications and research opportunities. Journal of Internet Services and Applications (2018).

[13] https://www.scirp.org/html/1-9601348_73781.htm

[14] Kajaree, D., Rabi, N.B. A Survey on Machine Learning : Concept, Algorithms and Applications. International Journal of Innovative Research in Computer and Communication Engineering, February 2017.

[15] https://deeplylearning.fr/tag/neurone/

[16] https://alphabold.com/neural-networks-and-deep-learning-an-overview/

[17] *https://medium.com/data-science-bootcamp/multilayer-perceptron-mlp-vs-convolutional-neural-network-in-deep-learning-c890f487a8f1*

[18] Guo, G., Zhang, J., Thalmann, D., Basu, A., Yorke, N. From Ratings to Trust: an Empirical Study of Implicit Trust in Recommender Systems 2014

[19] Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I. Recommender Systems with Social Regularization (2011).

[20] https://deeplylearning.fr/cours-theoriques-deep-learning/fonctions-de-perte-cout-loss-function/

ANNEXE

A.1 – Développement du système de recommandation avec Python

- Pseudo code correspondant au calcul de similarité en utilisant la formule de jaccard :

Algorithme A1.1 : Calcul de similarité de Jaccard

INPUT : Fichier csv Friendship, N : nombre d'utilisateurs ;

OUTPUT : Data [] : un fichier pour stocker les utilisateurs avec leurs degrés d'amitiés.

Début

Pour i de 1 à N
Faire
NB_Friends1←nombre d'amis de l'utilisateur i ;
FriendsList1← la liste d'amis de l'utilisateur i ;
Pour j de 1 à Taille(FriendsList1)
Faire
ID_Friend← l'id de l'utilisateur j dans la liste d'amis ;
NB_Friends2← nombre d'amis de l'utilisateur j ;
NBC← Nombre d'amis en commun ;
Sim ←NBC/ (NB_Friends2+ NB_Friends1-NBC) ;
Ajouter (i, ID_Friend, Sim) ;
Fait ;
Fait ;

Fin

- Pseudo code correspondant au calcul de la confiance explicite :

Algorithme A1.2 : Trust Explicite

INPUT : Fichier csv trustee, N : nombre d'utilisateurs ;

OUTPUT : ex_trust [] : pour stocker les degrés de confiances directe pour chaque utilisateur.

Début

Pour i de 1 à N
Faire
U← id de l'utilisateur i.
User_rows ← sélectionne les lignes où figure l'utilisateur U.
Nb_arcs_sortant ← Taille(User_rows).
Pour j de 1 à Nb_arcs_sortant
Faire

Trustee ← l'id de l'utilisateur dans la ligne j.

Nb_arcs_sortantT← le nombre de fois où l'utilisateur Trustee apparait Dans la Première colonne (User_id).

Nb_arcs_entrantT ← le nombre de fois où l'utilisateur Trustee apparait Dans la Deuxième colonne (Trustee).

Deg_Trust←racine(Nb_arcs_entrantT/(Nb_arcs_sortantT+Nb_arcs_sortant) ;

Si Deg_Trust, U, Trustee Not IN Data alors Ajouter (Deg_Trust, U,Trustee).

Fait

Fait ;

Sauvegarder ex_trust dans un fichier csv.

Fin

- Pseudo code correspondant au calcul de la confiance implicite :

Algorithme A1.3 : **Trust Implicite**

INPUT : Fichier csv user, Ratings, nombre d'utilisateurs ;

OUTPUT : imp_trust[] : pour stocker les degrés de confiance indirecte pour chaque utilisateurs.

Début

Pour i de 1 à N

Faire

U← id de l'utilisateur i.

Avrg_Rating1= la moyenne des notations de l'utilisateur U.

Pour j de 1 à N

Faire

U2← id de l'utilisateur j.

Avrg_Rating2= la moyenne des notations de l'utilisateur U2.

CommonR= sélectionne pairs (items, R) des items qui ont été évalué de la Même manière.

