République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Chadli Bendjedid
Faculté des Sciences et de la Technologie
Département d'Informatique

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد
كلية العلوم و التكنولوجيا
قسم الاعلام الالي

جامعة الشاذلي بن جديد
UNIVERSITE CHADLI BENDJEDID

MEMOIRE

Présenté par

MERZOUGUI Dalel

Pour l'obtention de diplôme de

MASTER

Filière : Informatique

Spécialité : Systèmes Informatiques Intelligents

Thème

Un Système de Recommandation basé sur l'Apprentissage Profond

Soutenue le : / /2020 Devant le Jury composé de :

Qualité	Nom et Prénom	Grade	Université
Président	Mr. BENTRAD Sassi	MCB	Chadli Bendjedid El-Tarf
Rapporteur	**Mme. GASMI Ibtissem**	**MCB**	**Chadli Bendjedid El-Tarf**
Examinateur	Mme. MAATALLAH Majda	MCB	Chadli Bendjedid El-Tarf

Année Universitaire : 2019 -2020

Remerciements

Avant tout, nous remercions le dieu de nous avoir donné le courage et la force pour accomplir ce modeste travail

Nous tenons à remercier Docteur GASMI Ibtissem pour nous avoir suivies, encouragées et aidées tout au long de ce travail.

Sa rigueur intellectuelle, Sa vision pragmatique de la recherche, ainsi que ses qualités personnelles d'écoute et de soutien nous ont aidées à accomplir avec confiance notre travail.

Nous remercions également, tous les enseignants qui nous ont fait honneur par leur présence dans le jury.

A tous ceux qui de loin ou de prés participé à la réalisation de ce mémoire

Dédicace

Je dédie ce modeste travail

A ma chèremère qui n'a jamais cessé de m'encourager et de m'apporter

réconfort et aide morale indispensables pendant les difficiles moments

A mon cher père

A mes deux chers frères

A toutes la famille

A tous mescollègues de travail de la faculté des sciences et de la technologie

Je vous aime tous

A ma promo 2019/2020

A tous ceux qui m'ont donné la force de continuer …

Table des matières

Chapitre I : Systèmes de Recommandation et apprentissage profond

Chapitre II : Conception

Chapitre III : Implémentation

Liste des figures

Chapitre I : Systèmes de Recommandation et apprentissage profond

Chapitre II : Conception

Chapitre III : Implémentation

Liste des Tableaux

Chapitre I : Systèmes de Recommandation et apprentissage profond

Chapitre II : Conception

Chapitre III : Implémentation

Liste des acronymes

Adam : Adaptive Moment Estimation

ALS : Alternating Least Squares

ART : Adaptive Resonance Theory

CNN : Convolutional neural network (Réseau de neurones convolutionnels)

DL : Deeplearning

FC : Le filtrage collaboratif

HMAE : High Mean Absolute Error

IA : Intelligence Artificielle

LDA : Latent Dirichlet Allocation

MAE : Mean Absolute Error

MLP : Perceptron multicouche (Multilayer Perceptron)

MSE : Erreur quadratique moyenne

NMAE : L'erreur absolue moyenne normalisée (Normalized Mean Absolute Error)

RNN : Réseau de neurones récurrents (Recurrent Neural Network)

SDAE : Stacked Denoising Autoencoder

SPPMI :shifted positive pointwise mutual information)

SR : System de Recommandation

TF-IDF : TermFrequency-Inverse Document Frequency

Introduction Générale

Depuis une dizaine d'années, les modèles neuronaux prennent une grande importancedans de nombreuses tâches complexe grâce à des avancées algorithmiques et à la mise à disposition d'outils de calcul puissants comme les processeurs graphiques [1]. Ils connaissent aujourd'hui un grand succès dans de nombreux domaines comme la modélisation, la traduction automatique, la reconnaissance et la prédiction [2].

D'autre part, les systèmes de recommandation sont des outils très puissants permettant la suggestion de services et d'objets qui s'adaptent aux profils des utilisateurs. Cependant, un nombre considérable de recherches faites sur les systèmes de recommandation se focalisent seulement sur la recommandation des items les plus pertinents pour les utilisateurs sans prendre en compte les informations contextuelles (la localisation, le temps, etc.) [3].

Le contexte est une notion très large et vague qui peut varier selon le domaine d'application [4]. Il doit d'abord être acquis en utilisant éventuellement des capteurs ; puis interprété afin d'obtenir des informations utiles. Ensuite, il doit être modélisé pour simplifier son utilisation. En effet, il est difficile de modéliser le contexte à partir des informations contextuelles non observables comme l'intention des utilisateurs. Une alternative pour résoudre ce problème est la modélisation des thèmes à partir des données textuelles qui sont une source d'information particulièrement riche et expressive. Ainsi, nous proposons un système de recommandation sensible au contexte, qui se base sur l'apprentissage profond pour la suggestion des films. Le modèle proposé utilise les descriptions textuelles des items (résumés, titres et genres des films) pour extraire les centres d'intérêt des usagers. Ce contexte implicite est ensuite intégré dans un réseau de neurones profond pour améliorer la qualité des prédictions. Afin d'atteindre cet objectif, nous combinons deux approches :

- la modélisation des thèmes en utilisant la méthode LDA (Latent Dirichlet Allocation) afin de représenter les centres d'intérêt des utilisateurs à partir des descriptions textuelles des films qu'ils ont consulté.
- l'intégration des centres d'intérêt des utilisateurs via un réseau de neurones multicouches, basé sur l'apprentissage profond.

Ce mémoire est organisé en trois chapitres. Le premier chapitre est consacré à la présentation des concepts de base sur les systèmes de recommandation et l'apprentissage

profond. Nous exposons principalement les méthodes de recommandation, leurs limitations ainsi que les métriques d'évaluation. Nous mettons également l'accent sur les réseaux de neurones et leurs applications dans les systèmes de recommandation.

Le deuxième chapitre est dédié à une étude conceptuelle détaillée du modèle proposé en présentant l'objectif principale de notre travail, l'architecture du réseau de neurones et les différents algorithmes utilisés.

Le dernier chapitre présente les outils et les langages utilisés pour le développement de notre système de recommandation ainsi que les différents tests et expérimentations effectués.

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

Chapitre I :

Systèmes de Recommandation et apprentissage profond

I. INTRODUCTION

Avec la quantité croissante de produits et services offerts sur Internet, les utilisateurs n'ont plus le temps de consulter toutes les informations qui sont à leur disposition. Au lieu de laisser l'utilisateur dépenser son temps à chercher l'information dont il a besoin, il est devenu primordial de concevoir des mécanismes qui lui facilitent la tâche en lui suggérant continuellement l'information qui l'intéresse [5]. Les systèmes de recommandations sont alors apparus comme un domaine de recherche propre depuis les années 90.

Les dernières années ont aussi vu le succès de méthodes à base de réseaux neuronaux dans plusieurs domaines, tels que la reconnaissance des formes et de la parole, la vision informatique, le traitement du langage naturel, etc. Ces modèles ont permis la création des représentations plus complexes et plus complètes en utilisant d'importantes quantités de données. Ainsi, les systèmes de recommandation à base de réseaux de neurones ont montré récemment leur capacité d'améliorer la qualité des prédictions offertes aux utilisateurs.

II. LE SYSTEME DE RECOMMANDATION

Un système de recommandation est un type spécifique du filtrage d'information qui consiste à proposer des listes d'items aux utilisateurs en se basant sur leurs profils. De tels items peuvent correspondre à différents types de données tels que des films, des livres, de la musique, des restaurants, des news, des pages Web, des blagues, des articles scientifiques, des documents, etc. [6]. Un système de recommandation permet donc d'estimer l'utilité d'un item pour un utilisateur. En effet, le problème le plus pertinent est le calcul des prédictions parce qu'une fois le système estime les valeurs des évaluations, la recommandation se réduit à suggérer les *N* premiers items ayant les plus grandes valeurs prédites [5].

III. METHODES DE RECOMMANDATION

Un système de recommandation permet d'extraire les informations qui sont susceptibles d'intéresser un utilisateur particulier à partir de l'observation de son comportement et de ses éventuelles appréciations. Une revue de la littérature dans ce domaine montre qu'il existe plusieurs méthodes de recommandation dont le filtrage basé sur le contenu et le filtrage collaboratif sont les plus utilisés [5].

III.1. Le filtrage basé sur le contenu (Content-basedrecommendation)

Le filtrage basé sur le contenu consiste à identifier les objets similaires à ceux précédemment appréciés par un utilisateur en fonction de leur contenu. Il peut être vu comme un système de recherche d'informations dont les requêtes sont implicitement composées du contenu des items déjà consultés.Les mots-clés représentant l'item sont, soit attribués manuellement ou extraits via une indexation automatique. L'importance d'un mot dans un item est généralement calculée en utilisant la mesure TF-IDF (TermFrequency-Inverse Document Frequency) [7]. Ce type de filtrage regroupe des approches communes avec la recherche d'informations telles que la classification (clustering) et les arbres de décision [5].

L'avantage des systèmes basés sur le contenu est que l'utilisateur soit indépendant des autres, ce qui lui permet d'avoir des recommandations même s'il est le seul usager du système. Cependant, cette méthode est inadéquate pour la prédiction des documents multimédia à cause de la difficulté de leur indexation [8]. Elle est également incapable de traiter d'autres critères de pertinence que les critères strictement thématiques.

Le filtrage basé sur le contenu souffre aussi du problème du démarrage à froid à cause du manque des données nécessaires pour construire le profil d'un nouvel utilisateur. Pour que la recommandation soit pertinente, il faut que l'utilisateur fournisse un certain nombre d'appréciations ou d'informations [7].

III.2. Le filtrage collaboratif (Collaborative filtering)

Le filtrage collaboratif est l'une des méthodes de recommandation les plus populaires. Il se base sur l'idée que les personnes à la recherche d'information devraient se servir de ce que d'autres ont déjà trouvé et évalué. Ainsi, il recommande à un utilisateur donné des items que d'autres usagers, similaires dans leurs goûts et préférences, ont appréciés. Ce type de filtrage ne prend pas en considération les informations sur les items, il utilise uniquement la matrice des évaluations pour calculer les prédictions. Par exemple, un utilisateur $u_1$qui s'intéresse seulement aux films policiers est similaire à un autre utilisateur u_2 qui regarde les films policiers et dramatiques. On peut donc proposer à l'utilisateur u_1 des films dramatiques.

Afin de suggérer un item à un utilisateur, l'algorithme de filtrage collaboratif détermine les usagers ayant des préférences similaires à cet utilisateur sur certains items. Les données disponibles sont ensuite exploitées pour calculer la prédiction (Figure I.1.).

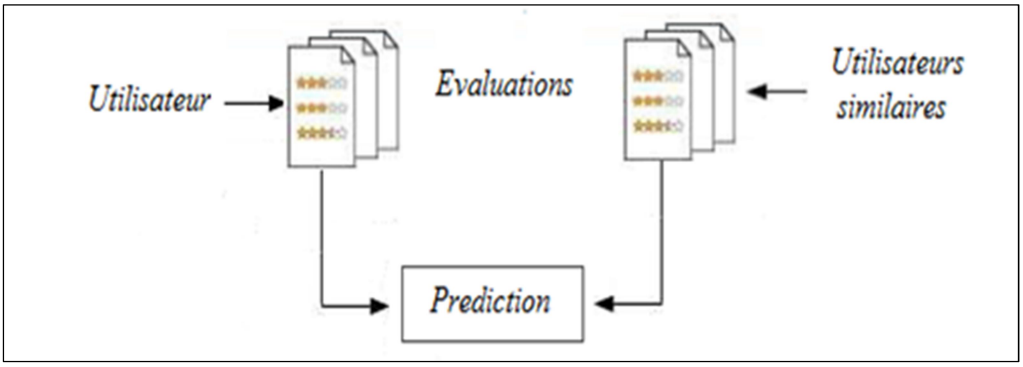


Figure I.1. Processus du filtrage collaboratif.

Formellement, le filtrage collaboratif est défini par un ensemble $U = \{u_1, u_2, \ldots, u_m\}$ de m utilisateurs et un ensemble $I = \{i_1, i_2, \ldots, i_n\}$ de n items. Chaque utilisateur u_k évalue un sous-ensemble d'items $I_{u_k} \subseteq I$. Une note $R_{u,i}$ indique l'appréciation attribuée par l'utilisateur u à l'item i. Elle peut être acquise de manière implicite en observant le comportement de l'utilisateur. Dans ce cas, toute interaction de l'utilisateur avec le système est considérée comme une estimation de son jugement d'intérêts [5]. Nous pouvons aussi obtenir l'évaluation de manière explicite en invitant les utilisateurs à donner des annotations simples à partir d'une échelle de notes fixée (par exemple de 1 ≡ très mauvais à 5≡ très bien). Les évaluations des utilisateurs sont stockées dans une matrice appelée *user*×*item*(tableau I.1)

	Item 1	*Item 2*	*Item 3*	*Item 4*	*Item 5*	…….
User 1	4	-	3	1	-	…….
User 2	5	4	2	-	-	…….
User 3	-	-	-	2	5	…….
…….	…….	…….	…….	…….	…….	…….

Tableau I.1. Exemple de la matrice *user*×*item*.

Les algorithmes de filtrage collaboratif sont faciles à implémenter et évoluent dynamiquement selon les profils des utilisateurs. Ils peuvent être appliqués à n'importe quel type d'items. Ils sont également capables de suggérer des objets non nécessairement similaires à ceux déjà recommandés. Ainsi, [9] affirment que le filtrage collaboratif donne généralement des résultats meilleurs que ceux obtenus par le filtrage basé sur le contenu, surtout lorsqu'il y a suffisamment de notes disponibles. Toutefois, cette technique souffre du coût élevé des calculs et du nombre réduit des évaluations disponibles surtout celles en commun entre deux utilisateurs, ce qui fournit des degrés de similarités peu pertinents. Elle est également incapable de produire des recommandations pour les nouveaux utilisateurs et de suggérer les nouveaux items.

III.3. Les systèmes de recommandation hybrides (Hybrid Recommender Systems)

Ces systèmes combinent deux ou plusieurs méthodes de recommandation pour améliorer les performances avec moins d'inconvénients. L'hybridation ou la combinaison peuvent être effectuées de différentes manières. Cependant, si le choix des paramètres de chaque approche n'est pas bien fait, elles donnent des résultats plus mauvais. Plusieurs méthodes issues de la fouille de données, de la recherche d'informations ou du clustering sont utilisées afin d'effectuer la combinaison [7]. En effet, la combinaison peut être réalisée en deux phases :

1. Effectuer de manière indépendante des recommandations via deux ou plusieurs méthodes.
2. Combiner les prédictions en utilisant des méthodes de combinaison telles que la pondération, la commutation, etc. [W1]

III.4. Les systèmes de recommandation contextuels

Les systèmes de recommandation traditionnels ne considèrent que les évaluations précédemment fournies par les utilisateurs. Ils opèrent dans un espace bidimensionnel parce qu'ils ne tiennent compte que de deux profils : usager et item. Les besoins informationnels des usagers sont généralement liés à des facteurs contextuels. Par conséquent, la recommandation pourra être fortement dépendante du contexte dans lequel elle est effectuée. Les systèmes de recommandation sensibles au contexte sont apparus afin de personnaliser et d'adapter dynamiquement les comportements des applications [10].

Brown [11] considère qu'«*un système sensible au contexte doit automatiquement extraire de l'information ou effectuer des actions en fonction du contexte utilisateur détecté par les capteurs* ». L'objectif principal de ces systèmes est donc la modélisation du contexte de recherche et son intégration comme une structure d'information additionnelle dans la chaîne d'accès à l'information.

IV. LES LIMITATIONS DES SYSTEMES DE RECOMMANDATION

L'utilisation des systèmes de recommandation est devenue une nécessité, vu qu'ils sont capables de suggérer des ressources pertinentes avec moindre effort et dans un délai de réponse satisfaisant. Toutefois, ces systèmes souffrent de certaines limitations, telles que le démarrage à froid, le manque des évaluations, la sélection de voisins fiables, la robustesse et la précision des recommandations, etc.

IV.1. Démarrage à froid (Cold-start)

Ce problème survient lorsque le système ne possède pas assez de données. L'algorithme de prédiction sera donc incapable de générer des recommandations. Le démarrage à froid se produit pour un nouvel utilisateur qui débute avec un profil vide, pour un nouvel item qui risque de ne pas être recommandé ou pour un nouveau système où les données, sur lesquelles se base le processus de filtrage, ne sont pas disponibles [12].

IV.2. Manque de données (saprsity problem)

Ce problème illustre le fait que la matrice *user×item* possède souvent un petit nombre d'évaluations, ce qui rend le calcul des similarités difficile. Plusieurs approches ont été proposées afin d'alléger l'impact du manque des appréciations sur l'identification des voisins fiables.

IV.3.Identification de voisinage fiable (Neighbors formation)

Le voisinage joue un rôle très important dans un système de filtrage collaboratif du fait que la qualité des recommandations fournies aux utilisateurs est étroitement liée à la qualité des voisins formés par le système. Les mesures de similarité statistiques, telles que le coefficient de Pearson, permettent d'identifier les voisins les plus proches en se basant sur

le calcul des similarités des appréciations entre un utilisateur actif et les autres utilisateurs vis-à-vis d'items co-notés dans le passé.

IV.4. Evolutivité (scalability)

Les systèmes de recommandations contiennent généralement un nombre important de données. Dans le cadre du filtrage collaboratif, le calcul des prédictions utilise un voisinage d'utilisateurs ou d'objets. Ainsi, avec des données qui ne cessent d'accroître, le temps de calcul des similarités risque d'augmenter considérablement, ce qui diminue les performances du système. Il est donc indispensable d'utiliser des techniques permettant aux systèmes de recommandation d'émettre leurs suggestions rapidement. Les méthodes basées modèle permettent de surmonter ce problème en limitant le nombre d'utilisateurs ou d'items utilisés dans le calcul des prédictions.

IV.5. Robustesse (Reliability)

Les systèmes de recommandation demeurent vulnérables aux attaques malveillantes et aux données bruitées. En effet, il n'y a aucune garantie que les évaluations intégrées aux systèmes reflètent réellement les préférences des utilisateurs. Il existe plusieurs formes d'attaques et de bruit. Certains individus peuvent par exemple insérer des informations afin de forcer le système à générer des notes élevées pour certains items [13].

IV.6. Dynamicité (dynamicity)

Le filtrage collaboratif ignore complètement les changements des habitudes des usagers selon le contexte. Il ne prend en considération que les évaluations des utilisateurs exprimées sous forme de notes [14]. La prédiction d'une valeur positive ne signifie en aucun cas que la recommandation est bonne, il faut prendre en considération le contexte et les besoins de l'utilisateur.