Sum1=0 ;

It = Nan ; // vide

NB←nombre _d'items_évalué_en_commun ;

Pour k de 1 à NB

Faire

RatingU1= la note de l'utilisateur U1 pour l'item K.

RatingU2= la note de l'utilisateur U2 pour l'item K.

P= Avrg_Rating1+ (RatingU2- Avrg_Rating2) ;

Sum1=Sum1+ (1-valeur absolue (P- RatingU1) /4);
Si NB>0 alors It = Sum1/NB ; FSI
Ajouter (U1, U2, It) ;
Fait ;
Fait ;
Fait ;
Sauvegarder imp_trust dans un fichier csv.

Fin

- **Pseudo code correspondant à l'intégration du contexte dans le processus de recommandation :**

Algorithme A1.4 : Recommandation avec Contexte

INPUT : Prediction1 : résultat de prédiction d'interaction du module HybMLP.
Predition2 : résultat de prédiction d'interaction du module SocHybMLP.
OUTPUT : DataResult : Liste des recommandations.

Début

Pour i de 1 à taille (Prediction1)
Faire
Prediction1 [i]= Prediction1 [i]*alpha ;
Fait ;
Pour i de 1 à taille (Prediction2)
Faire
Prediction2 [i]= Prediction2 [i]*(alpha-1) ;
Fait ;
Pour i de 1 à taille (Prediction2)
Faire
Item_id= id de l'item i dans prediction1 ;
Val = valeur de prédiction d'item_id dans Prediction1 + valeur de prédiction d'item_id dans Prediction2 ;
Ajouter (item_id, val) ;
Fait ;
Localisation_user // on a travaillé avec la latitude et longitude
Distance // spécifier par le user la distance souhaité

//les items sont des restaurants

Pour tout item dans DataResult

Faire

Localisation_item = la latitude et longitude de l'item i ;

Calcultaed_distance = la distance entre user et l'item i ;

If (Calcultaed_distance<distance) result.ajouter(item_id) ;

Fait ;

Fin

A.2 – Interface Graphique

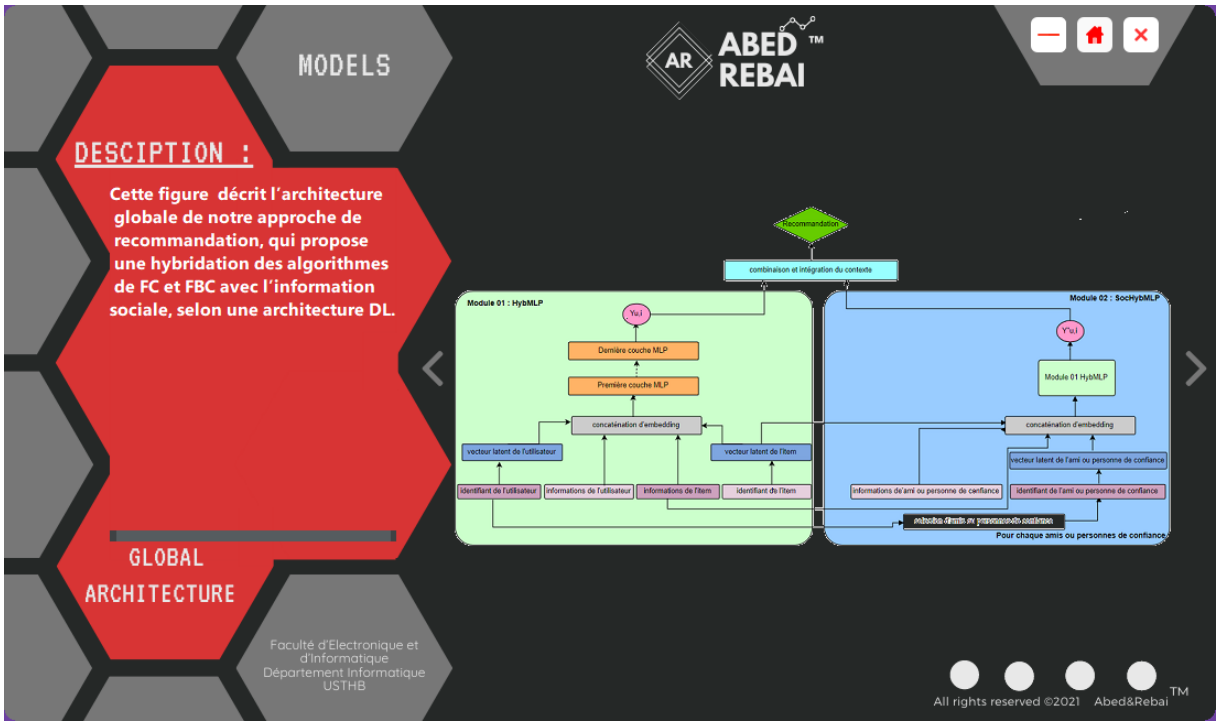

Figure A2.1 : Implémentation de notre système de recommandation – Fenêtre Modèles

Figure A2.2 : Implémentation de notre système de recommandation – Fenêtre Datasets

Les numérotations 1, 2, 3 sur la figure A2.2 indiquent que l'utilisateur doit d'abord choisir un dataset puis préciser le nombre d'instances négatives après sélectionner le chemin où il souhaite sauvegarder ses résultats.

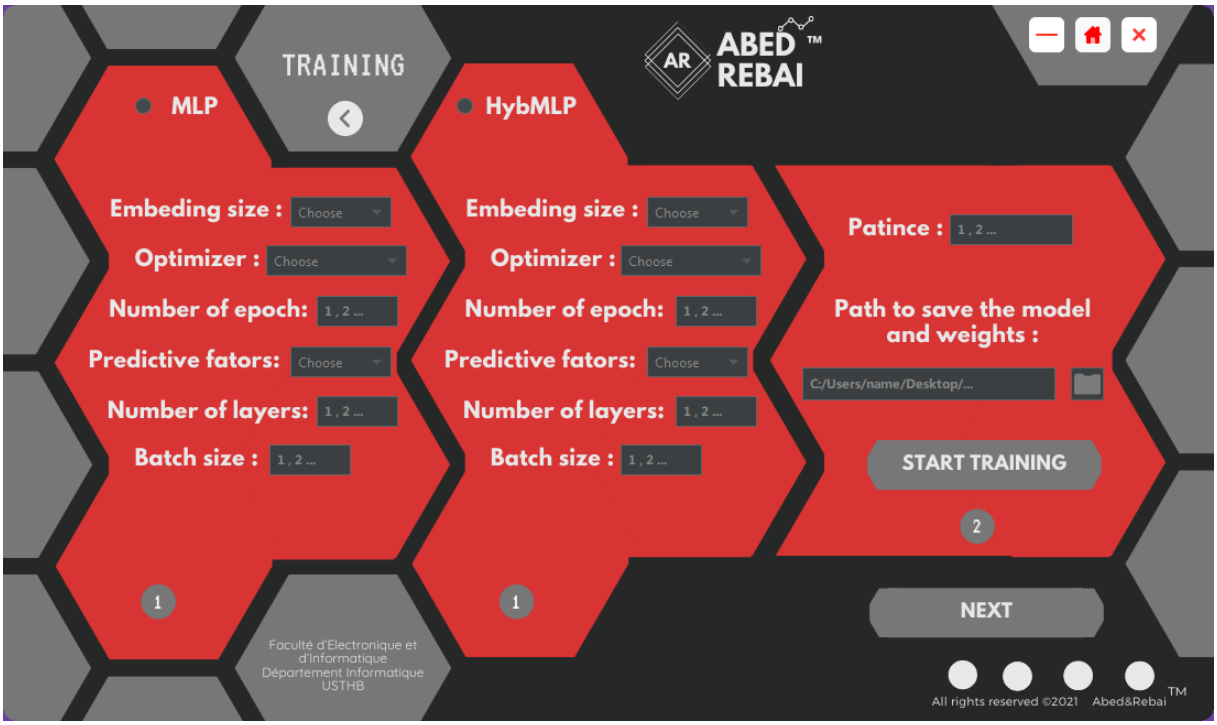

Figure A2.3 : Implémentation de notre système de recommandation – Fenêtre Training.