IV.7. Protection de la vie privée (privacy)

Les systèmes de recommandation doivent assurer la protection des informations personnelles de l'utilisateur et garantir la séparation entre données privées et publiques. Il s'agit d'une part, de préserver l'anonymat des utilisateurs et de protéger leurs informations et d'autre part, de crypter les données transmises et d'effectuer les transferts d'information

sur des liaisons sécurisées. Polat et Du [15] proposent un algorithme de filtrage collaboratif basé sur une décomposition en valeurs singulières. Le modèle proposé utilise des protocoles de communication spécifiques pour assurer la confidentialité des données. Les auteurs exploitent des techniques de perturbation aléatoire pour protéger la vie privée.

IV.8. Précision des recommandations

L'amélioration de la précision des prédictions est souvent un enjeu majeur de la plupart des travaux de recherche proposés dans le cadre des systèmes de recommandation [7]. La précision permet en effet de calculer la performance de l'intégralité du système. Ainsi, prédire des ressources satisfaisant les besoins des utilisateurs conduits à leur fidélisation.

IV.9. Nouveauté et diversité (Novelty and diversity)

La diversité des recommandations a une grande influence sur la satisfaction des utilisateurs. Ainsi, un bon système de recommandation doit trouver le meilleur compromis pour offrir à l'utilisateur des ressources suffisamment similaires à ses goûts connus sans l'ennuyer avec le même type d'items [16]. Ce qui permet d'éviter la redondance dans les ressources suggérées. Le filtrage collaboratif fournit une certaine forme de diversité. Ainsi, le problème de la diversité touche particulièrement les méthodes basées sur le contenu.

V. SYSTEME DE RECOMMANDATION ET RESEAUX DE NEURONES

Un réseau de neurones artificiels est une méthode d'apprentissage automatique inspirée du modèle biologique. Il consiste à imiter les fonctions humaines de mémorisation et d'apprentissage [17]. Ainsi, un réseau de neurone profond consiste en des milliers voire des millions de neurones, répartis en plusieurs dizaines de couches, chacune représente une étape de traitement. En effet, plus on augmente le nombre de couches, plus les réseaux de neurones apprennent des choses compliquées, abstraites, correspondant de plus en plus à la manière dont un humain raisonne.

Formellement, un réseau de neurones est un graphe orienté et pondéré. Un nœud dans le réseau représente un neurone formel qui est une unité de calcul élémentaire dotée d'une valeur d'activation. Les neurones sont organisés sous forme de couches et interconnectés par l'intermédiaire des poids qui jouent le rôle de liens synaptiques. Chaque poids

représente le degré d'interconnexion des nœuds qu'il relie.Un neurone formel est donc caractérisé par (Figure I.2) :

- Un ensemble d'influences provenant d'autres neurones par le biais des liens synaptiques pondérés.
- Une activation pondérée calculée à partir de ces influences.
- Un seuil θ qui est une valeur au-dessous de laquelle la réponse du neurone est ignorée.
- Eventuellement, une entrée extérieure permettant d'exprimer des informations aux neurones d'entrée.
- Une fonction d'activation qui détermine l'activation réelle du neurone à partir de son activation pondérée, de son seuil, et de son entrée extérieure.

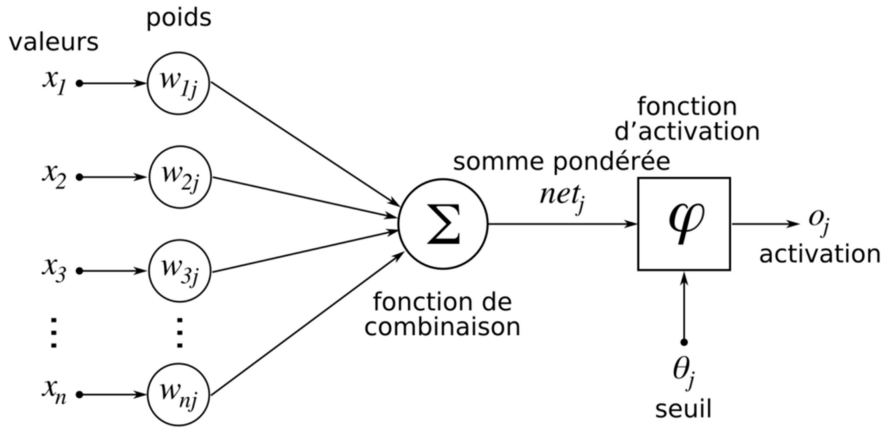


Figure I.2. Structure d'un neurone formel [W2]

De façon générale, un neurone reçoit un signal provenant de la couche d'entrée ou des neurones adjacents, et émet à son tour un signal représentant sa valeur d'activation lorsqu'il est supérieur à un certain seuil.

Un réseau de neurones peut apprendre et généraliser à partir d'exemples représentatifs. Ce qui permet de développer un modèle sans disposer de connaissances théoriques du domaine. Ainsi, à partir d'un ensemble de données, il est capable de modéliser une grande variété de comportements. Il est également robuste aux données bruitées. Toutefois, cette technique présente quelques inconvénients. En plus du problème de sur-apprentissage, elle manque de méthode permettant de définir la meilleure topologie du réseau, les valeurs initiales des poids, le pas d'apprentissage et le nombre de neurones des différentes couches. En effet, ces différents paramètres ont un rôle primordial sur la convergence du réseau.

V.1 Entraînement

Le réseau de neurones généralise des connaissances à partir d'une base d'exemples. Il ajuste son comportement grâce à des règles d'apprentissage permettant l'évolution des poids des connexions au cours du temps. En effet, l'apprentissage permet aux réseaux de neurones de réaliser des tâches complexes dans différents types d'application (classification, identification, reconnaissance de caractères, reconnaissance de la parole, vision, système de contrôle, etc.). C'est un processus graduel et itératif se basant sur la propagation des signaux d'activation depuis les neurones d'entrées jusqu'aux sorties. Il consiste à régler les poids synaptiques grâce à une grande quantité de données, que l'on regroupe dans un *corpus d'apprentissage*. Ainsi, le réseau reçoit en entrée des données sur lesquelles il applique une fonction de transformation pour produire un résultat. Les poids synaptiques sont alors ajustés en minimisant une fonction d'erreur [17]. On distingue usuellement deux types d'apprentissage: l'apprentissage supervisé et l'apprentissage non supervisé.

L'apprentissage supervisé permet d'apprendre une fonction qui, à partir d'un échantillon d'apprentissage, se rapproche le mieux de la relation entre entrée et sortie désirée. Les données utilisées pour l'apprentissage supervisé sont une série d'exemples comportant des paires composées de données en entrée et de sorties attendues. L'apprentissage non supervisé permet de trouver des structures sous-jacentes à partir de données non étiquetées. Ainsi, le système doit détecter les similarités dans les données qu'il reçoive et les organiser en groupes ou clusters. Cette technique est utilisée pour des réseaux de neurones spécifiques tels que les réseaux auto-encodeur.

V.1.1 Apprentissage profond (Deep Learning)

L'apprentissage profond (deep learning) est un ensemble de techniques d'apprentissage automatique basées sur des réseaux de neurones artificiels dont le nombre de couches et de neurones est important. Il permet aux machines d'apprendre plusieurs niveaux de représentations et d'abstractions d'objets à partir de grande quantité de données. Il est caractérisé par sa puissance et sa flexibilité considérables en apprenant à représenter le monde comme une hiérarchie imbriquée de concepts [W3]. Il est capable d'apprendre des caractéristiques directement depuis les données, sans avoir à effectuer une extraction manuelle. Les progrès de l'apprentissage profond ont été possibles grâce à l'augmentation

de la puissance des ordinateurs et au développement de grandes bases de données (big data).

V. 2 Architecture des réseaux de neurones

Dans un réseau de neurones, les neurones peuvent être structurés de différentes manières. De façon générale, l'architecture des réseaux peut aller d'une connectivité totale (tous les neurones sont reliés les uns aux autres) à une connectivité locale où les neurones ne sont reliés qu'à leurs plus proches voisins. Il existe différentes architectures possibles pour construire un réseau de neurones [17]. Chaque modèle est développé pour une tâche particulière. Les différentes architectures ont leurs forces et faiblesses et peuvent être combinées pour optimiser les résultats.

V.2.1 Réseaux de neurones récurrents

Les réseaux de neurones récurrents permettent d'analyser les séquences de vecteurs comme les modèles de Markov cachés. Ils comportent des cycles dans leur graphe de connectivité. Ces cycles permettent au réseau d'entretenir une information en se l'envoyant à lui-même (Figure I.3). En effet, la réponse de la couche de sortie ou des couches cachées est réinjectée dans la couche d'entrée ou dans des couches cachées [18]. Ces modèles permettent d'apprendre, de stocker et de prendre en considération l'information contextuelle passée lors du traitement de l'information à l'instant actuel.

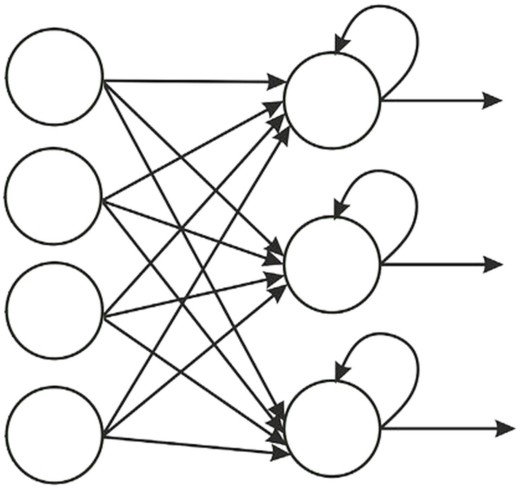


Figure I.3. Réseau de neurones récurrent

V.2.2 Réseaux de neurones convolutionnels

Les réseaux de neurones convolutionnels sont basés sur l'utilisation successive de plusieurs filtres afin de détecter des caractéristiques dans l'image (Figure I.4). Les premiers filtres permettent en général de détecter des caractéristiques de bas niveau alors que les derniers isolent des caractéristiques de haut niveau [19].

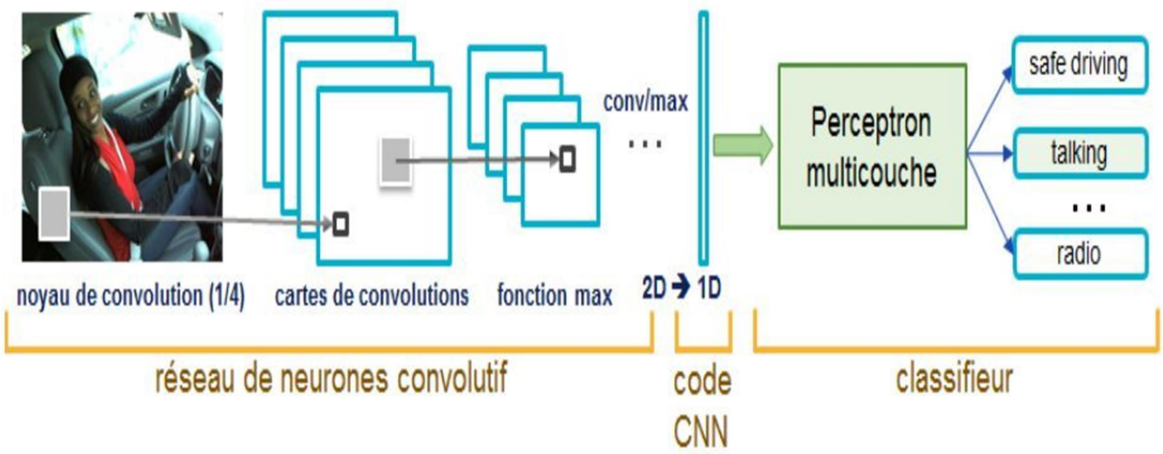


Figure I.4. Réseau de neurones convolutionnel [18]

V.2.3 Réseaux de neurones auto-encodeur

Les réseaux de neurones auto-encodeur utilisent des méthodes d'apprentissage non supervisées. Ils peuvent effectuer une réduction de dimensionnalité non linéaire (Figure I.5). Ils sont surtout adaptés pour le traitement des informations spatiales. Ces modèles sont capables d'étudier la répartition de données dans des grands espaces comme par exemple pour des problématiques de clusterisation ou de classification[W3].

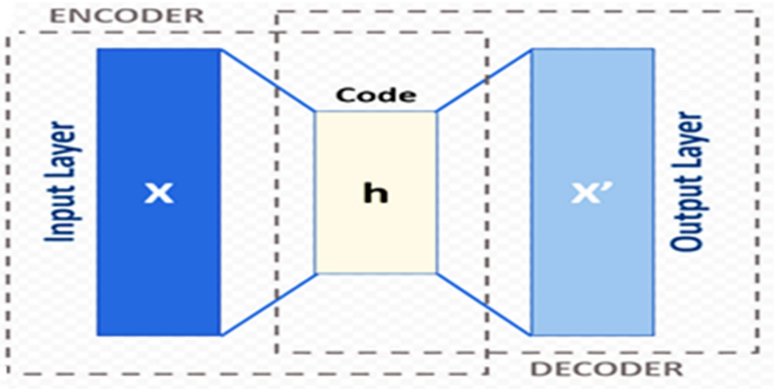


Figure I.5. Réseaux de neurones auto-encodeur [W4]

V.2.4 Réseaux de neurones multicouches

L'idée principale des réseaux de neurones multicouches est de grouper des neurones dans des couches. On place ensuite bout à bout plusieurs couches et on connecte complètement les neurones de deux couches adjacentes. Les entrées des neurones de la deuxième couche sont en fait les sorties des neurones de la première couche. Les neurones de la première couche sont reliés au monde extérieur et reçoivent tous le même vecteur d'entrée (c'est en fait l'entrée du réseau) (Figure I.6). Ils calculent alors leurs sorties qui sont transmises aux neurones de la deuxième couche. Les sorties des neurones de la dernière couche forment la sortie du réseau. Un réseau de neurones multicouches calcule donc une fonction vectorielle. On peut ajuster les valeurs des connexions synaptiques et des seuils afin de modifier la fonction calculée. [20]

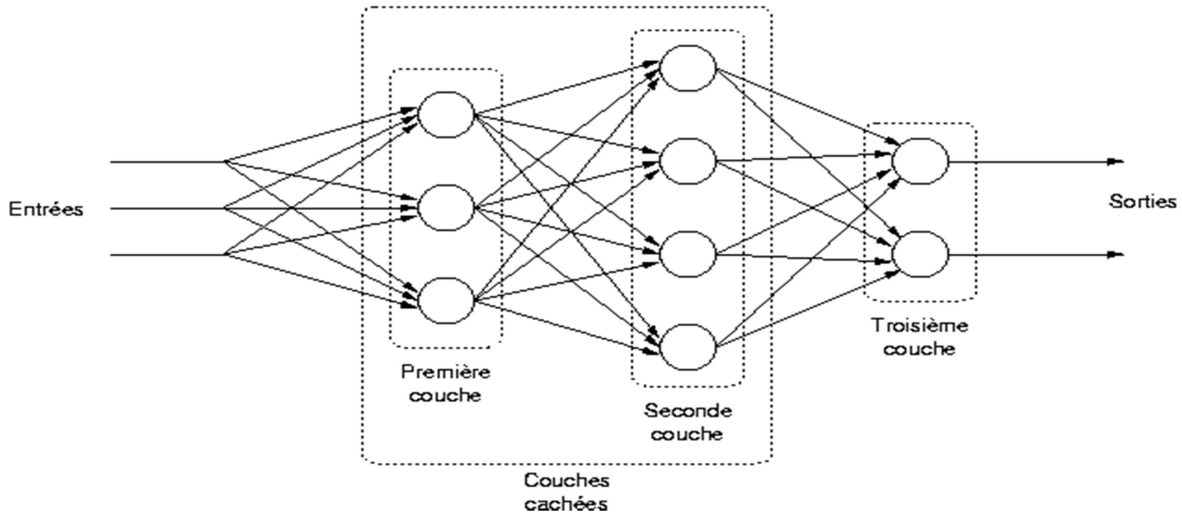


Figure I.6. Réseau de neurones multicouches

V.3. Les fonctions d'activation

Il existe plusieurs types de fonctions d'activation pour les réseaux de neurones artificiels. Chacune d'elles est utilisée dans certaines circonstances.

V.3.1 Fonction Sigmoïde

La fonction sigmoïde est l'une des premières fonctions d'activation (Figure I.7). Elle est aussi connue sous le nom de fonction logistique. Elle transforme les variables réelles en valeurs allant de 0 à 1 selon l'équation (1).

$$g(x) = 1/(1 + e^{-x}) \qquad (1)$$

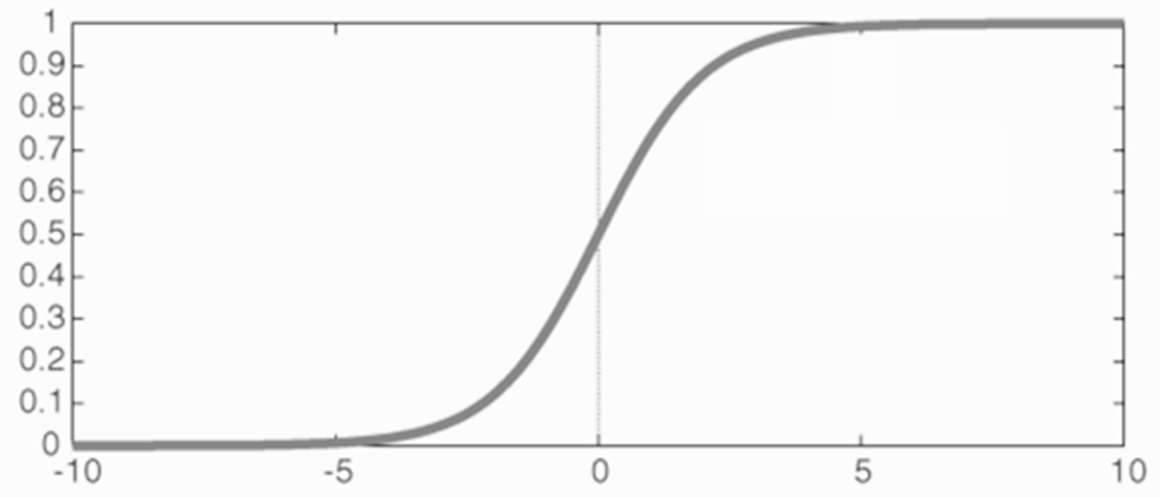


Figure I.7. La fonction Sigmoïde

V.3.2 Fonction Tangente hyperbolique

La Tangente hyperbolique (*tanh*) est couramment utilisée dans la couche de sortie.Elle transforme toute entrée réelle en une valeur comprise entre -1 et 1(Figure I.8). La Tangente hyperbolique est calculée selon l'équation (2).

$$g(x) = tanh(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x}) \qquad (2)$$

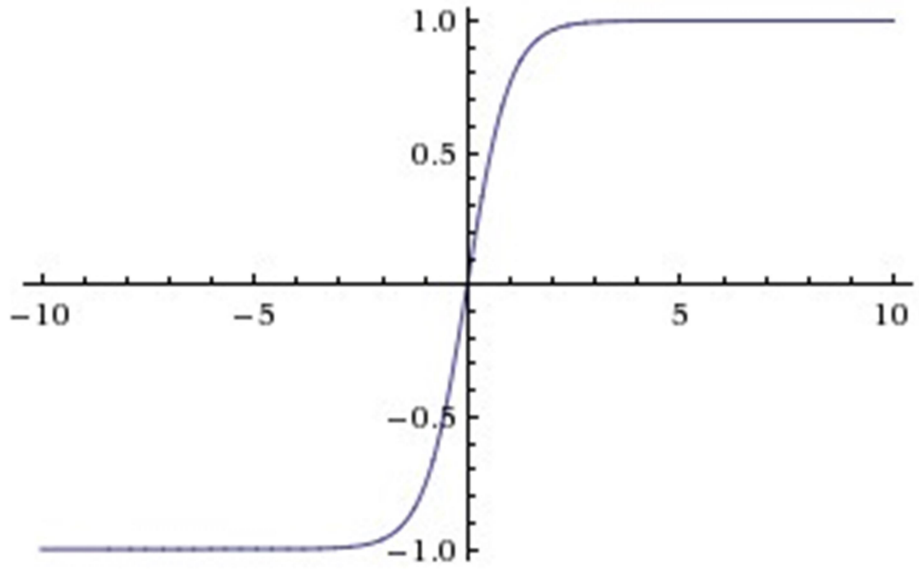


Figure I.8. La fonction Tangente hyperbolique

V.3.3 Fonction Softmax

Softmax est couramment utilisée dans la couche de sortie. Elle peut être considérée comme une distribution de probabilité sur les catégories. Cette fonction d'activation est calculée selon l'équation (3).

$$g(x) = e^{x}_{i} / \sum_{i} e^{x}_{i} \qquad (3)$$

V.3.4 Unité linéaire rectifiée ("Rectified Linear Unit", ReLU)

Cette fonction d'activation et ses variantes sont fréquemment utilisées en apprentissage profond pour les couches cachées. Elles permettent un entrainement plus rapide comparé aux fonctions *sigmoid* et *tanh*. Il existe plusieurs variantes de *ReLU*, parmi lesquelles : *Softplus* ou *SmoothReLU*, *LeakyReLUs*, *Noisy ReLUs*, *PReLUs* (ParametricReLUs), *ELUs* (ExponentialLinear Unit). *ReLU* peut être calculée selon l'équation (4) (Figure I.9).

$$R(x) = max(0, z) \qquad (4)$$

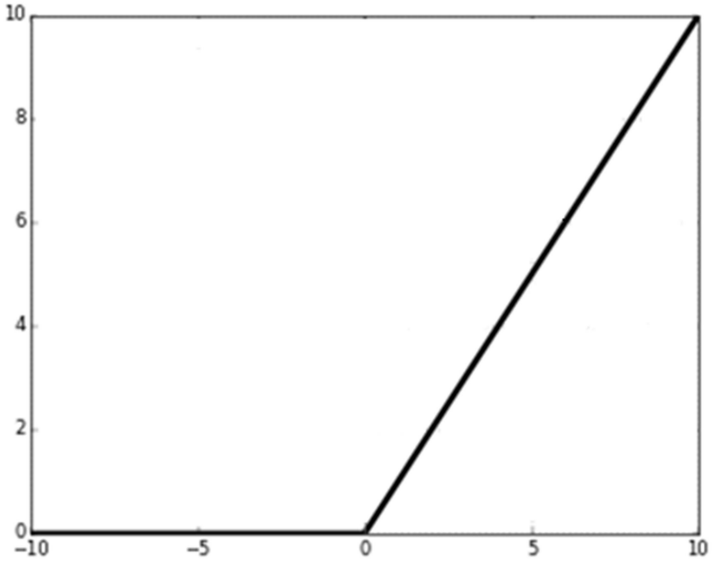


Figure I.9. La fonction ReLU

V.4 Algorithme d'optimisation

Différents algorithmes d'optimisation sont proposés, parmi lesquels :

V.4.1. Algorithme de la rétro-propagation

Plusieurs variantes de cet algorithme ont été développées par des chercheurs travaillant dans des domaines différents. La première formulation de la version actuelle a été faite par Werbos en 1974 [W3]. L'algorithme de la rétro-propagation altère les coefficients synaptiques (*wi*) du réseau dans le sens inverse du gradient du critère d'erreur, en utilisant seulement les données d'entrée/sortie. A chaque itération, on calcule une nouvelle estimation des poids synaptiques pour chaque exemple d'apprentissage (*xi*, *yi*). En effet, une itération consiste en deux phases :

- **Propagation** : à chaque itération, un élément de l'ensemble d'apprentissage est introduit à travers la couche d'entrée. L'évaluation des sorties du réseau se fait couche par couche, de l'entrée vers la sortie
- **Rétro-propagation** : cette étape est similaire à la précédente. Cependant, les calculs s'effectuent dans le sens inverse (de la sortie vers l'entrée). A la sortie du réseau, on calcule un critère de performance *E* en fonction de la sortie réelle du système et sa valeur désirée. Puis, on évalue le gradient de *E* par rapport aux différents poids en commençant par la couche de sortie et en remontant vers la couche d'entrée.

V.4.2. Algorithme d'optimisation de la descente de gradient

La méthode de descente de gradient a fait ses preuves pour l'optimisation des paramètres des réseaux de neurones. Elle est ainsi l'une des approches les plus efficaces et les plus utilisées [21]. Le but de cet algorithme est de trouver les paramètres qui minimisent l'erreur du modèle en utilisant les données d'apprentissage. Ainsi, il modifie le modèle en le déplaçant le long d'une pente d'erreur vers une valeur d'erreur minimale.Il existe plusieurs algorithmes d'optimisation par la descente de gradient, dontles plus populaires sont Momentum, Adagrad et Adam.

V.5 Travaux de recherche sur les réseaux de neurones dans les systèmes de recommandation

Les réseaux neuronaux sont largement utilisés dans de nombreux problèmes de classification (reconnaissance de formes, traitement de signal) d'estimation (modélisation de phénomènes complexes) et de prévision (bourse, ventes). Leurs capacités à intégrer des sources multiples d'information et à tirer profit d'importants volumes de données ont permis leur application aux systèmes de recommandation.

Plusieurs travaux utilisent les réseaux de neurones dans les systèmes de recommandation autant pour la mise en œuvre du processus de recherche que pour la modélisation des unités textuelles [22]. Un réseau de neurones est constitué d'un ensemble de nœuds et de liens. Cette structure peut être convenable pour représenter les relations entre les termes et les documents. Les neurones formels représentent des objets de la recherche d'informations. Le processus de recherche est alors basé sur un mécanisme d'activation/propagation depuis les neurones de la couche d'entrée vers la couche de sortie. Les items sont proposés à l'utilisateur en fonction de l'activation des neurones de sortie.

Les auteurs dans [23] proposent un système de recommandation hybride pour la personnalisationdes sites de commerce électronique. Le modèle proposé utilise un réseau de neurones de type ART ("Adaptive Resonance Theory") afin d'effectuer une classification non-supervisée des profils utilisateurs. Le réseau de neurones utilisé est constitué de deux couches : une couche d'entrée qui détecte les descriptions du profil utilisateur et une couche de sortie qui classe ces données en catégories. La méthode de *kano* est ensuite appliquée afin de garantir la satisfaction psychologique des clients. Cette méthode permet en effet d'extraire les besoins implicites des utilisateurs dans les différents

clusters. Le système proposé traite le problème du démarrage à froid pour un nouvel utilisateur, cependant il nécessite un nombre suffisant d'évaluations.

Dans [24] présentent un nouveau modèle comportemental de recommandation inspiré du *Web Usage Mining*. Les auteurs considèrent que deux utilisateurs ayant des motifs d'usage communs sont similaires. Ainsi, le système proposé modélise les utilisateurs en analysant leur comportement de navigation à partir des traces d'usage. Il utilise également un réseau de neurones artificiels afin de prédireles besoins potentiels desclients.

Les auteurs dans [25] développent une approche de recommandation des films basée sur l'hypothèse que les événements qui se produisent au fil du temps peuvent avoir une influence sur les recommandations. Les évaluations de chaque utilisateur sont donc regroupées dans desintervalles de temps prédéfinis. Un réseau de neurones est ensuite utilisé afin d'estimer les appréciations des usagers.

Afin de traiter le problème du manque de données et de démarrage à froid, le travail présenté dans [26] propose un système de recommandation qui utilise un réseau de neurones probabiliste. Ce dernier se base sur la matrice des évaluations pour calculer la confiance entre les utilisateurs. Les valeurs de confiance sont ensuite utilisées pour construire des groupes d'utilisateurs à partir desquels les recommandations seront calculées.

Dans [27] les auteurs proposent un système de recommandation sensible au contexte pour la TV numérique. Le modèle proposé utilise un réseau de neurones artificiel afin de traiter le problème du démarrage à froid. En effet, lesprogrammesde télévision disponiblessont représentés par des vecteurs de 24 caractéristiques. Cet espacevectorielsubit ensuite unetransformation linéaire afin de réduire sa dimension. Le système apprend les habitudes des téléspectateurs en se basant sur un réseau de neurones multicouches. La couche d'entrée contient cinq neurones dont deux neurones sont utilisés pour prendre en considération les informations temporelles. La couche de sortie comporte deux neurones afin de déterminersi l'utilisateur aime ou nonun programmede télévision.

Une étude sur l'utilisation des réseaux de neurones pour les systèmes de recommandations a été réalisée dans [28]. Les papiers étudiés utilisent des systèmes de recommandations dans plusieurs domaines tels que la recommandation de musiques, de nouvelles ou de

citations. Les auteurs divisent les modèles proposés en deux catégories en se basant sur l'utilisation d'un seul type de réseau de neurones ou de la combinaison de plusieurs.

Avant de combiner des utilisateurs et des éléments avec un autre réseau multicouche, [29] [20] [31] utilisent un réseau neuronal multicouche pour coder les utilisateurs et les éléments séparément.

Dans [32] les chercheurs s'inspirent des modèles utilisant des représentations vectorielles de mots (word2vec). Ils proposent un système de recommandations utilisant une factorisation de matrices avec des filtres collaboratifs. Ils décomposent ainsi la matrice des évaluations en un produit de facteurs latents pour les utilisateurs et les items. Ils calculent également la matrice SPPMI (*shifted positive pointwise mutual information*) qui correspond aux patrons de co-occurrence entre chaque item et son contexte. En effet, le contexte représente tous les items apparaissant dans l'historique d'un usager. Les auteurs proposent une fonction objective qui combine la différence entre la valeur prédite et la valeur réelle, et une régularisation des facteurs latents pour les utilisateurs et les items. D'autre part, ils intègrent à la fonction précédente la différence entre la matrice de co-occurrence et une combinaison linéaire des représentations d'item par la factorisation de matrice et les représentations vectorielles du contexte. L'apprentissage se fait par une méthode similaire aux moindres carrés alternés (*Alternating Least Squares* ou ALS).

Le système de recommandation proposé par [30] est un modèle hiérarchique bayésien composé de deux parties. La première partie utilise un *Stacked Denoising Autoencoder (SDAE)*, qui est un réseau de neurones encodant des données d'entrées dans une représentation plus petite, pour créer une représentation vectorielle de l'item. La sortie de l'autoencodeur devient l'entrée d'un décodeur afin de reconstruire les données de départ. Les auteurs utilisent deux jeux de données de citations d'articles et un troisième provenant d'un concours présenté par Netflix[1]. Les données d'entrée des deux premiers ensembles de données sont le titre et le résumé, tandis que les données d'entrée de Netflix sont le résumé du film obtenu à partir d'IMDB[2]. En fait, ces données ont été converties en un sac de mots avant d'être traitées par SDAE. Ensuite, le score de l'utilisateur pour l'item est calculé comme le produit des représentations cachées des items et des utilisateurs.

[1]https://www.netflix.com
[2]https://www.imdb.com/

Dans [33] les auteurs utilisent des auto-encodeurs afin d'apprendre la représentation cachée non linéaire des items. Ainsi, [34] étende ce modèle afin de traiter le problème de démarrage à froid.

Le travail présenté dans [35] utilise deux réseaux multicouches pour modéliser les données et les règles d'experts. L'apprentissage des deux réseaux se base sur la minimisation de la distance entre leurs sorties.

De leur côté, les chercheur dans [36] proposent une approche qui permet de modéliser les interactions de haut niveau entre les utilisateurs et les items par des réseaux multicouches et les interactions de bas niveau par la machine de factorisation. Les deux méthodes sont ensuite combinées pour calculer les prédictions.

En se basant sur les différences de préférence entre items, le modèle proposé dans [37] combine une approche bayésienne avec un auto-encodeur pour créer une liste ordonnée de recommandations.

Dans [38] les auteurs étendent le modèle précédent en ajoutant un réseau multicouches afin de connecter directement la dernière couche avec certains attributs qui doivent être sélectionnés. Dans un autre travail, [39] utilise un réseau neuronal pour sélectionner les items et les attributs les plus importants pour chaque usager afin d'améliorer les filtres collaboratifs traditionnels.

En utilisant deux réseaux de neurones convolutionnels, [40] utilisent les textes des critiques pour encoder le comportement des utilisateurs et les attributs des items. Les résultats des deux réseaux sont ensuite combinés avec la dernière couche par une machine de factorisation.

Le travail présenté dans [41] combine des réseaux de neurones convolutionnels basés sur un mécanisme d'attention pour recommander des hashtags en utilisant le texte d'un tweet comme donnée. Une couche d'attention est utilisée pour donner de l'importance aux mots les plus informatifs.

Un RNN peut aussi être utilisé afin de considérer l'évolution des préférences des utilisateurs au cours du temps [42].

Le modèle est par la suite étendu en utilisant des informations additionnelles et des RNNs pour chaque type de données [43]. D'autres chercheurs [44] ont utilisé un réseau de neurones récurrents (RNN) pour recommander la prochaine action d'un utilisateur en se basant sur ses actions précédentes. Les résultats sont enfin concaténés pour prédire le prochain item. Dans [45], l'information additionnelle est directement utilisée comme entrée d'un RNN pour prédire les clics suivants.

Le travail présenté dans [46] et [47] propose un modèle de recommandation en deux étapes. Dans la première phase, un auto-encodeur est utilisé pour reconstruire les attributs d'un utilisateur. Ainsi une couche cachée est concaténée à un *embedding* de l'utilisateur. Ce dernier est un vecteur qui représente l'utilisateur, dont les valeurs sont ajustées après la phase d'apprentissage. Le même travail est effectué pour l'item avec un auto-encodeur et un autre *embedding*. Ensuite, les auteurs utilisent deux méthodes différentes pour prédire la sortie. Ils appliquent la méthode GMF++ qui multiplie ces deux vecteurs par élément puis elle passe le résultat à un réseau multicouches. La seconde méthode, appelée MLP++, se base sur la concaténation.

Pour prédire des hashtags, [47] combinent un mécanisme d'attention avec un RNN. Le modèle proposé utilise la méthode LDA pour trouver les thèmes des différentes publications. En effet, l'attention est calculée en utilisant les distributions de thèmes.

D'autres chercheurs [48] proposent une méthode d'incorporation d'informations contextuelles en se basant sur les réseaux de neurones. Le modèle proposé facilite la modélisation des relations entre les entrées et les sources. En effet, ils transforment les paramètres contextuels en vecteur. Puis, ils les additionnent ensemble avec 1. Ce vecteur est ensuite multiplié par élément avec soit l'entrée ou la sortie d'un RNN. Les auteurs appellent cette méthode *Latent Cross*. Ils la considèrent comme étant un masque ou un mécanisme d'attention sur des sources d'information contextuelles.

VI. EVALUATION DES SYSTEMES DE RECOMMANDATION

L'évaluation des systèmes de recommandation s'effectue généralement selon un processus de validation croisée (cross-validation) [49]. Elle consiste à estimer la fiabilité d'un modèle proposé en utilisant une technique d'échantillonnage. Ainsi, l'ensemble des

données est découpé aléatoirement en deux sous-échantillons : un pour l'apprentissage, il est utilisé pour construire le modèle, et l'autre pour le test, à partir duquel l'erreur est estimée en calculant une mesure de performance. Il existe plusieurs mesures de l'erreur parmi les plus utilisées, l'erreur moyenne quadratique et l'erreur absolue moyenne.

VI.1. L'erreur moyenne quadratique (Mean Squared Error : MSE)

L'erreur moyenne quadratique permet de mesurer la qualité des systèmes de recommandation qui utilisent les votes des utilisateurs comme information explicite. Elle calcule le carré de la distance entre la valeur prédite et l'évaluation réelle pour donner plus d'importance à l'erreur. Cette métrique peut être calculée selon l'équation (5).

$$MSE = \frac{\sum_{i=1}^{N} (prediction_i - reel_i)^2}{N} \quad (5)$$

$prediction_i$ est la valeur prédite par le système.

$reel_i$est l'évaluation réelle de l'utilisateur.

N représente le nombre total des évaluations prédites.

La racine de l'erreur moyenne quadratique (Root Mean Squared Error : RMSE) est une variante de l'erreur moyenne quadratique [50]. Elle permet de calculer la racine carrée de l'erreur entre l'évaluation prédite par le système et la note réelle donnée par l'utilisateur selon l'équation (6).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (prediction_i - reel_i)^2}{N}} \quad (6)$$

VI.2. L'erreur absolue moyenne (Mean Absolute Error : MAE)

Cette mesure permet de calculer la moyenne des erreurs absolues entre les évaluations prédites et réelles. Cette métrique se calcule selon l'équation (7).

$$MAE = \frac{\sum_{i=1}^{N} |prediction_i - reel_i|}{N} \quad (7)$$

$reel_i$ représente la note réelle de l'utilisateur.

$prediction_i$ est la valeur estimée par le système.