L'utilisateur doit remplir les informations du modèle souhaité (HybMLP ou MLP) pour pouvoir lancer son entraînement.

Le bouton « NEXT » sur la figure A2.3 permet à l'utilisateur de passer à la fenêtre suivante (Figure A.2.4) :

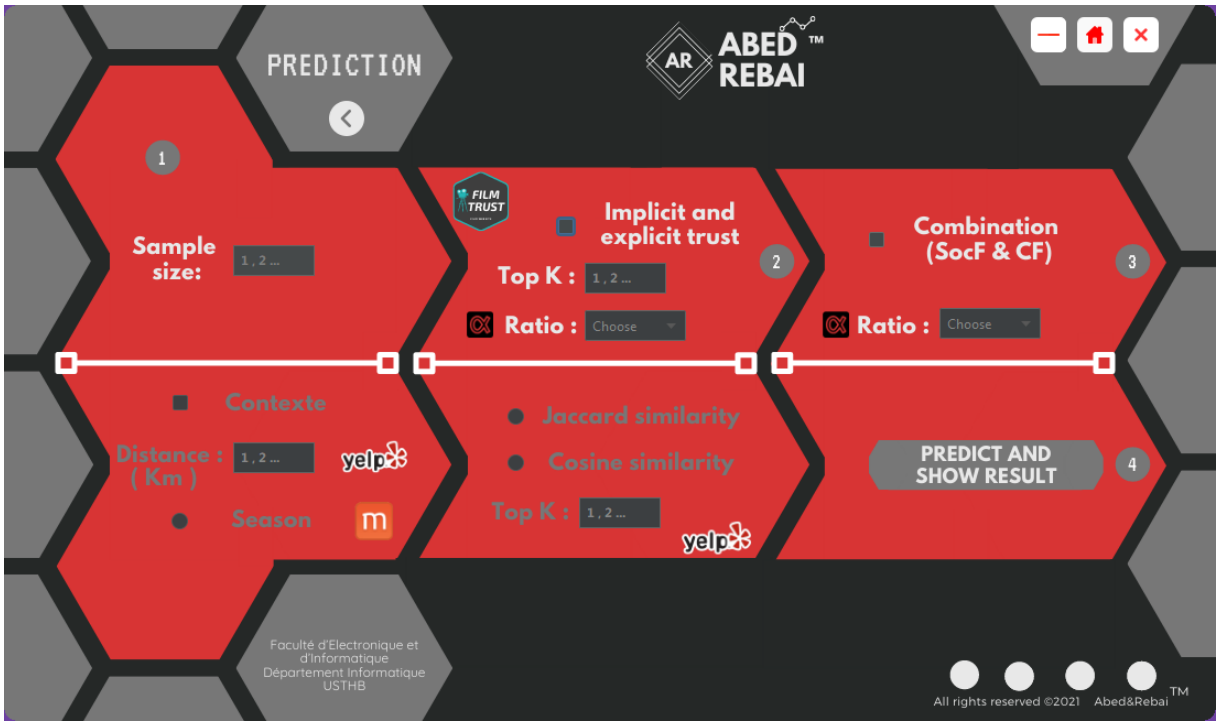

Figure A2.4 : Implémentation de notre système de recommandation – Fenêtre Prédiction.

Cette fenêtre présente le coté social dans Film trust d'où elle permet de personnaliser ses paramètres. L'utilisateur peut varier le ratio implicit/explicit trust de 0 à 1 d'où 0 signifie implicit trust et 1 explicit trust quant au ratio Fsoc et FC il varie de 0.1 à 1.

A3 – Développement du système de recommandation avec Python

Dans cette section nous expliquons quelques détails sur le développement du système de recommandation sous Python.