N représente le nombre total des évaluations prédites.

VI.3.L'erreur absolue moyenne normalisée (Normalized Mean Absolute Error : NMAE)

Cette mesure est l'une des variantes de l'erreur absolue moyenne. Elle est calculée lorsqu'on procède à des expérimentations qui utilisent différents corpus d'évaluation. L'erreur absolue moyenne normalisée peut être calculée selon l'équation (8).

$$NMAE = \frac{MAE}{u_{max} - u_{min}} \quad (8)$$

u_{max} et u_{min} représentent respectivement les bornes supérieure et inférieure des évaluations.

VI.4. High Mean Absolute Error (HMAE)

Après le calcul des prédictions, seulement les ressources ayant les plus grandes valeurs sont suggérées à l'utilisateur. Ainsi, la performance des systèmes de recommandation dépend de l'erreur relative aux items recommandés et non pas à ceux ayant de faibles valeurs de prédiction.La métrique High MAE calculela moyenne des erreurs absolues (MAE) uniquement sur les appréciations élevées [9]. Elle permet d'évaluer seulement les items jugés pertinents par le système et ne considère pas les erreurs dans les notes basses. Par exemple, sur une échelle de 1 à 5, la métrique High MAE considère seulement les votes entre 4 et 5.

VI.5. L'erreur absolue moyenne par utilisateur (UMAE : User Mean Absolute Error)

La métrique *UMAE* est proposée par Massa et Avesani pour estimer la satisfaction de chaque utilisateur. Elle calcule pour chaque utilisateur l'erreur absolue moyenne ensuite elle mesure la moyenne de ces erreurs sur l'ensemble de tous les utilisateurs afin d'estimer la capacité du système à satisfaire le maximum de ses clients. L'erreur absolue moyenne par utilisateur peut être calculée selon l'équation (9).

$$UMAE = \frac{\sum_{j=1}^{N} \frac{\sum_{i=1}^{N} j|p_{ij} - a_{ij}|}{N_j}}{N} \quad (9)$$

N représente le nombre total d'utilisateurs.

N_j correspond au nombre des notes prédites par l'utilisateur j.

VII. CONCLUSION

Dans ce chapitre, nous avons exposé les systèmes de recommandation, les principales techniques de recommandation et leurs limitations. Nous avons également présenté les systèmes de recommandation à base de réseaux de neurones ainsi que les différents travaux de recherche dans ce domaine. Ces travaux s'attachent généralement à résoudre un ou plusieurs problèmes parmi les enjeux auxquels doit faire face ce genre de systèmes.

Le chapitre suivant présente notre système de recommandation à base de réseau de neurones profond. Le modèle proposé permet l'intégration de l'information temporelle des items afin d'améliorer la qualité des prédictions.

Chapitre II :

Conception

I. MOTIVATION ET OBJECTIF

Face à la démocratisation du web et l'augmentation exponentielle de la quantité d'informations disponibles et accessibles, les systèmes de recommandation deviennent de plus en plus indispensables. Cependant, ils ont un grand challenge dans le cadre d'un environnement dynamique caractérisé par l'évolution des besoins de l'usager et par la dynamicité du contenu des informations disponibles. En effet, la pertinence,l'opportunité et l'adaptation des ressources délivrées aux besoins et aux contextes des utilisateurs, constituent des facteurs clés de succès des systèmes de recommandation.

Afin de proposer des items à un utilisateur, un système de recommandation à base de filtrage collaboratif s'appuie sur un profil représentatif des préférences de cet utilisateur. Pour le construire, il doit recueillir des informations sur celui-ci, soit explicitement via un formulaire ou implicitement par analyse de traces. En effet, le filtrage collaboratif exploite les appréciations des utilisateurs sur des items afin de déterminer les ressources à suggérer. Une fois les prédictions sont générées, une liste d'items jugés pertinents, triée généralement par ordre d'importance, est proposée automatiquement à l'utilisateur qui choisit d'accepter ou non la consultation des ressources recommandées.

Le filtrage collaboratif s'appuie sur l'idée que les personnes à la recherche d'information devraient se servir de ce que d'autres ont déjà trouvé et évalué. Par exemple, avant de lire un livre ou même d'acheter un appareil électroménager, beaucoup de personnes ont l'habitude de se renseigner auprès de leur entourage afin de récolter des avis. Toutefois, fixer un vote pour un utilisateur n'est pas une tâche facile ; cette estimation peut être influencée par de nombreux facteurs.

Dans ce travail, nous proposons un système de recommandation sensible au contexte pour suggérer des films aux usagers. Nous modélisons le contexte à partir d'informations contextuelles non observables. Ainsi, nous avons choisi de capturer le contexte implicite à travers les thèmes associés aux items consultés par lesutilisateurs. En effet, la méthode LDA (Latent Dirichlet Allocation) permet cette modélisation des thèmes. Le résultat obtenus à partir du modèle LDA est ensuite intégré dans un réseau de neurones multi-couches (MLP) basé sur un apprentissage profond afin d'améliorer la qualité des prédictions. Ainsi, l'objectif principal de notre travail est la combinaison de la méthode LDA avec un réseau de neurones profond afin d'aider les utilisateurs a surmonté le problème de surcharge d'informations.

II. DESCRIPTION DU SYSTEME PROPOSE

Un système de recommandation doit faire des prédictions qui s'adaptent aux intérêts des utilisateurs. Dans ce travail, on a combiné la méthode LDA avec un réseau de neurones multicouches (MLP) basé sur l'apprentissage profond pour améliorer la qualité des recommandations. On a utilisé ces deux méthodes à cause de leur efficacité et performance prouvées à travers la littérature. De plus, le domaine de la recommandation a reconnu, ces dernières années, un grand changement après le passage des algorithmes de factorisation matricielle traditionnels aux méthodes basées sur l'apprentissage profond. L'architecture générale du système de recommandation proposé dans ce mémoire est présentée par la figure suivante :

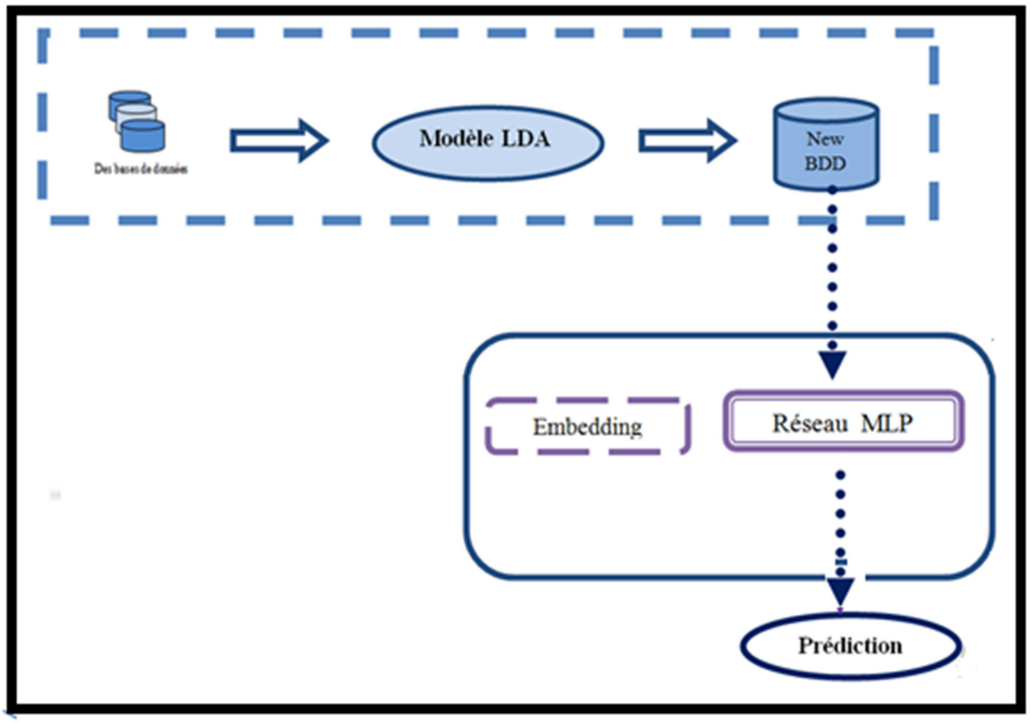


Figure II.10. Architecture du système proposé

III. PREPARATION DES DONNEES

Afin d'effectuer les tests nécessaires pour l'évaluation du système proposé, nous avons utilisé la base de donnés *MovieLens* ainsi que CMU Movie Summary Corpus. Ce dernier est utilisé pour la modélisation générative de sujets des résumés de films. Il contient des résumés pour 42306 films et des métadonnées telles que le genre, l'année de sortie, le casting, etc. Chaque film est indexé par un identifiant du film Wikipedia. Dans le cadre de ce travail, nous utilisons les résumés, les genres et les titres des films à partir de ces deux

bases. Ainsi, nous sélectionnons l'ensemble de films pour lesquels nous avons à la fois des résumés et des notes d'utilisateurs. Les données nécessaires sont donc extraites à partir de plusieurs fichiers, et rangées dans un seul fichier pour simplifier l'évaluation de l'algorithme implémenté (Figure II.11). Les données sont ensuite triées dans un ordre croissant selon les *user id* puis les *timestamps*.

```
Entrée [53]: test.head()
```

Out[53]:

	userid	itemid	rating	sexe	age	temp	ecart	occupation	title	code	runtime	genre	resume
27839	490	993	1	1.0	0.333333	0.037884	0	0.05	Infinity	3450620	0.500000	Drama	The film follows the book What Do You Care Wha...
30900	543	135	5	0.5	0.393939	0.050783	1322	0.85	Down Periscope	643850	0.386555	Comedy	Lieutenant Commander Thomas Dodge , a capable ...
17130	314	1089	1	1.0	0.196970	0.170431	0	0.90	Reservoir Dogs	297111	0.415966	Crime|Thriller	Eight men eat breakfast at a Los Angeles diner...
32127	567	257	3	0.5	0.257576	0.415084	95	0.25	Just Cause	2529009	0.428571	Mystery|Thriller	Paul Armstrong , a liberal Harvard professor o...
4064	75	118	3	0.5	0.257576	0.502574	27	0.25	If Lucy Fell	5649318	0.386555	Comedy|Romance	Joe MacGonaughgill and Lucy Ackerman are roo...

Figure II.11.Données utilisées

Les données sont utilisées pour l'apprentissage et le test du modèle proposé. 20% des données sont utilisées pour le test de l'algorithme implémenté et les évaluations restantes sont utilisées pour l'apprentissage.

```
Entrée [2]: apprentissage,test = train_test_split(bdd , test_size = 0.2)
```

```
Entrée [22]: apprentissage
```

Out[22]:

	user id	item id	rating	sexe	age	temp	Ecart	occupation	title	code	runtime	genre	
26934	474	696	3	M	51	887916330	0	executive	Butterfly Kiss	25955119	88	Thriller	He is t of espion
18889	343	283	4	M	43	876403212	91	engineer	New Jersey Drive	14311655	98	Crime|Drama	Ne Dr w hea
23009	407	180	4	M	29	875044597	0	engineer	Mallrats	236651	94	Comedy	The da the Clerks
22404	403	111	4	M	37	879785974	0	other	Taxi Driver	11943293	114	Drama|Thriller	A journe 1950s

Billion

FigureII.12.Diviser la base de données

IV. Fichiers utilisées

Les principaux fichiers utilisés sont :

- le fichier movies.csv. Il contient des informations sur les films (items) :
  - movieid : identifiant du film. Il est unique pour chaque film.
  - titre : contient le titre de chaque film.
  - genres : comporte les genres des films (Drama, action, comédie …).
- le fichier rating.csv. Il contient les informations suivantes:
  - userid : identifiant de l'utilisateur. Il est unique pour chaque utilisateur.
  - movieid : identifiant du film.
  - rating : la note donnée par un utilisateur à un film.
  - timestamps : represente le nombre de secondes qui s'est écoulé depuis le 1er janvier 1970. Par exemple, le timestamps de 26 mai 2016 à 22 : 35 : 45 , est 1464294945. [51]
- Le fichier data : contient plusieurs informations, telles que tconst ,titleType, primaryTitle ,originalTitle , isAdult , startYear , endYear , runtime , genre,
- Le fichier résume : contient le code wiki et le résume de chaque film.

V. Description des méthodes utilisées pour la recommandation

V.1. Filtrage collaboratif traditionnel

Le filtrage collaboratif basé item exploite les similarités entre les ressources. Il suggère la note qu'un utilisateur *u* pourra donner à un item *i* en s'appuyant sur les appréciations de *u* pour des items similaires à *i*. La corrélation entre deux items est calculée en se basant sur la similarité de leur historique des évaluations. Deux items sont similaires si plusieurs utilisateurs les ont notés de la même manière.

Un algorithme de filtrage collaboratif traditionnel suggère un film *i* à un utilisateur *u* en se basant sur les évaluations données à des films similaires. D'abord, l'ensemble des évaluations est représenté sous forme d'une matrice *user*×*item*. Le degré de similarité entre deux films *i* et *j* est calculé en utilisant le coefficient de corrélation de Pearson selon l'équation (1). Le résultat de cette étape est une matrice *item*×*item* contenant les degrés de corrélation entre tous les films (figure II.13).

$$corr_{i,j} = \frac{\sum_{u \in U}(R_{u,i} - \overline{R}_i)(R_{u,j} - \overline{R}_j)}{\sqrt{\sum_{u \in U}\left(R_{u,i} - \overline{R}_i\right)^2}\sqrt{\sum_{u \in U}\left(R_{u,j} - \overline{R}_j\right)^2}} \quad (1)$$

$R_{u,i}$ est l'évaluation de l'utilisateur u sur le film j.

$\overline{R}_i$ et $\overline{R}_j$ représentent les moyennes des évaluations des items i et j, respectivement.

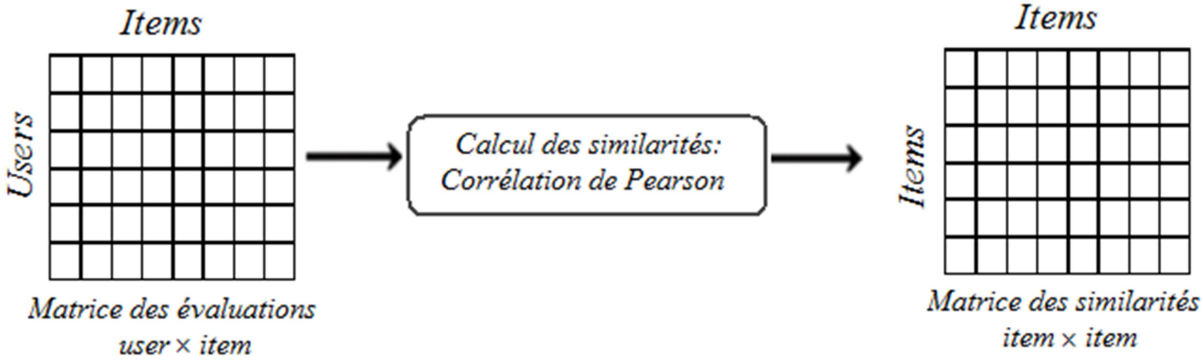


Figure II.13. Le calcul de similarité.

Une fois les corrélations entre les films sont calculées, la prochaine étape consiste à estimer pour l'usager u une valeur pour l'item i (figure II.14). La valeur prévue est basée sur la somme pondérée des notes de l'utilisateur u sur des films similaires à l'item i en termes d'évaluation. Ainsi cette valeur peut être calculée selon l'équation (2)

$$P_{u,i} = \frac{\sum_j^m R_{u,j} \times cor_{i,j}}{\sum_j corr_{i,j}} \quad (2)$$

m représente le nombre de films similaires à l'item i et qui sont déjà évalués par l'utilisateur u.

$R_{u,j}$ est l'évaluation de l'utilisateur u sur le film j.

$Corr_{i,j}$ est le degré de similarité entre les films i et j.

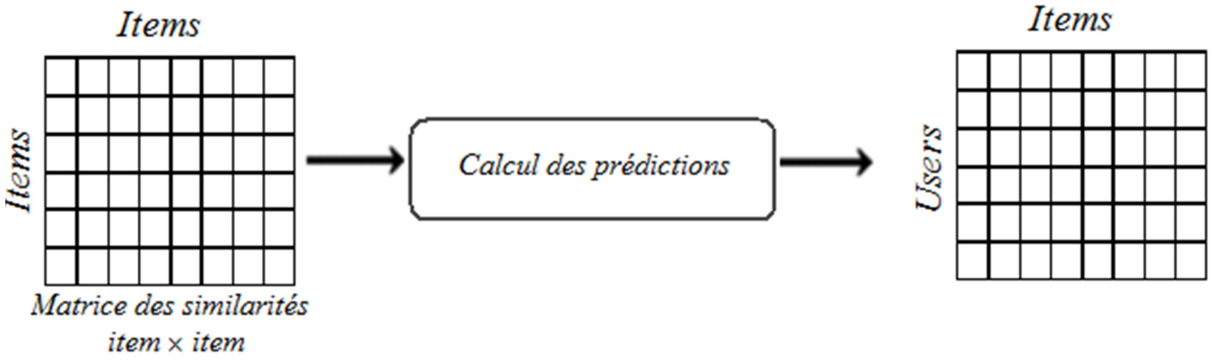


Figure II.14. Le calcul des prédictions.

Le tableau II.2 illustre un exemple de recommandation de films. Il résume les préférences des utilisateurs sur les films. La valeur 1 signifie que la personne n'a pas aimé le film et 5 désigne qu'elle l'a adoré. Le symbole - indique que les utilisateurs n'ont pas fourni de notes aux films. Par conséquent, le système doit suggérer à chacun une liste de films non vus pouvant l'intéresser.

	Film					
	Film 1	Film 2	Film 3	Film 4	Film 5	Film 6
User 1	3	1	2	-	5	5
User 2	3	1	2	-	5	5
User 3	3	-	2	3	5	5
User 4	-	5	3	3	1	1
User 5	-	5	-	3	2	1
User 6	3	-	1	3	-	1
User 7	3	5	2	4	2	-

Tableau II.2 Exemple de recommandation d'item.

V.2. Filtrage collaboratif basé LDA

L'allocation de Dirichlet latente (LDA : Latente Dirichlet Allocation) est une technique d'apprentissage automatique qui analyse les mots des textes dans les documents pour découvrir les thèmes qu'ils traitent. Les documents sont donc des mélanges de thèmes où un thème est une attribution de probabilité sur les mots. LDA a été largement étudie dans l'analyse de documents, la catégorisation de documents et le regroupement de documents. Elle a été introduite pour la première fois dans les systèmes de recommandations pour analyser le contexte dans les systèmes de recommandations basées sur le contenu et les systèmes de recommandations basées sur les tags. Ainsi, LDA est utilisé pour trouver la relation cachée entre les mots-clés des descriptions d'items et les tags d'items créés par les utilisateurs afin de proposer des objets en fonction des tags [52] .

V.2.1. Pré-traitement

Comme les résumés des films sont sous forme textuel, certains processus de traitement sont nécessaires, à savoir la conversion des majuscules en minuscules, la suppression des ponctuations et des mots vides. Les mots traités sont ensuite extrait afin de construire notre le dictionnaire du vocabulaire. Cette procédure de LDA utilise les packages Python, en particulier la librairie de traitement de langage NLTK ainsi que les librairies classiques pandas, numpy et scikit:

- NLTK [W5], une boîte à outils en langage naturel pour Python. Un package utile pour tout traitement du langage naturel.
- stop_words [W6], un package Python contenant des mots vides.
- Gensim [W7], un package de modélisation de sujet contenant le modèle LDA.

La première étape du traitement est la récupération et le nettoyage des données afin de pouvoir les utiliser ultérieurement dans le modèle.

Le pré-traitement du corpus utilisé comprend les étapes suivantes :

1. Récupération du corpus en téléchargeant des fichiers textes, par exemple.
2. La tokenization**,** qui désigne le découpage en mots des divers documents constituant le corpus
3. La normalisation et la construction du dictionnaire qui permet d'ignorer des détails inutiles (ponctuation, majuscules, conjugaison, etc.).

```
Entrée [1]: import pandas as pd
            import numpy as np
            from sklearn.model_selection import train_test_split
            bdd = pd.read_excel('BDDFinal.xlsx')
            bdd.head()
```

Figure II.15 .Insertion et lecture des données

a) Tokenization

La tokenisation est un processus de segmentation qui décompose une phrase en ses multiples éléments. Elle permet de découper le texte en mots. En d'autre termes, elle segmente le texte en entrée en unités linguistiques manipulables comme les mots, la ponctuation, les nombres, les données alphanumériques. Chaque élément correspond à un token qui sera utile à l'analyse.

b) Normalisation du texte

La normalisation est une étape de prétraitement qui comprend le nettoyage du texte, la conversion de la casse, la correction de l'orthographe, la suppression des mots inutiles, la lemmatisation, etc. :

- Nettoyage

Le nettoyage est la première étape souvent effectuée dans le traitement du texte. Il peut être divisé en trois tâches principales: la normalisation de la casse des lettres en conversant le texte en minuscules; la suppression des signes de ponctuation ; et la suppression des mots vides (*stopwords* en anglais). Ces derniers sont des mots très courants dans la langue étudiée ("il", "à", "la", etc.) qui n'apportent pas de valeur informative pour la compréhension du "sens" d'un texte. Il faut donc les supprimer parce qu'ils sont très fréquents et ralentissent le travail. Il existe dans la librairie NLTK une liste par défaut des stopwords dans plusieurs langues. Dans le cadre de ce travail, on supprime d'abord les mots les plus fréquents du corpus puis les stopwords fournis par NLTK.

Entrée [14]:
```
stemmer = PorterStemmer()
#fonction de lemmatisation
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text, pos='v'))
```

Entrée [15]:
```
def preprocess(text):
    result = []
    for token in gensim.utils.simple_preprocess(text):
        if token not in gensim.parsing.preprocessing.STOPWORDS and len(token) > 3:
            result.append(lemmatize_stemming(token))
    return result
```

Figure II.16.Nettoyage des données

- Le Stemming

Le Stemming ou racinisation est un procédé de transformation des flexions en leur radical ou racine. C'est un processus de suppression des affixes flexionnels et dérivatifs pour obtenir une forme de base à laquelle tous les mots originaux sont probablement liés.

- La lemmatisation

La lemmatisation ramène un texte à des mots-racines afin d'éliminer les différences dues à des formes particulières des mots. Ce processus permet de rapporter automatiquement des variations morphologiques ou graphiques parfois très grandes à leur lemme. Elle utilise un dictionnaire pour rechercher des mots et les remplacer par la racine (lemme). Un vocabulaire commun et largement utilisé en anglais est WordNet. Ainsi, la lemmatisation ne supprime les suffixes et les préfixes que si le mot résultant fait partie du dictionnaire.

c) Filtrer les motspeu fréquents

Le filtrage des mots qui n'apparaissent pas souvent dans le corpus peut-être bénéfique. Ces mots contiennent des fautes de frappe, ou ils sont des noms ou des mots rares. Ainsi, ils ne font qu'exploser le vocabulaire sans apporter de valeur informative pour la compréhension du sens d'un document.

V.2.2. Dictionnaire de vocabulaire

Après la phase de nettoyage et du prétraitement, on obtient un dictionnaire des mots représentant une nouvelle base des données des films avec le nombre d'occurrences de chaque mot. On représente donc chaque film par ce qu'on appelle un *bag-of-words* qui correspond à l'ensemble des mots que le film contient. En pratique, ça peut être effectué par un vecteur de fréquence d'apparition des différents mots utilisés. Une représentation bag-of-words classique sera donc celle dans laquelle on représente chaque film par un vecteur de la taille du vocabulaire et on utilise la matrice composée de l'ensemble de ces N items qui forment le corpus comme entrée de notre algorithme.

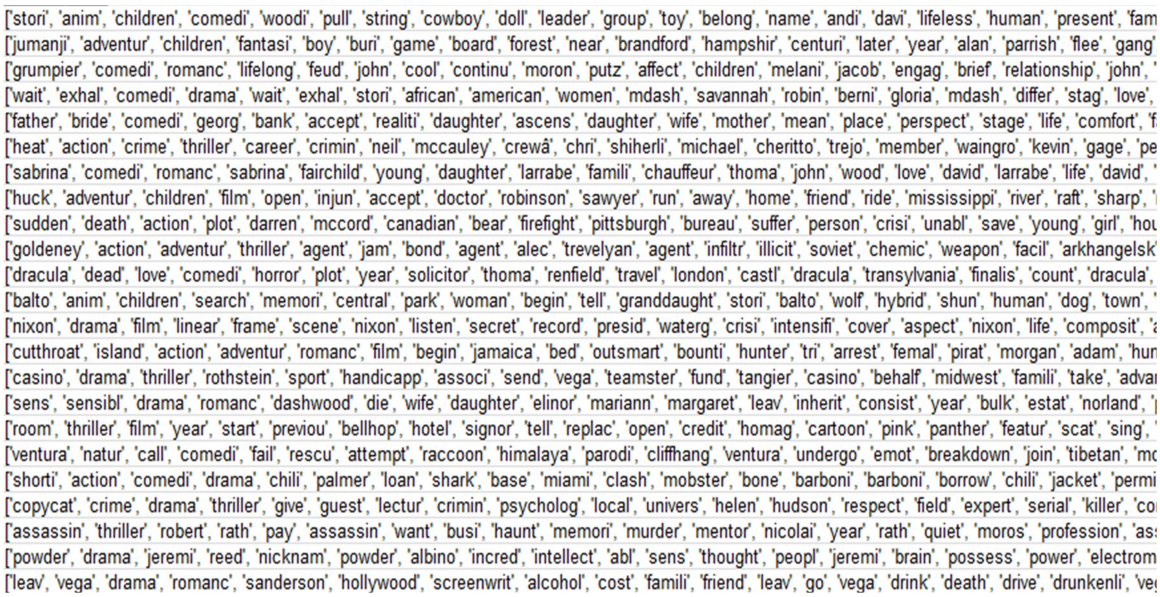


Figure II.17.Dictionnaire de vocabulaire

V.2.3. Extraction des thèmes

L'allocation de Dirichlet latente (LDA) est un modèle probabiliste génératif d'un corpus. L'idée de base est que les documents sont représentés comme des mélanges aléatoires de thèmes latents, où chaque thème est caractérisé par une distribution sur les mots. Le mélange de ces sujets latents pourrait être utilisé comme vecteurs de documents, la taille de vecteur peut être modifiée en fonction du nombre de sujets à extraire. Ainsi, le nombre de thèmes doit être défini avant d'exécuter l'algorithme [53].

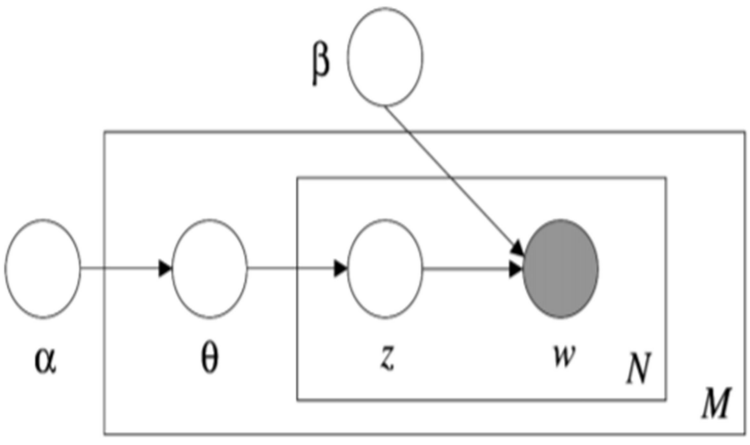


Figure II.18. Une représentation graphique de LDA [53].

Dans la figure II.18, les boîtes sont appelées «plaques» et ils représentent des répliques. La plaque extérieure représente les documents, et la plaque intérieure représente un choix répété de documents et de mots d'un document. [54].

LDA est un modèle d'apprentissage automatique non supervise qui reçoit en entrée des descriptions textuelles des items et fournit les thèmes en sortie. Le modèle indique également avec quel pourcentage chaque item appartient à chaque thème (figure II.19).

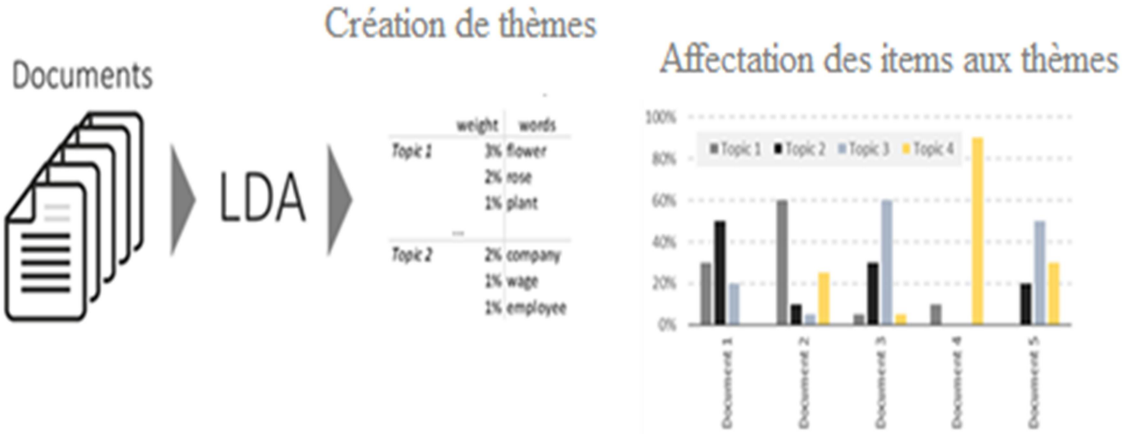


Figure II.19. Illustration du flux de travail de LDA [54].

```
Entrée [16]: doc_filtre = datatext['dictionnaire'].map(preprocess)

Entrée [24]: doc_filtre

Out[24]: 0      [stori, anim, children, comedi, woodi, pull, s...
         1      [jumanji, adventur, children, fantasi, boy, bu...
         2      [grumpier, comedi, romanc, lifelong, feud, joh...
         3      [wait, exhal, comedi, drama, wait, exhal, stor...
         4      [father, bride, comedi, georg, bank, accept, r...
                                    ...
         789    [star, map, drama, allegor, tale, concern, exi...
         790    [comedi, howard, brackett, like, english, lite...
         791    [confidenti, crime, film, noir, mysteri, thril...
         792    [soul, food, drama, soul, food, tell, eye, yea...
         793    [wishmast, horror, narrat, angu, scrimm, word,...
         Name: dictionnaire, Length: 794, dtype: object
```

Figure II.20. Extraction des thèmes

V.2.4. Implémentation de LDA

La classe du modèle LDA est décrite en détail dans [W8]. Les paramètres utilisés dans notre modèle sont :

num_topics : obligatoire – le modèle LDA nécessite que l'utilisateur détermine le nombre de thèmes à générer.

id2word : obligatoire – le modèle LDA nécessite un dictionnaire pour mapper les identifiants aux chaines.

Chunksize : la taille du bloc contrôle le nombre de documents traités à la fois dans l'algorithme d'apprentissage. L'augmentation de la taille du bloc permet l'accélération de l'apprentissage.

Passes : optionnel – représente le nombre d'itérations. Il contrôle la fréquence à laquelle nous répétons une boucle particulière sur chaque item. Plus le nombre d'époques est élevé, plus le modèle sera précis. Cependant, un nombre élevé peut augmenter le temps d'exécution de l'algorithme.

Eta : obligatoire – correspond à la distribution des mots par thème.

Alpha : obligatoire – est à la distribution des thèmes par item

```
Entrée [45]: lda_model = gensim.models.LdaMulticore(bow_corpus,
                                                    num_topics=60,
                                                    id2word=dictionary,
                                                    passes=2,
                                                    alpha=0.5,
                                                    eta=0.1)
```

Figure II.21.Implémentation de LDA

Notre modèle LDA est maintenant stocké sous le nom de ldamodel. Nous pouvons afficher nos thémes avec les méthodes print_topic et print_topics :

```
Entrée [22]: print(lda_model)

LdaModel(num_terms=1678, num_topics=60, decay=0.5, chunksize=2000)

Entrée [23]: for idx, topic in lda_model.print_topics(-1):
                 print('Topic: {} \nWords: {}'.format(idx, topic))

Topic: 0
Words: 0.011*"jack" + 0.009*"kill" + 0.006*"return" + 0.006*"tell" + 0.006*"friend" + 0.005
*"shoot" + 0.005*"tri" + 0.005*"father" + 0.005*"happi" + 0.005*"help"
Topic: 1
Words: 0.011*"tommi" + 0.008*"kill" + 0.007*"hous" + 0.007*"friend" + 0.006*"tell" + 0.005
*"begin" + 0.005*"thoma" + 0.005*"year" + 0.005*"plan" + 0.005*"famili"
Topic: 2
Words: 0.020*"kill" + 0.009*"shoot" + 0.008*"escap" + 0.008*"attempt" + 0.006*"return" + 0.0
06*"polic" + 0.006*"offic" + 0.005*"discov" + 0.005*"begin" + 0.005*"arriv"
Topic: 3
Words: 0.006*"tell" + 0.006*"film" + 0.005*"tri" + 0.005*"home" + 0.005*"kill" + 0.004*"yea
r" + 0.004*"go" + 0.004*"return" + 0.004*"come" + 0.004*"love"
Topic: 4
Words: 0.012*"town" + 0.009*"mountain" + 0.009*"tell" + 0.007*"film" + 0.006*"marri" + 0.006
*"time" + 0.005*"begin" + 0.005*"love" + 0.005*"girl" + 0.005*"life"
```

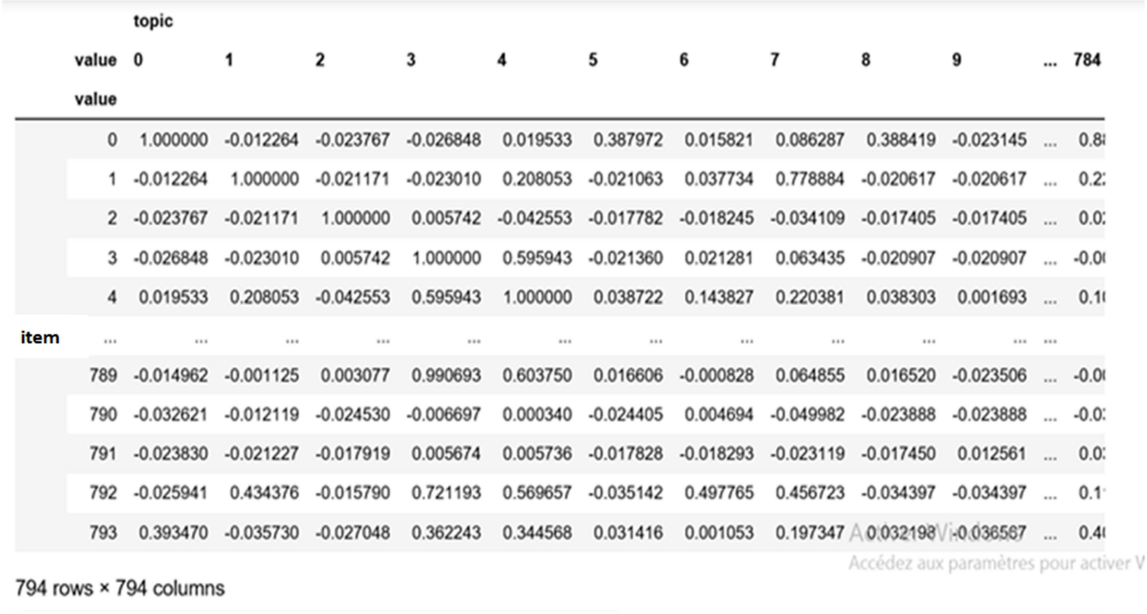

Figure II.22. Modèle obtenu après apprentissage

```
Entrée [24]: affichage = lda_model.get_document_topics(bow_corpus)

Entrée [25]: for item in affichage:
                 print(item)

[(2, 0.053604666), (8, 0.0757966), (14, 0.07194558), (19, 0.018081708), (22, 0.08149536), (2
9, 0.09026928), (30, 0.04134693), (33, 0.04886292), (43, 0.27801737), (54, 0.081047565)]
[(35, 0.8256718), (54, 0.019766074)]
[(8, 0.88737494)]
[(10, 0.08166123), (34, 0.034963854), (35, 0.56031615)]
[(2, 0.013648791), (10, 0.21270648), (17, 0.011573229), (19, 0.011331272), (22, 0.01449650
5), (25, 0.059624996), (28, 0.013315909), (29, 0.011679666), (30, 0.03889541), (32, 0.044935
685), (34, 0.012555586), (35, 0.018217627), (42, 0.012506849), (43, 0.068884075), (54, 0.021
927718), (55, 0.010483287), (57, 0.05348974)]
[(2, 0.78532904), (8, 0.034637272), (17, 0.011555188), (25, 0.010758586), (30, 0.012616253),
(35, 0.011733141)]
[(8, 0.016373402), (42, 0.3261766), (46, 0.31250665), (54, 0.0103791505), (57, 0.010620957)]
[(17, 0.010149269), (29, 0.5050457), (35, 0.053094044), (43, 0.10948501), (57, 0.057012256)]
[(2, 0.48975098), (35, 0.12583482), (36, 0.047570888), (43, 0.18469329)]
[(49, 0.8703643)]
[(19, 0.043439876), (29, 0.092780285), (54, 0.5191461), (57, 0.15286018)]
[(29, 0.87644106)]
```


Figure II.23.Pourcentage des thèmes par item



	topic											
	value	0	1	2	3	4	5	6	7	8	9	...	784
	value												
	0	1.000000	-0.012264	-0.023767	-0.026848	0.019533	0.387972	0.015821	0.086287	0.388419	-0.023145	...	0.8
	1	-0.012264	1.000000	-0.021171	-0.023010	0.208053	-0.021063	0.037734	0.778884	-0.020617	-0.020617	...	0.2
	2	-0.023767	-0.021171	1.000000	0.005742	-0.042553	-0.017782	-0.018245	-0.034109	-0.017405	-0.017405	...	0.0
	3	-0.026848	-0.023010	0.005742	1.000000	0.595943	-0.021360	0.021281	0.063435	-0.020907	-0.020907	...	-0.0
	4	0.019533	0.208053	-0.042553	0.595943	1.000000	0.038722	0.143827	0.220381	0.038303	0.001693	...	0.1
item	...	...	...	...	...	...	...	...	...	...	...	...	
	789	-0.014962	-0.001125	0.003077	0.990693	0.603750	0.016606	-0.000828	0.064855	0.016520	-0.023506	...	-0.0
	790	-0.032621	-0.012119	-0.024530	-0.006697	0.000340	-0.024405	0.004694	-0.049982	-0.023888	-0.023888	...	-0.0
	791	-0.023830	-0.021227	-0.017919	0.005674	0.005736	-0.017828	-0.018293	-0.023119	-0.017450	0.012561	...	0.0
	792	-0.025941	0.434376	-0.015790	0.721193	0.569657	-0.035142	0.497765	0.456723	-0.034397	-0.034397	...	0.1
	793	0.393470	-0.035730	-0.027048	0.362243	0.344568	0.031416	0.001053	0.197347	0.032198	-0.036567	...	0.4

794 rows × 794 columns

Figure II.24. Matrice item ×topic

Pour résumé, LDA suppose que les items sont produits à partir d'un mélange de thèmes. Ces thèmes génèrent ensuite des mots en fonction de leur distribution de probabilité. En d'autres termes, LDA suppose qu'un item est créé à partir des étapes suivantes :

1- Déterminer le nombre de mots dans un item.