A3.1 – Prétraitement des données

1- Lecture des fichiers : les fichiers doivent être en format csv avec un séparateur entre les colonnes. La lecture se fait avec la fonction read_csv() de Pandas.
2- Transformation des données : le tableau suivant (table A1.1) décrit les fonctions utilisées.
3- Sauvegarde des fichiers d'entraînement et de test.

Tableau A3.1 : Fonctions de transformation des données.

Dataset	*Bibliothèques utilisées*	*Fonctions*	*Description*	*Complexité*	
MovieLens	Pandas, Numpy, datetime	*Timestamp_to_season()*	Permet de transformer Timestamp sous format date puis en saison.	$O(n)$	n : le nombre d'interactions du datasets (reviews)
	Pandas, Numpy	Item_season()	Elle va traiter le résultat issu de la fonction « Timestamp_to_season() »,en calculant le nombre de fois ou un utilisateur a interagit avec un item et de retenir que la saison ou y'a eu le plus d'interaction, pour tous les items.	$O((k+4)*(n+4))$	n : le nombre d'items. k : le nombre d'interactions associé à un item donné. 4 : nombre de saisons
	Numpy, Pandas	Genre_item()	Transforme les genres des items (films) donnés (action, comedy…) en vecteurs booléens.	$\boldsymbol{O(y*k*n)}$	**n** : nombre d'items. **k** : taille des données relatives à chaque item (k=18 pour ce dataset). **y** : taille de la chaine des genres d'items.
	Pandas	Context_season()	Retient seulement les items qui sont de la même saison que l'item pertinent.	$O(n+1)$	n : taille de la liste des items recommandés pour un utilisateur.

Yelp	Pandas, geopy	Context_dis()	Calcule la distance entre l'item pertient et les autres items et les selectionne selon un seuil donné.	$O(n+1)$	n : taille de la liste des items recommandés pour un utilisateur.
Film Trust	Numpy, Pandas, Graph	Exp_prop()	Permet de calculer la propagation de la confiance explicite à l'aide d'un graphe orienté.	O(2*k+n^2*(y+(z*(z-1))))	k : le nombre de liens de confiance explicite. N : taille de la matrice de confiance (graphe). Y :l complexité du parcours des chemins possibles (la longueur ne dépasse pas 3) entre deux utilisateurs ca dépends de nombre de liens exprimée par un utilisateur. Z : le nombre de chemins entre utilisateurs u1et u2.

	Pandas	Imp_exp_Trust_Ratio()	Permet d'affecter a la confiance explicite un poids α et a l'implicite β de sorte que α+ β=1	$O(n)$	N : nombre de liens de confiance explicite exprimé par un utilisateur choisi.

A3.2 – Génération du jeu d'entraînement et de test

1- Lecture des données : lecture des fichiers des jeux d'entraînement et de test.

2- Génération des fichiers : génère les fichiers d'entraînement et de test avec les fonctions décrites dans le tableau suivant

Tableau A3.2 : Fonctions de génération des jeux d'entraînement et jeux de données.

Bibliothèques utilisées	**Fonctions**	**Description**	**Complixité**	
Numpy, Pandas	Datasetsplit ()	Partage le dataset en trainset et testset tel que le testset contient instance positive par utilisateur. Supprime les interactions choisies pour le test du trainset. Ajoute les instances négatives au testset ainsi que le trainset.	$\boldsymbol{O(u*(n+y)+x(n-u))}$	**u** : nombre d'utilisateurs du dataset. **n** : nombre d'interactions du dataset. **x** : nombre de tentatives afin de trouver le nombre d'instances négatives par instance positive souhaité dans le trainset (number of negatives). **y** : nombre de tentatives pour trouver une instance positive et 99 autres négatives pour le testset.

A3.3 – Construction, entraînement des modèles, évaluation

1- Construction des modèles : les modèles MLP, HybMLP sont construits à partir de Keras.
2- Entraînement : entraînement d'un modèle donné : utilise la fonction fit() de Keras afin d'effectuer une itération sur tout le jeu de données.
3- Evaluation : évalue le modèle après chaque itération et sauvegarde le modèle de la meilleure itération ainsi que les scores des métriques de toutes les itérations. Les fonctions d'évaluation sont décrites dans le tableau A1.3

Tableau A3.3 : Fonctions d'évaluation de modele.