2- Déterminer le mélange de thèmes dans cet item. Par exemple, la première vidéo est représentée comme suit : 5 % du thème 2, 7 % du thème 8, 7 % du thème 14, 1 % du thème 19, 8% du thème 22, 9 % du thème 29, 4% du thème 30, 4 % du thème 33, 27 % du thème 43 et 8% du thème 50.

V.2.5. Apprentissage de l'algorithme LDA

On a fait l'apprentissage de LDA en utilisant un algorithme de filtrage base Item pour trouver les meilleures valeurs de ses paramètres. En effet, le calcul des similarités est basé sur les vecteurs d'appartenance des items aux différents thèmes.

```
Entrée [41]: sortedSimilarMovies=sorted(similarmovies,key=lambda x:x[1],reverse=True)[1:]

Entrée [42]: sortedSimilarMovies

Out[42]: [(209, 0.9262629050585863),
          (253, 0.8881831743231172),
          (378, 0.8700227106390857),
          (624, 0.8700048121196116),
          (691, 0.8605414240211531),
          (339, 0.8404394076835309),
          (8, 0.8331312334188686),
          (491, 0.8267359678002694),
          (31, 0.8228849530621183),
          (549, 0.8217520668362334),
          (81, 0.8025764406035905),
          (662, 0.7847763456381299),
          (429, 0.7647280357917489),
          (460, 0.7616748486098325),
          (399, 0.7455482276551327),
          (568, 0.7416527764692858),
          (221, 0.7401497725919799),
          (234, 0.7397914092329145),
          (365, 0.7354437176630462),
          (61, 0.7346700508022054),
```

Figure II.25. Calcul des degrés de similarité

```
#remplacer la ligne du dessus par  -->   RMSE=sqrt(RMSE/len(test))
0.9628161214017612

item id :
542
user id :
541
rating :
1
-0.7469341548986979

item id :
13
user id :
365
rating :
3
0.6431725691104262

RMSE :
1.0540052693410549
```

Figure II.26.Calcul des prédictions

V.2.6. Calcul de l'erreur

Dans notre travail on a utiliséla racine de l'erreur moyenne quadratique (Root Mean Squared Error : RMSE) pour calculer l'erreur commise par le modèle proposé selon l'équation (3). RMSE calcule la racine carrée de l'erreur entre l'évaluation prédite par le système et la note réelle donnée par l'utilisateur.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(p_i - reel_i)^2}{N}} \qquad (3)$$

- **N** : le nombre total des évaluations prédites
- **P i** : la valeur prédite
- **Réel i** : la valeur réelle

```
Entrée [66]: from sklearn.metrics import mean_squared_error
             from math import sqrt
             rmse = sqrt(mean_squared_error(rating,pui))
             print(rmse)
```

Figure II. 27. Calcul de l'erreur

V.3. Réseau de neurones multicouches (MLP)

La réalisation du réseau de neurones comprend à la fois la partie conception (le but est de sélectionner la meilleure architecture) et la partie calcul numérique pour l'apprentissage du réseau. Un MLP peut avoir n'importe quel nombre de couches, mais pour optimiser son fonctionnement d'une part et réduire au maximum le temps de calcul d'autre part, on doit rechercher la meilleure architecture, en terme de nombre de couches et nombre de neurones dans chaque couche. À partir de l'architecture choisie et de la base d'apprentissage disponible, on détermine les poids optimaux.

V.3.1. Propriétés générales d'un MLP:

Les principales propriétés d'un MLP sont:

- Un neurone est connecté en entrée à tous les neurones de la couche suivante.
- La sortie d'un neurone est connectée à tous les neurones de la couche précédente.
- Il n'y a aucune connexion entre les neurones d'une même couche.

- Il n'y a pas de bouclage des sorties vers les entrées.
- Le nombre de couches cachées est quelconque, ainsi que le nombre de neurones dans ces couches cachées.

V.3.2. Déterminer le nombre de neurones d'entrée et de sortie

Le réseau utilisé dans le cadre de ce travail, possède une couche d'entrée, une couche de sortie, et des couches cachées (intermédiaires). Le nombre de neurones de la couche d'entrée dépend du nombre de caractéristiques utilisées. Les entrées de la première couche représentent les utilisateurs et les films : on utilise 50 neurones d'entrées pour représenter les degrés d'appartenance de l'item aux différents thèmes et 100 neurones d'entrée pour représenter le code one-hot de chaque utilisateur. On utilise également la couche d'intégration (embedding) pour utiliser une représentation efficace et dense dans laquelle des utilisateurs similaires ont un codage similaire.Une incorporation est donc un vecteur dense de valeurs à virgule flottante. Ces valeurs sont apprises par le modèle lors de l'apprentissage. Les neurones de sortie représentent les cinq valeurs (1 à 5) utilisées pour l'évaluation des films.

V.3.3.Définir le nombre de couches cachées

En plus des couches d'entrée et de sortie, on doit également déterminer le nombre de couches intermédiaires. En effet, sans couches cachées, l'adaptation du réseau sera très limitée.

V.3.4.Fixer le nombre de neurones pour chaque couche cachée

Chaque neurone supplémentaire permet de prendre en considération des caractéristiques spécifiques des neurones d'entrée. Il suffit donc d'ajouter un nombre de neurones suffisant au niveau de chaque couche cachée pour adapter le réseau à notre problème.

V.3.5.Choisir la fonction d'activation

Dans le cadre de ce travail, on considère la fonction sigmoïde pour le passage de la couche d'entrée aux couches intermédiaires ; et la fonction ReLU pour le passage de la couche cachée à la couche de sortie.

V.3.6.Apprentissage du réseau

L'apprentissage est supervisé car on dispose d'un comportement de référence précis qu'on désire inculquer au réseau. Le réseau est donc capable de mesurer la différence entre son comportement de référence, et de corriger ses poids de façon à réduire cette erreur.

On présente au réseau l'ensemble de données d'entrée et on lui demande de modifier ses poids afin de trouver la sortie désirée : la première étape consiste donc à *propager* les entrées vers l'avant jusqu'à obtenir une sortie calculée par le réseau. Cette dernière sera ensuite comparée avec la sortie désirée. Les poids du réseau seront donc modifiés de telle sorte qu'à la prochaine itération, l'erreur entre la sortie calculée et la sortie désirée soit minimisée. L'erreur est *rétro propagée* vers l'arrière jusqu'à la couche d'entrée pour calculer sa contribution sur la modification des poids synaptiques.

L'apprentissage du réseau MLP est défini comme un problème d'optimisation qui consiste à trouver les coefficients qui minimisent une fonction d'erreur globale (fonction coût).

V.3.7. Fonction Coût

La fonction coût permet de mesurer l'écart entre les réponses réelles du réseau et les sorties désirées. On a utilisé dans ce mémoire l'erreur absolue moyenne (Mean Absolute Error : MAE). Cette mesure est l'une des métriques d'évaluation les plus populaires. Elle est utilisée pour évaluer le niveau d'exactitude des recommandations fournies par le système. Elle calcule l'écart entre les prédictions et les notes réelles selon l'équation suivante. Plus la valeur de MAE est petite, plus l'erreur sera faible.

$$MAE = \frac{\sum_{i=1}^{N} |prediction_i - reel_i|}{N} \qquad (4)$$

N est la taille de la base d'apprentissage. $prediction_i$ et $reel_i$ représentent respectivement l'évaluation prédite et l'évaluation réelle.

```
Entrée [25]: from sklearn.metrics import mean_absolute_error

             print(mean_absolute_error(test.rating, model.predict([test.userId, test.movieId])))
```

Figure II. 28. Fonction coût

V.3.8. Algorithme d'optimisation

Un algorithme d'optimisation est utilisé afin d'estimer les poids du réseau pour lesquels la fonction coût est minimale. Le principe de ces algorithmes est de se mettre au point de départ, de trouver la direction qui permet la réduction du coût dans l'espace des

paramètres, puis de faire un pas dans cette direction. Le point obtenu devient le point de départ et le même processus se répète tant *qu'un critère d'arrêt* n'est pas *satisfait.* On a utilisé dans le cadre de ce travail Adam qui est l'un des algorithmes d'optimisation les plus récents [62 55]. Il peut être utilisé à la place de la procédure classique de descente de gradient pour mettre à jour, de façon itérative, les poids du réseau en se basant sur les données d'entraînement. Sa particularité est de calculer des *estimations adaptatives des moments* :tant que le gradient est dans la même direction que ceux précédents, on accélère la descente de la courbe. Les avantages attrayants de l'utilisation d'Adam sur des problèmes d'optimisation non convexes, sont comme suit:

- Simple à mettre en œuvre.

- Efficacité informatique.

- Peu de mémoire requise.

- Mise à l'échelle invariante à diagonale des gradients.

- Convient bien aux problèmes importants en termes de données et / ou de paramètres.

- Convient aux objectifs non stationnaires.

- Convient aux problèmes avec des gradients très bruyants / ou clairsemés.

- Les hyper-paramètres ont une interprétation intuitive et nécessitent généralement peu de réglage.

```
model = keras.Model([user_input, movie_input], result)
opt = keras.optimizers.Adam(lr = 0.001)
model.compile(optimizer=opt,loss= 'mean_absolute_error')
```

Figure II. 29. l'algorithme d'optimisation ADAM

V.3.9. Notion de Dropout

Le Dropout (le décrochage) est l'une des techniques de régularisation les plus efficaces et les plus couramment utilisées pour les réseaux de neurones [56]. Elle est appliquée pour empêcher le sur-ajustement sur les données d'apprentissage en abandonnant des unités dans un réseau de neurones.L'idée est d'éteindre au hasard certains neurones à chaque itération avec une probabilité p. Toutes les connexions liées à ces neurones ne sont pas prises en considération, et seulement les poids des autres neurones du réseau sont mis à

jour. Chaque neurone ne voit donc qu'une partie de l'ensemble de données, ce qui permet de conserver la capacité distinctive du réseau

a) Réseau neurone standard b) Après l'application de droupout

Figure II.30. Un réseau de neurones lors de l'application de la technique de DroupOut

L'architecture générale du réseau de neurones utilisé dans ce travail, peut être résumée comme suit :

- La première couche comporte des neurones pour représenter les utilisateurs et les items.
- On utilise la couche d'intégration (embedding) pour obtenir des vecteurs spécifique pour chaque utilisateur.
- On utilise les fonctions dropout pour régulariser l'apprentissage.
- Les deux vecteurs (utilisateur et item) sont concaténés par la fonction « fully connected layer ». Les couches entièrement connectées dans un réseau neurone sont les couches où toutes les entrées d'une couche sont connectées à chaque unité d'activation de la couche suivante. Dans les modèles d'apprentissage automatique les plus courants, les dernières couches sont des couches entièrement connectées qui compilent les données extraites par les couches précédentes pour former la sortie finale. Chaque neurone d'une couche reçoit une entrée de tous les neurones présents dans la couche précédente. Ils sont donc densément connectés. En d'autres termes, la couche dense est une couche entièrement connectée, ce qui signifie que tous les neurones d'une couche sont connectés à ceux de la couche suivante.
- A la fin, on utilise l'algorithme ADAM pour l'optimisation du modelé et la fonction MAE pour tester sa performance.

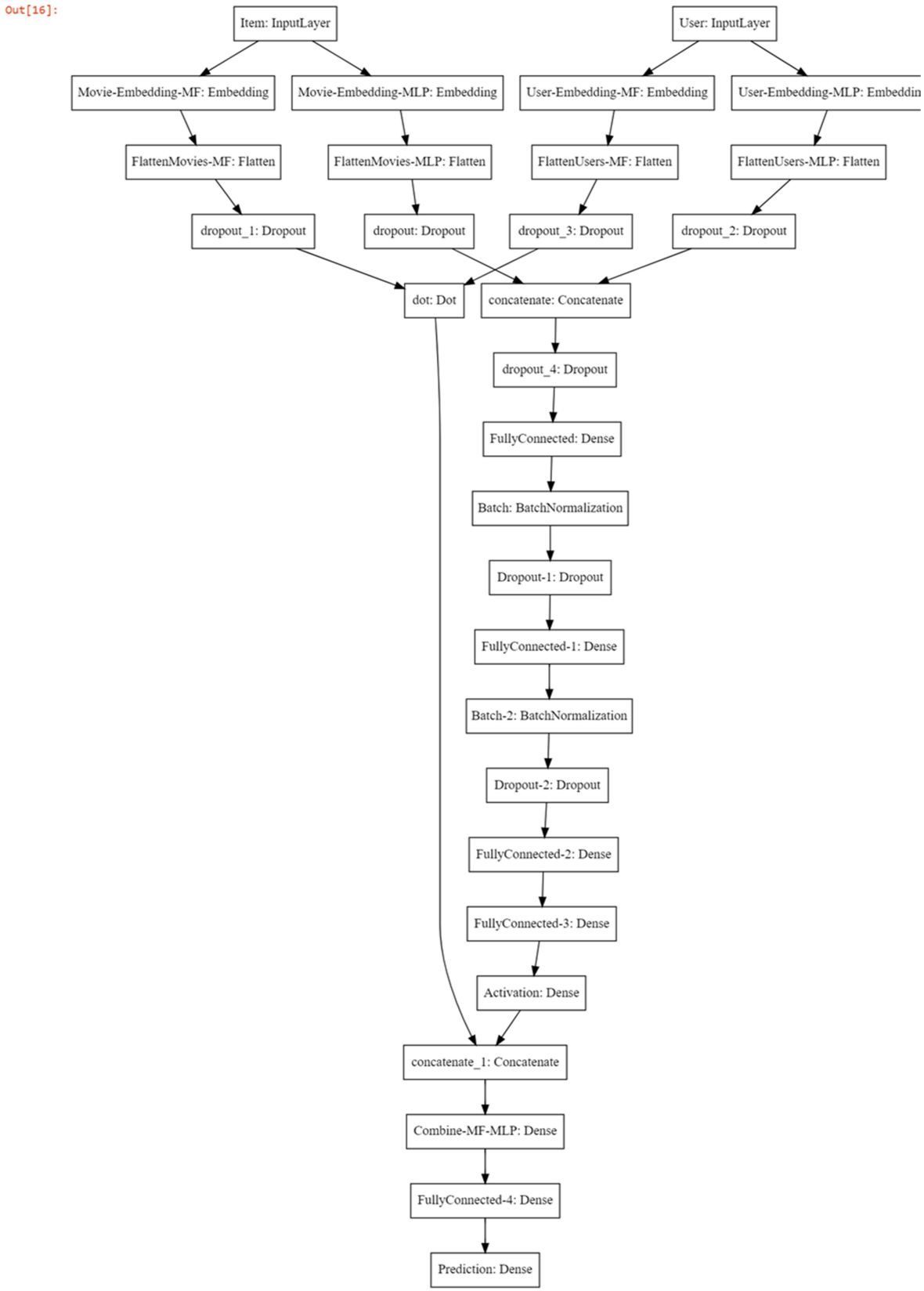


Figure II. 31. Organigramme du système de recommandation proposé

V.3.10. Les Top N recommandations

Finalement les préférences prédites par notre système sont triées dans l'ordre décroissant. Ainsi, la liste de recommandations est constituée des N premiers items qui sont les top-N des préférences prédites.

```
Entrée [43]: i=0
print ('the Top 5 similar movies to '+targetmovie+ ' are : ')
for element in sortedSimilarMovies:
    print(getTitlefromIndex(element[0]))
    i=i+1
    if i>=5:
        break
```

```
the Top 5 similar movies to Batman  are :
Castle Freak
Street Fighter
Spy Hard
Ran
Santa with Muscles
```

Figure II.32. Top N recommandations

VI. CONCLUSION

Dans ce chapitre, nous avons présenté les composantes de base de notre système. Dans le chapitre suivant, nous allons aborder les expérimentations et les tests de performance du modèle proposé, ainsi que les résultats obtenus.

Chapitre III :

Implémentation

I. INTRODUCTION

Dans ce chapitre, nous exposons l'implémentation de l'approche proposée dans le cadre d'un système de recommandation basé sur l'apprentissage profondenprésentant les outils utilisés ainsi que les résultats expérimentaux. Nous commençons tout d'abord par la présentation des ressources, du langage et de l'environnement de développement puis les différents tests effectués.

II. ENVIRONNNEMENT DE DEVELOPPEMENT

II.1. Environnement matériel

Le système de recommandation présenté dans ce chapitre est développé sur un Intel Core I7, avec une vitesse de 2.50 GHz, doté d'une capacité mémoire de 8G de RAM, et une carte graphique Intel (R) HD. Graphics 520.

II.2. Environnement logiciel

Nous avons implémenté le système proposé sous Windows 10 professionnel du type 64 bits. Ainsi, nous avons utilisé différents outils et langages de programmation qui seront présentés dans la section suivante.

II.2.1. Anaconda

Anaconda est une distribution libre et open source des langages de programmation python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique). Il vise à simplifier la gestion des paquets et de déploiement. Il permet ainsi d'éviter les erreurs d'installation des packages que ce soit pour Windows ou Linux [W2].

II.2.2. Jupyter

Le notebook Jupyter est un environement HTML, manipulable interactivement dans un navigateur web. Il peut rassembler du texte, des images, des formules mathématiques et du code informatique exécutable. Il est initialement développé pour les langages de programmation Julia, Python et R ; et il peut supporter près de 40 langages différents.

II.2.3. Python

Python est un langage de programmation puissant et relativement simple à apprendre. Il dispose de structures de données de haut niveau et permet la programmation orientée objet. C'est un langage interprété et multiplateforme, fonctionnant sur de nombreux systèmes d'exploitation (Windows, Mac OS X, Linux, Android, iOS) depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs [57]. Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines telsque l'apprentissage automatique et l'analyse de données. Dans le cadre de notre travail, nous avons utilisé la version 3.6 de python.

II.2.4.Bibliothèques utilisées

Afin d'implémenter le modèle proposé, nous avons utilisé plusieurs bibliothèques de python comme pandas,numpy,matplotlib, etc.

II.2.4.1. Tensorflow

TensorFlow est un framework Google avec lequel on peut créer des modèles d'apprentissage en profondeur. C'est une bibliothèque d'intelligence artificielle open source, qui utilise des graphes de flux de données pour construire des modèles. TensorFlow est populaire, gratuit et simple à utiliser. Il permet la création de réseaux de neurones à grande échelle comportant de nombreuses couches. Il est utilisé principalement pour la classification, la perception et la prédiction.

II.2.4.2. Keras

Kerasest une librairie open source pour l'apprentissage profond. Elle est écrite en Python et interfaçable avec TensorFlow, CNTK et Theano. Elle a été développée par François Chollet (Software Engineer @ Google) afin de permettre des expérimentations rapides.

II.2.4.3. Scikit-Learn

Scikit-Learnestla principale bibliothèque d'outils dédiés à l'apprentissage automatiqueet à la data-science dans l'univers Python. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom ParisTech.

III. CORPUS D'EVALUATION UTILISE

Afin d'obtenir un indice de performance correspondant à la capacité du système proposé à produire de bonnes recommandations, des tests sont effectuées en utilisant une base de données réelles. Le choix du corpus d'évaluation est très important, du fait qu'il est généralement difficile de trouver une bonne base qui peut être utilisée dans toutes les applications. Dans le cadre des systèmes de recommandation, il existe une multitude de bases de données: base de films, base de blagues, base de restaurants, base d'articles, etc. Dans le cadre de ce travail, le corpus *MovieLens* est utilisé pour faire les différents tests. En effet, cette base présente est à la fois publiquement disponible et facilement exploitable. Elle est également assez grande et dense. En outre, elle est largement utilisée par la communauté scientifique.

L'ensemble des évaluations de la base *MovieLens* est divisé en cinq ensembles d'apprentissage (*u1.base*, *u2.base*, *u3.base*, *u4.base* et *u5.base*) contenant les données d'apprentissage utilisées par les applications et cinq ensembles de test (*u1.test*, *u2.test*, *u3.test*, *u4.test* et *u5.test*) contenant les données qui servent à l'évaluation des algorithmes. Les ensembles d'apprentissage et de test contiennent respectivement 80 % et 20% des votes globaux. L'ensemble complet des votes est stocké dans le fichier *u.data*. Ce dernier contient 100000 votes créés par 943 utilisateurs sur 1682 films. Il est présenté sous forme d'une liste dont les informations sont *user id*, *movie id*, *rating* e t*timestamp* (Figure III.33.). *user id* est le code attribué à l'utilisateur et *movie id* représente le code donné au film. *rating* est l'évaluation donné à *movie id* par *user id*. *timestamps* est le nombre des secondes entre la date d'évaluation de l'item et le 1er janvier 1970.

1	1	5	874965758
1	2	3	876893171
1	3	4	878542960
1	4	3	876893119
1	5	3	889751712
1	7	4	875071561
1	8	1	875072484
1	9	5	878543541
1	11	2	875072262
1	13	5	875071805
1	15	5	875071608
1	16	5	878543541
1	18	4	887432020
1	19	5	875071515
1	21	1	878542772
1	22	4	875072404
1	25	4	875071805
1	26	3	875072442

Figure III.33. Fichier *u.data*.

Les évaluations sont des entiers qui correspondent à une échelle de 1 jusqu'à 5. La valeur 1 signifie que le film est *très mauvais*, 2 pour *mauvais*, 3 pour *moyen*, 4 pour *bien* et 5 pour dire que le film est *excellent*. Une grande proportion des votes sont des 3 et 4. Il y a globalement peu de votes correspondant à 1 et 2 (tableau III.3).

Base / Evaluation	1	2	3	4	5
U.data	6,11%	11,37%	27,14%	34,17%	21,20%
U1.data	5,90%	11,47%	27,45%	34,25%	20,93%
U2.data	6,06%	11,48%	27,26%	34,12%	21,07%
U3.data	6,15%	11,43%	26,95%	34,05%	21,40%
U4.data	6,24%	11,21%	26,97%	34,26%	21,31%
U5.data	6,19%	11,24%	27,08%	34,19%	21,29%
U1.test	6,96%	10,96%	25,91%	33,90%	22,27%
U2.test	6,29%	10,93%	26,67%	34,41%	21,71%
U3.test	5,94%	11,10%	27,90%	34,66%	20,40%
U4.test	4,59%	11,98%	27,85%	33,83%	20,75%
U5.test	5,79%	11,89%	12,39%	34,10%	20,83%

Tableau III.3. Distribution des données dans la base *MovieLens* [58].

Outre les évaluations des utilisateurs sur les films, cette collection est enrichie par des informations décrivant à la fois les utilisateurs et les films. Les informations sur les films sont contenues dans le fichier *u.item* (figure III.34.) : movie id | movietitle | release date | video release date | IMDb URL | unknown | Action | Adventure| Animation | Children's | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir | Horror |Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western.

```
62|Stargate (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Stargate%20(1994)|0|1|1|0|0|0|0|0|0|0|0|0|0|0|1|0|0|0
63|Santa Clause, The (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Santa%20Clause,%20The%20(1994)|0|0|0|0|1|1|0|0
|0|0|0|0|0|0|0|0|0|0
64|Shawshank Redemption, The (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Shawshank%20Redemption,%20The%20(1994)
|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0
65|What's Eating Gilbert Grape (1993)|01-Jan-1993||http://us.imdb.com/M/title-exact?What's%20Eating%20Gilbert%20Grape%20
(1993)|0|0|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
66|While You Were Sleeping (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?While%20You%20Were%20Sleeping%20(1995)
|0|0|0|0|0|1|0|0|0|0|0|0|0|1|0|0|0|0
67|Ace Ventura: Pet Detective (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Ace%20Ventura:%20Pet%20Detective%20
(1994)|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0
68|Crow, The (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Crow,%20The%20(1994)|0|1|0|0|0|0|0|0|0|0|0|0|0|1|0|1
|0|0
69|Forrest Gump (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Forrest%20Gump%20(1994)|0|0|0|0|0|1|0|0|0|0|0|0|0|0
|1|0|0|1|0
70|Four Weddings and a Funeral (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Four%20Weddings%20and%20a%20Funera
l%20(1994)|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0|0|0|0
71|Lion King, The (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Lion%20King,%20The%20(1994)|0|0|0|1|1|0|0|0|0|0|0
|0|1|0|0|0|0|0|0|0
72|Mask, The (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Mask,%20The%20(1994)|0|0|0|0|0|1|1|0|0|1|0|0|0|0|0|0|0
|0|0
73|Maverick (1994)|01-Jan-1994||http://us.imdb.com/M/title-exact?Maverick%20(1994)|0|1|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|1
74|Faster Pussycat! Kill! Kill! (1965)|01-Jan-1965||http://us.imdb.com/M/title-exact?Faster%20Pussycat!%20Kill!%20Kill!%
20(1965)|0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0
```

Figure III.34. Fichier *u.item*.

Les 19 derniers champs correspondent aux genres des items. La Liste des genres est stockée dans le fichier *u.genre* (figure III.35.). Dans le fichier *u.item*, la valeur *1* indique que le film fait partie de ce genre tandis que *0* signifie que l'item n'appartient pas à ce genre. Les *movie id* sont ceux utilisés dans le fichier *u.data*.Un film peut appartenir à plusieurs genres. Dans le cadre de ce manuscrit, les genres sont utilisés afin de décrire le contenu des films.

```
unknown|0
Action|1
Adventure|2
Animation|3
Children's|4
Comedy|5
Crime|6
Documentary|7
Drama|8
Fantasy|9
Film-Noir|10
Horror|11
Musical|12
Mystery|13
Romance|14
Sci-Fi|15
Thriller|16
War|17
Western|18
```

Figure III.35. Fichier *u.genre*.

IV. Résultats et Discussion

IV.1. Apprentissage de la méthode LDA

L'objectif principal de cette étape est le réglage des différents paramètres de LDA ainsi que la sélection de la meilleure valeur du degré de similarité afin de minimiser l'erreur RMSE. Par conséquent, différents tests ont été réalisés. Le tableau III.4 représente la variation de RMSE en fonction du degré de similarité et du paramètre alpha. Cette expérimentation est réalisée avec Num_topics = 50, passes = 10, eta = 0.1 et alpha ∈ [0.01, 0.50].

	RMSE								
Similarité S>=	alpha = 0.01	alpha = 0.05	alpha = 0.10	alpha = 0.15	alpha = 0.25	alpha = 0.30	alpha = 0.35	alpha = 0.40	alpha = 0.50
0,1	1,077	1,054	1,071	0,808	0,976	0,999	1,121	1,289	1,361
0,2	1,269	1,109	1,034	0,948	1,246	1,363	0,934	1,074	1,496
0,3	1,135	1,172	1,274	1,046	1,121	0,651	1,038	1,126	1,209
0,4	1,804	1,242	1,118	0,752	1,047	0,998	1,166	0,965	1,005
0,5	1,290	1,154	1,131	1,089	1,344	1,153	1,062	0,837	1,014
0,6	0,618	0,728	0,705	1,124	0,693	1,099	1,091	1,140	1,355
0,7	**0,984**	**0,441**	**0,765**	**0,491**	**0,878**	**0,778**	**1,094**	**0,948**	**0,849**
0,8	1,230	0,894	0,535	0,410	0,682	1,015	1,709	1,731	1,716
0,9	1,746	1,730	1,720	1,304	1,031	2,912	2,919	3,294	1,549

Tableau III .4: Les résultats de RMSE pour le paramètre alpha

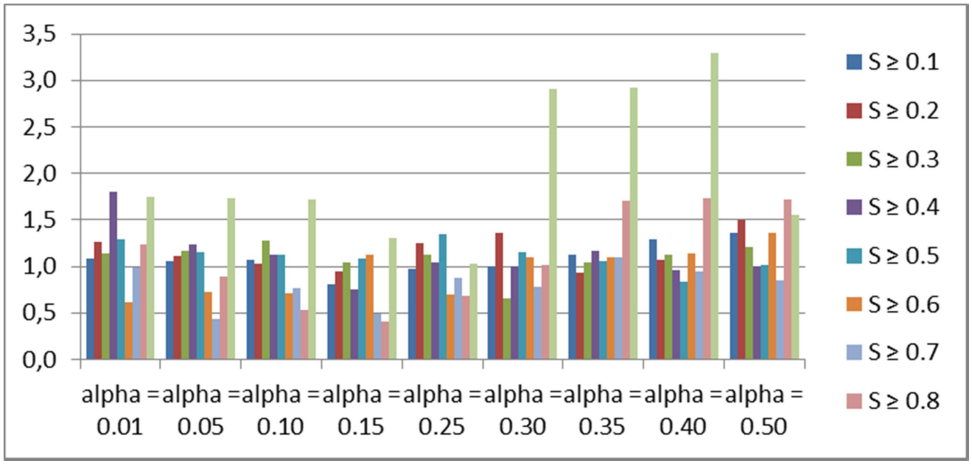


Figure III.36 : Représentation graphique du tableau III.4

Le tableau III.5 représente la variation de RMSE en fonction du degré de similarité et du paramètre eta. Cette expérimentation est réalisée avec Num_topics = 50, passes = 10, alpha = 0.1 et eta∈ [0.01, 0.50].

	RMSE								
Similarité S>=	eta = 0.01	eta = 0.05	eta = 0.10	eta = 0.15	eta = 0.25	eta = 0.30	eta = 0.35	eta = 0.40	eta = 0.50
0,1	1,161	1,160	1,034	0,925	2,974	1,052	1,100	0,871	1,026
0,2	1,104	1,029	1,007	1,108	1,103	1,626	1,011	1,026	0,872
0,3	0,885	1,802	1,050	1,047	0,786	1,131	0,996	0,979	0,960
0,4	0,994	1,875	0,845	2,192	0,925	2,045	0,875	0,537	1,004
0,5	1,190	1,066	1,013	2,014	0,851	0,583	1,102	1,028	1,031
0,6	**1,097**	**0,754**	**1,274**	**1,455**	**0,836**	**1,102**	**0,790**	**1,020**	**2,054**
0,7	0,895	1,111	1,033	2,112	1,146	1,132	0,947	1,020	2,212
0,8	0,911	1,839	0,969	0,511	1,323	0,745	0,547	1,699	3,294
0,9	0,818	1,043	1,907	0,899	0,505	0,865	0,834	1,481	3,124

Tableau III.5 : Les résultats de RMSE pour le paramètre eta

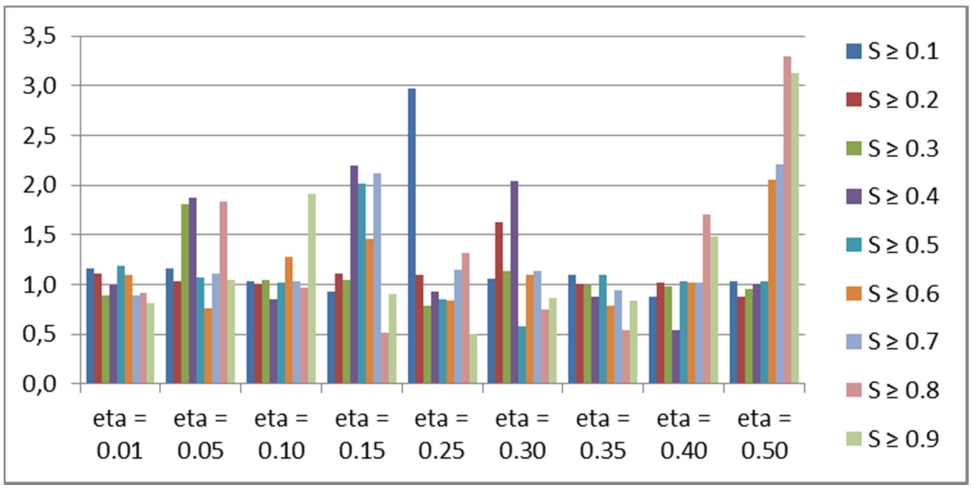


Figure III.37 : Représentation graphique du tableau III.5

Le tableau III.6 représente la variation de RMSE en fonction du degré de similarité et du nombre de thèmes. Cette expérimentation est réalisée avec alpha = 0.5,eta = 0.1, passes = 10 et un nombre de thèmes (num_topic) qui varie entre 20 et 60.