Biblioth-èques utilisées	***Fonctions***	***Description***	***Complexité***	
Numpy, math	*evaluate_()*	Recherche l'item pertinent dans la liste des prédictions	$O(k)$	***k :*** taille de la lise des items recommandés.
Numpy, Pandas	*evaluate_model()*	-Prédit l'interaction des pair utilisateur-item présents dans le *testset* -Trie le *testset* selon la prédiction du modèle ($\in [0,1]$) dans l'ordre décroissant. -Sélectionne pour chaque utilisateur les k items les plus recommandés. -Recherche l'item pertinent dans la liste des prédictions en utilisant evaluate_by_users().	$(n * \log(n) + u * (n + k) + X)$	***n :*** taille des données du test. ***u :*** nombre d'utilisateurs du *testset.* ***k :*** taille de la liste d'items recommandés. ***X :*** complexité de la fonction de prédiction, dépend du modèle choisi ainsi que de sa structure.

A.4 – Evaluations et tests effectués

1) Movie lens 1m

Tableau A4.1 : Evaluation de HybMLP selon différents nombres de couches.

Nombre de couches		0	1	2	3	4	5
Modèle	Métrique @10						
MLP	HR	0.561	0.735	0.741	0.763	0.765	0.779
	NDCG	0.332	0.468	0.482	0.495	0.499	0.513
HybMLP	HR	0.561	0.750	0.759	0.771	0.779	0.789
	NDCG	0.334	0.479	0.489	0.504	0.508	0.524

Tableau A4.2 : Evaluation des modèles selon différentes tailles d'embedding.

TE	8		16		32		64		128	
Métrique @10	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
MLP	0.754	0.485	0.768	0.499	0.777	0.508	0.778	0.513	0.773	0.511
HybMLP	**0.761**	**0.494**	**0.771**	**0.513**	**0.789**	**0.522**	**0.789**	**0.524**	**0.779**	**0.520**

Tableau A4.3 : Evaluation selon le nombre de nœuds de la dernière couche (predictive factors).

Predictive factors	*HybMLP*		*MLP*	
Métrique @10	*HR*	*NDCG*	*HR*	*NDCG*
8	0.779	0.517	0.777	0.509
16	0.789	0.524	0.778	0.513
32	0.789	0.526	0.780	0.514
64	0.795	0.529	0.781	0.518
128	**0.797**	**0.532**	**0.784**	**0.522**

Tableau A4.4 : Evaluation des différents modèles selon le nombre d'instances négatives par instance positive.

Nombre d'instances négatives	1		2		3		4		5	
Métrique @10	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
MLP	0.784	0.522	0.793	0.539	0.806	0.557	0.794	0.553	0.807	0.556
HybMLP	**0.797**	**0.532**	**0.808**	**0.55**	**0.817**	**0.566**	**0.815**	**0.564**	**0.817**	**0.565**

Tableau A4.5 : Evaluation du Top K de recommandation d'items.

K		1	2	3	4	5	6	7	8	9	10
Modèle	**Métrique @k**										
MLP	**HR**	0.335	0.48	0.568	0.636	0.684	0.723	0.743	0.768	0.788	0.807
	NDCG	0.335	0.426	0.474	0.5	0.520	0.532	0.538	0.548	0.552	0.557
MLP	**HR**	**0.342**	**0.493**	**0.580**	**0.644**	**0.692**	**0.727**	**0.755**	**0.779**	**0.798**	**0.812**
	NDCG	**0.342**	**0.433**	**0.482**	**0.512**	**0.522**	**0.539**	**0.550**	**0.557**	**0.562**	**0.566**

2) Yelp

Tableau A4.6 : Evaluation de HybMLP selon différents nombres de couches.

Nombre de couches		**0**	**1**	**2**	**3**	**4**	**5**
Modèle	***Métrique @10***						
MLP	***HR***	0.413	0.653	0.669	0.665	0.664	**0.666**
	NDCG	0.230	0.381	0.387	0.382	0.384	**0.386**
HybMLP	***HR***	0.411	0.553	0.687	0.652	0.651	**0.689**
	NDCG	0.227	0.311	0.4	0.380	0.381	**0.402**

Tableau A4.7 : Evaluation des modèles selon différentes tailles d'embedding.

TE	**8**		**16**		**32**		**64**		**128**	
Métrique @10	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
MLP	0.666	0.386	0.666	0.388	0.667	0.39	0.668	0.391	**0.676**	**0.394**
HybMLP	**0.689**	**0.402**	**0.569**	**0.317**	**0.641**	**0.362**	**0.645**	**0.368**	**0.622**	**0.350**

Tableau A4.8 : Evaluation des différents modèles selon le nombre d'instances négatives par instance positive.

Nombre d'instances négatives	**1**		**2**		**3**		**4**		**5**	
Métrique @10	**HR**	**NDCG**	**HR**	**NDCG**	**HR**	**NDCG**	**HR**	**NDCG**	**HR**	**NDCG**
MLP	0.676	0.394	0.711	0.423	0.746	0.445	0.753	0.456	**0.755**	**0.458**
HybMLP	0.651	0.366	0.662	0.378	0.671	0.386	0.679	0.403	**0.683**	**0.407**

Tableau A4.9 : Evaluation du Top K de recommandation d'items.

K		1	2	3	4	5	6	7	8	9	10
Modèle	**Métrique @k**										
MLP	**HR**	0.219	0.357	0.456	0.534	0.582	0.631	0.675	0.711	0.738	0.755
	NDCG	0.219	0.308	0.352	0.383	0.406	0.424	0.436	0.442	0.451	0.456
HybMLP	**HR**	0.203	0.326	0.422	0.487	0.543	0.584	0.621	0.663	0.673	0.683
	NDCG	0.203	0.281	0.329	0.358	0.380	0.391	0.395	0.400	0.404	0.407

Tableau A4.10 : Evaluation du Friendship avec Jaccard similarity.

Yelp Friendship					
Nombre TOP amis	**1**	**2**	**3**	**4**	**5**
Métrique	**NDCG**	**NDCG**	**NDCG**	**NDCG**	**NDCG**
HybMLP	0.452	0.458	0.460	0.46106	0.462
MLP	0.468	0.486	0.49	0.493	0.494

Tableau A4.11 : Evaluation du Friendship avec cosine similarity.

Yelp					
Nombre TOP amis	**1**	**2**	**3**	**4**	**5**
Métrique	**NDCG**	**NDCG**	**NDCG**	**NDCG**	**NDCG**
HybMLP	0.435	0.453	0.458	0.4606	0.4623
MLP	0.491	0.501	0.504	0.506	0.506

Tableau A4.12 : Evaluation du modele selon le ratio alpha –HybMLP.

Alpha	0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
NDCG	0,320	0.407	0.407	0.407	0.407	0.407	0.407	0.407	0.407	0.407	0.407
HR	0.536	0.683	0.683	0.683	0.683	0.683	0.683	0.683	0.683	0.683	0.683
ECART	0.494	0.575	0.572	0.569	0.566	0.563	0.560	0.557	0.554	0.551	0.548
user number	4543	5436	5436	5436	5436	5436	5436	5436	5436	5436	5436

Tableau A4.13 : Evaluation du modele selon le ratio alpha –MLP.

Alpha	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
NDCG	**0.458**	**0.458**	**0.458**	**0.458**	**0.458**	**0.458**	**0.458**	**0.458**	**0.458**	**0.458**
HR	**0.755**	**0.755**	**0.755**	**0.755**	**0.755**	**0.755**	**0.755**	**0.755**	**0.755**	**0.755**
ECART	**0.506**	**0.501**	**0.4961**	**0.491**	**0.486**	**0.4811**	**0.4761**	**0.471**	**0.4662**	**0.4612**
user number	**5436**	**5436**	**5436**	**5436**	**5436**	**5436**	**5436**	**5436**	**5436**	**5436**

3) Film Trust

Tableau A4.14 : Evaluation de MLP selon différents nombres de couches.

Nombre de couches		**0**	**1**	**2**	**3**	**4**	**5**
Modèle	***Métrique @10***						
MLP	***HR***	**0.9065**	0.9019	0.9045	0.9045	0.8912	0.9005
	NDCG	**0.8015**	0.7925	0.7893	0.7902	0.7814	7809

Tableau A4.15 : Evaluation de MLP selon différentes tailles d'embedding.

TE	**8**		**16**		**32**		**64**		**128**	
Métrique @10	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
MLP	0.9065	0.7976	0.9038	0.7997	0.9065	0.8005	0.9045	0.7985	**0.9065**	**0.8015**

TableauA4.16 : Evaluation de MLP selon le nombre de nœuds de la dernière couche (Predictive factors)

Predictive factors	***MLP***	
Métrique @10	***HR***	***NDCG***
8	0.8939	0.7813
16	0.8939	0.7824
32	0.8939	0.7839
64	0.9032	0.7964
128	**0.9065**	**0.8015**

Tableau A4.17 : Evaluation de MLP selon le nombre d'instances négatives par instance positive.

Nombre d'instances négatives	**1**		**2**		**3**		**4**		**5**	
Métrique @10	**HR**	**NDCG**	**HR**	**NDCG**	**HR**	**NDCG**	**HR**	**NDCG**	**HR**	**NDCG**
MLP	**0.9065**	**0.8015**	0.9012	0.7942	0.8966	0.7902	0.8906	0.7856	0.9012	0.7972

Tableau A4.18 : Evaluation du Top K de recommandation d'items.

k		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Modèle	**Métrique @k**										
MLP	**HR**	0.677	0.798	0.845	0.865	0.876	0.879	0.891	0.895	0.901	0.907
	NDCG	0.68	0.753	0.777	0.783	0.789	0.794	0.795	0.796	0.797	0.802

Tableau A4.19 : Evaluation du modèle SocMLP sur FilmTrust avec différents K (nombre d'amis de confiance) et ratio Alpha.

Film Trust Implicite-Explicite tust					
Nombre TOP amis	**1**	**2**	**3**	**4**	**5**
Métrique / **Alpha**	**Ecart**	**Ecart**	**Ecart**	**Ecart**	**Ecart**
0,1	0.153334	0.159518	0.161793	0.163990	0.165829
0,2	0.138508	0.144656	0.146474	0.149455	0.151820
0,3	0.132380	0.136523	0.139256	0.142375	0.144885
0,4	0.131910	0.133746	0.136771	0.140221	0.142433
0,5	0.132278	0.134362	0.137116	0.140302	0.142624
0,6	0.133132	0.134894	0.137751	0.141012	0.142680
0,7	0.134569	0.134754	0.138467	0.141193	0.143071
0,8	0.135387	0.135646	0.139180	0.142129	0.143505
0,9	0.136173	0.137067	0.139817	0.142759	0.144174

Tableau A4.20 : Evaluation du modèle SocMLP sur FilmTrust avec différents K(nombre d'amis de confiance).

Film Trust Implicite-Explicite tust					
Nombre TOP amis	**1**	**2**	**3**	**4**	**5**
Métrique / **Trust**	**Ecart**	**Ecart**	**Ecart**	**Ecart**	**Ecart**
Implicite	0.1636509	0.16633277422	0.16781201148	0.16885498822	0.16944237727
explicite	0.1752207	0.16599428756	0.16960267115	0.17557395896	0.179005

Tableau A4.21 : Evaluation du modèle selon le ratio Alpha.

Alpha	**0**	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
NDCG	0.797	0.8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
HR	0.901	0.904	0.904	0.904	0.904	0.904	0.904	0.904	0.904	0.904	0.904
ECART	0.1319	0.1362	0.1365	0.1369	0.1372	0.1375	0.1378	0.1381	0.1384	0.1387	0.139
user number	1501	1508	1508	1508	1508	1508	1508	1508	1508	1508	1508