	RMSE								
Similarité S>=	num_topics = 20	num_topics = 25	num_topics = 30	num_topics = 35	num_topics = 40	num_topics = 45	num_topics = 50	num_topics = 55	num_topics = 60
0,1	3,040	3,053	3,058	3,863	2,861	2,143	1,361	1,524	1,727
0,2	3,053	2,969	2,117	3,002	2,451	2,115	1,496	1,112	1,717
0,3	2,154	2,722	2,069	2,235	2,001	2,096	1,209	1,889	1,562
0,4	2,119	2,159	1,855	1,887	1,996	1,924	1,005	1,975	1,235
0,5	1,138	1,764	1,131	1,977	1,913	1,623	1,014	1,547	1,417
0,6	1,101	1,046	1,068	1,167	1,778	1,430	1,355	1,985	1,117
0,7	**0,918**	**0,991**	**1,109**	**1,095**	**1,451**	**1,493**	**0,849**	**1,456**	**1,036**
0,8	1,067	0,889	1,014	1,028	0,995	0,997	1,716	1,031	0,997
0,9	1,036	0,928	0,979	0,850	0,594	1,443	1,549	0,971	0,852

Tableau III.6. : Les résultats de RMSE pour le paramètre num_topics

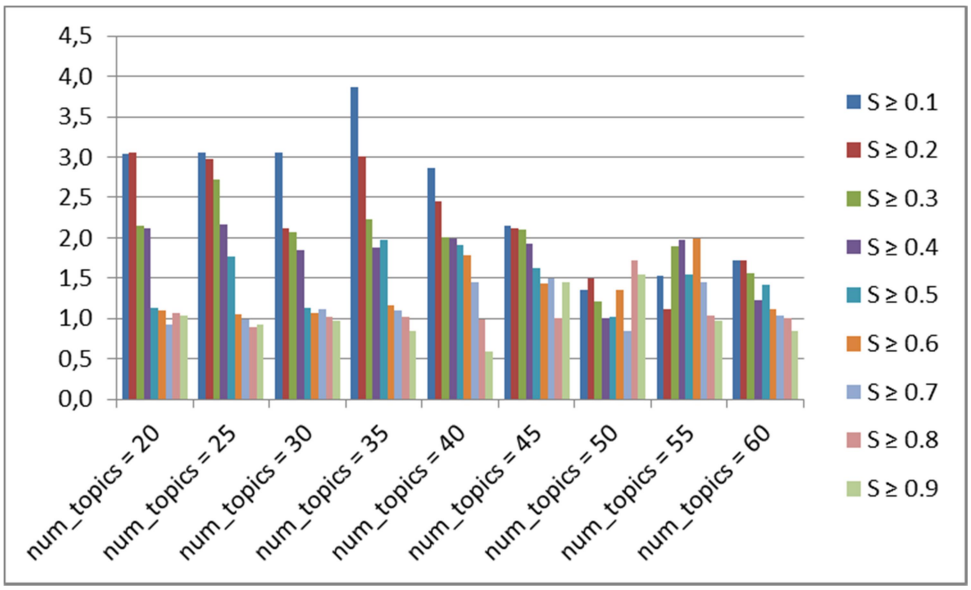


Figure III.38 : Représentation graphique du tableau III.6

Discussions des résultats

Selon les résultats obtenus, on peut constater queles meilleurs paramètres du modèle LDA utilisé dans le cadre de ce travail sont : alpha = 0.5, eta = 0.1et nom_topic = 50. On peut également remarquer que les meilleurs résultats sont obtenus avec un seuil de similarité appartenant à l'intervalle [0.6 – 0.7]

IV.2. Apprentissage MLP

Une fois l'apprentissage du modèle LDA est terminé, on passe à l'apprentissage et au test du réseau de neurones de notre système. Les tests concernent la vérification des performances du réseau de neurones. En effet, cette validation est exprimée par la valeur de perte (loss) et la validation de perte (val_loss) ainsi que le taux d'erreur. Les résultats obtenus sont présentés ci-dessous :

IV.2.1 Performance du modèle proposé avec 3 couches cachées

a) Test de 5 époques d'apprentissage

Nous remarquons d'après les résultats obtenus dans le test de 5époques d'apprentissage que la fonction de perte a commencéà 0.74 et la précision à 0.70 et après 5 époques d'apprentissage la perte est de0.51 et la précision est égale à 0.74. La valeur de MAE = 0.7485

```
Epoch 1/5
5625/5625 [==============================] - ETA: 0s - loss: 0.7436
Epoch 00001: val_loss improved from inf to 0.70935, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h5
5625/5625 [==============================] - 148s 26ms/step - loss: 0.7436 - val_loss: 0.7094
Epoch 2/5
5623/5625 [==============================>.] - ETA: 0s - loss: 0.7011
Epoch 00002: val_loss improved from 0.70935 to 0.69914, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h
5
5625/5625 [==============================] - 150s 27ms/step - loss: 0.7010 - val_loss: 0.6991
Epoch 3/5
5625/5625 [==============================] - ETA: 0s - loss: 0.6455
Epoch 00003: val_loss did not improve from 0.69914
5625/5625 [==============================] - 149s 26ms/step - loss: 0.6455 - val_loss: 0.7271
Epoch 4/5
5625/5625 [==============================] - ETA: 0s - loss: 0.5669
Epoch 00004: val_loss did not improve from 0.69914
5625/5625 [==============================] - 149s 26ms/step - loss: 0.5669 - val_loss: 0.7553
Epoch 5/5
5624/5625 [==============================>.] - ETA: 0s - loss: 0.5173
Epoch 00005: val_loss did not improve from 0.69914
5625/5625 [==============================] - 149s 27ms/step - loss: 0.5173 - val_loss: 0.7465
```

b) Test de 10 époques d'apprentissage

Nous remarquons d'après les résultats obtenus dans le test de 10 époques d'apprentissage que la fonction de perte a commencé à 0.39 et la précision à 0.75 et après 10 époques

d'apprentissage la perte est de 0.36 et la précision est égale à 0.75.La valeur deMAE = 0.7571

```
Epoch 1/10
5624/5625 [============================>.] - ETA: 0s - loss: 0.3939
Epoch 00001: val_loss improved from inf to 0.75416, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h5
5625/5625 [==============================] - 142s 25ms/step - loss: 0.3939 - val_loss: 0.7542
Epoch 2/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3881
Epoch 00002: val_loss did not improve from 0.75416
5625/5625 [==============================] - 143s 25ms/step - loss: 0.3881 - val_loss: 0.7593
Epoch 3/10
5623/5625 [============================>.] - ETA: 0s - loss: 0.3816
Epoch 00003: val_loss did not improve from 0.75416
5625/5625 [==============================] - 145s 26ms/step - loss: 0.3816 - val_loss: 0.7582
Epoch 4/10
5624/5625 [============================>.] - ETA: 0s - loss: 0.3796
Epoch 00004: val_loss did not improve from 0.75416
5625/5625 [==============================] - 146s 26ms/step - loss: 0.3796 - val_loss: 0.7569
Epoch 5/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3747
Epoch 00005: val_loss did not improve from 0.75416
5625/5625 [==============================] - 152s 27ms/step - loss: 0.3747 - val_loss: 0.7561
Epoch 6/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3724- ETA
Epoch 00006: val_loss improved from 0.75416 to 0.75376, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h
5
5625/5625 [==============================] - 157s 28ms/step - loss: 0.3724 - val_loss: 0.7538
Epoch 7/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3679
Epoch 00007: val_loss did not improve from 0.75376
5625/5625 [==============================] - 159s 28ms/step - loss: 0.3679 - val_loss: 0.7586
Epoch 8/10
5623/5625 [============================>.] - ETA: 0s - loss: 0.3665
Epoch 00008: val_loss did not improve from 0.75376
5625/5625 [==============================] - 153s 27ms/step - loss: 0.3665 - val_loss: 0.7606
Epoch 9/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3623
Epoch 00009: val_loss did not improve from 0.75376
5625/5625 [==============================] - 158s 28ms/step - loss: 0.3623 - val_loss: 0.7562
Epoch 10/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3605
Epoch 00010: val_loss did not improve from 0.75376
5625/5625 [==============================] - 150s 27ms/step - loss: 0.3605 - val_loss: 0.7555
```

c) Test de 15 époques d'apprentissage

Nous remarquons d'après les résultats obtenus dans le test de 15 époques d'apprentissage que la fonction de perte a commencéà 0.35 et la précision à 0.761 et après 15 époques d'apprentissage la perte est de 0.33 et la précision est égale à 0.753.La valeur de MAE = 0.7621.

```
Epoch 1/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3588
Epoch 00001: val_loss improved from inf to 0.76131, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h5
5625/5625 [==============================] - 161s 29ms/step - loss: 0.3588 - val_loss: 0.7613
Epoch 2/15
5623/5625 [============================>.] - ETA: 0s - loss: 0.3564
Epoch 00002: val_loss improved from 0.76131 to 0.75653, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h
5
5625/5625 [==============================] - 162s 29ms/step - loss: 0.3564 - val_loss: 0.7565
Epoch 3/15
5625/5625 [==============================] - ETA: 0s - loss: 0.3534
Epoch 00003: val_loss improved from 0.75653 to 0.75402, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h
5
5625/5625 [==============================] - 151s 27ms/step - loss: 0.3534 - val_loss: 0.7540
Epoch 4/15
5623/5625 [============================>.] - ETA: 0s - loss: 0.3536
Epoch 00004: val_loss did not improve from 0.75402
5625/5625 [==============================] - 150s 27ms/step - loss: 0.3536 - val_loss: 0.7571
Epoch 5/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3519
Epoch 00005: val_loss improved from 0.75402 to 0.75394, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h
5
5625/5625 [==============================] - 155s 28ms/step - loss: 0.3519 - val_loss: 0.7539
Epoch 6/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3474
Epoch 00006: val_loss did not improve from 0.75394
5625/5625 [==============================] - 165s 29ms/step - loss: 0.3474 - val_loss: 0.7598
Epoch 7/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3480
Epoch 00007: val_loss did not improve from 0.75394
5625/5625 [==============================] - 165s 29ms/step - loss: 0.3480 - val_loss: 0.7556
Epoch 8/15
5625/5625 [==============================] - ETA: 0s - loss: 0.3453
Epoch 00008: val_loss did not improve from 0.75394
5625/5625 [==============================] - 166s 30ms/step - loss: 0.3453 - val_loss: 0.7564
Epoch 9/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3441
Epoch 00009: val_loss improved from 0.75394 to 0.75393, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h
5
5625/5625 [==============================] - 166s 30ms/step - loss: 0.3441 - val_loss: 0.7539
Epoch 10/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3431
Epoch 00010: val_loss did not improve from 0.75393
5625/5625 [==============================] - 160s 28ms/step - loss: 0.3432 - val_loss: 0.7604
Epoch 11/15
5623/5625 [============================>.] - ETA: 0s - loss: 0.3421
Epoch 00011: val_loss did not improve from 0.75393
5625/5625 [==============================] - 146s 26ms/step - loss: 0.3421 - val_loss: 0.7565
Epoch 12/15
5625/5625 [==============================] - ETA: 0s - loss: 0.3406
Epoch 00012: val_loss did not improve from 0.75393
5625/5625 [==============================] - 144s 26ms/step - loss: 0.3406 - val_loss: 0.7612
Epoch 13/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3394
Epoch 00013: val_loss did not improve from 0.75393
5625/5625 [==============================] - 148s 26ms/step - loss: 0.3394 - val_loss: 0.7586
Epoch 14/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3388
Epoch 00014: val_loss did not improve from 0.75393
5625/5625 [==============================] - 154s 27ms/step - loss: 0.3388 - val_loss: 0.7624
```

d) Test de 20 époques d'apprentissage :

Nous remarquons d'après les résultats obtenus dans le test de 20 époques d'apprentissage que la fonction de perte a commencéà 0.33 et la précision à 0.75 et après 20 époques d'apprentissage la perte est de 0.32 et la précision est égale à 0.76.La valeur de MAE = 0.7644.

```
Epoch 1/20
5625/5625 [==============================] - ETA: 0s - loss: 0.3379
Epoch 00001: val_loss improved from inf to 0.75532, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h5
5625/5625 [==============================] - 159s 28ms/step - loss: 0.3379 - val_loss: 0.7553
Epoch 2/20
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3373
Epoch 00002: val_loss did not improve from 0.75532
5625/5625 [==============================] - 149s 26ms/step - loss: 0.3373 - val_loss: 0.7607
Epoch 3/20
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3364
Epoch 00003: val_loss improved from 0.75532 to 0.75428, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h
5
5625/5625 [==============================] - 150s 27ms/step - loss: 0.3364 - val_loss: 0.7543
Epoch 4/20
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3351
Epoch 00004: val_loss did not improve from 0.75428
5625/5625 [==============================] - 154s 27ms/step - loss: 0.3351 - val_loss: 0.7593
Epoch 5/20
5623/5625 [==============================>.] - ETA: 0s - loss: 0.3349
Epoch 00005: val_loss did not improve from 0.75428
5625/5625 [==============================] - 165s 29ms/step - loss: 0.3349 - val_loss: 0.7584
Epoch 6/20
5625/5625 [==============================] - ETA: 0s - loss: 0.3344
Epoch 00006: val_loss improved from 0.75428 to 0.75366, saving model to C:/Users/PC/Desktop/test\2020-09-07_recommender_movie.h
5
5625/5625 [==============================] - 169s 30ms/step - loss: 0.3344 - val_loss: 0.7537
Epoch 7/20
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3326
Epoch 00007: val_loss did not improve from 0.75366
5625/5625 [==============================] - 156s 28ms/step - loss: 0.3326 - val_loss: 0.7584
Epoch 8/20
5623/5625 [==============================>.] - ETA: 0s - loss: 0.3330
Epoch 00008: val_loss did not improve from 0.75366
5625/5625 [==============================] - 160s 28ms/step - loss: 0.3330 - val_loss: 0.7564
Epoch 9/20
5625/5625 [==============================] - ETA: 0s - loss: 0.3328
Epoch 00009: val_loss did not improve from 0.75366
5625/5625 [==============================] - 157s 28ms/step - loss: 0.3328 - val_loss: 0.7563
Epoch 10/20
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3309
Epoch 00010: val_loss did not improve from 0.75366
5625/5625 [==============================] - 158s 28ms/step - loss: 0.3309 - val_loss: 0.7621
Epoch 11/20
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3303
Epoch 00011: val_loss did not improve from 0.75366
5625/5625 [==============================] - 160s 28ms/step - loss: 0.3303 - val_loss: 0.7631
```

Le tableau III.7 représente les principaux résultats obtenus pendant la phase de test pour différentes époques (5, 10, 15 et 20). Il illustre les résultats obtenus au début et à la fin des époques ainsi que le pourcentage de l'erreur MAE.

	Loss		**Val_loss**		**L'erreur MAE %**
5 époques	0.7436	0.5173	0.7094	0.7465	74.85
10époques	0.3939	0.3605	0.7542	0.7555	75.71
15époques	0.3588	0.3377	0.7613	0.7634	76.21
20époques	0.3379	0.3288	0.7553	0.7664	76.44

Tableau III.7. Principaux résultats avec 3 couches cachées

IV.2.2 Performance du modèle proposé avec 5 couches cachées

a) Test de 5 époques d'apprentissage

Nous remarquons d'après les résultats obtenus dans le test de 5époques d'apprentissage que la fonction de perte a commencéà 0.4061et la précision à 0.7430 et après 5 époques d'apprentissage la perte est de0.3553et la précision est égale à 0.7469. La valeur deMAE = 0.7482

```
Epoch 1/5
5625/5625 [==============================] - ETA: 0s - loss: 0.4061
Epoch 00001: val_loss improved from inf to 0.74303, saving model to C:/Users/PC/Desktop/test\2020-09-21_recommender_movie.h5
5625/5625 [==============================] - 284s 50ms/step - loss: 0.4061 - val_loss: 0.7430
Epoch 2/5
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3876
Epoch 00002: val_loss did not improve from 0.74303
5625/5625 [==============================] - 278s 49ms/step - loss: 0.3876 - val_loss: 0.7437
Epoch 3/5
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3749
Epoch 00003: val_loss improved from 0.74303 to 0.74290, saving model to C:/Users/PC/Desktop/test\2020-09-21_recommender_movie.h
5
5625/5625 [==============================] - 280s 50ms/step - loss: 0.3749 - val_loss: 0.7429
Epoch 4/5
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3629
Epoch 00004: val_loss improved from 0.74290 to 0.74281, saving model to C:/Users/PC/Desktop/test\2020-09-21_recommender_movie.h
5
5625/5625 [==============================] - 288s 51ms/step - loss: 0.3629 - val_loss: 0.7428
Epoch 5/5
5625/5625 [==============================] - ETA: 0s - loss: 0.3553
Epoch 00005: val_loss did not improve from 0.74281
5625/5625 [==============================] - 286s 51ms/step - loss: 0.3553 - val_loss: 0.7469
```

b) Test de 10 époques d'apprentissage

Nous remarquons d'après les résultats obtenus dans le test de 10 époques d'apprentissage que la fonction de perte a commencéà 0.3471 et la précision à 0.7391 et après 10 époques d'apprentissage la perte est de 0.3130et la précision est égale à 0.7442. La valeur de MAE = 0.7445.

```
Epoch 1/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3471
Epoch 00001: val_loss improved from inf to 0.73906, saving model to C:/Users/PC/Desktop/test\2020-09-21_recommender_movie.h5
5625/5625 [==============================] - 276s 49ms/step - loss: 0.3471 - val_loss: 0.7391
Epoch 2/10
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3395
Epoch 00002: val_loss did not improve from 0.73906
5625/5625 [==============================] - 275s 49ms/step - loss: 0.3395 - val_loss: 0.7413
Epoch 3/10
5624/5625 [==============================>.] - ETA: 0s - loss: 0.3346
Epoch 00003: val_loss did not improve from 0.73906
5625/5625 [==============================] - 290s 52ms/step - loss: 0.3346 - val_loss: 0.7417
Epoch 4/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3307
Epoch 00004: val_loss improved from 0.73906 to 0.73870, saving model to C:/Users/PC/Desktop/test\2020-09-21_recommender_movie.h
5
5625/5625 [==============================] - 289s 51ms/step - loss: 0.3307 - val_loss: 0.7387
Epoch 5/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3261
Epoch 00005: val_loss did not improve from 0.73870
5625/5625 [==============================] - 290s 52ms/step - loss: 0.3261 - val_loss: 0.7408
Epoch 6/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3225
Epoch 00006: val_loss did not improve from 0.73870
5625/5625 [==============================] - 293s 52ms/step - loss: 0.3225 - val_loss: 0.7436
Epoch 7/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3184
Epoch 00007: val_loss did not improve from 0.73870
5625/5625 [==============================] - 291s 52ms/step - loss: 0.3184 - val_loss: 0.7439
Epoch 8/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3144
Epoch 00008: val_loss did not improve from 0.73870
5625/5625 [==============================] - 291s 52ms/step - loss: 0.3144 - val_loss: 0.7405
Epoch 9/10
5625/5625 [==============================] - ETA: 0s - loss: 0.3127
Epoch 00009: val_loss did not improve from 0.73870
5625/5625 [==============================] - 306s 54ms/step - loss: 0.3127 - val_loss: 0.7418
```

c) Test de 15 époques d'apprentissage :

Nous remarquons d'après les résultats obtenus dans le test de 15 époques d'apprentissage que la fonction de perte a commencéà 0.3101et la précision à 0.7420 et après 15 époques d'apprentissage la perte est de0.2901et la précision est égale à 0.7445. la valeur de MAE = 0.7458.

```
Epoch 1/15
5625/5625 [==============================] - ETA: 0s - loss: 0.3101
Epoch 00001: val_loss improved from inf to 0.74197, saving model to C:/Users/PC/Desktop/test\2020-09-21_recommender_movie.h5
5625/5625 [==============================] - 283s 50ms/step - loss: 0.3101 - val_loss: 0.7420
Epoch 2/15
5625/5625 [==============================] - ETA: 0s - loss: 0.3073
Epoch 00002: val_loss did not improve from 0.74197
5625/5625 [==============================] - 284s 50ms/step - loss: 0.3073 - val_loss: 0.7436
Epoch 3/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.3044
Epoch 00003: val_loss did not improve from 0.74197
5625/5625 [==============================] - 279s 50ms/step - loss: 0.3045 - val_loss: 0.7434
Epoch 4/15
5625/5625 [==============================] - ETA: 0s - loss: 0.3020
Epoch 00004: val_loss improved from 0.74197 to 0.74056, saving model to C:/Users/PC/Desktop/test\2020-09-21_recommender_movie.h
5
5625/5625 [==============================] - 279s 50ms/step - loss: 0.3020 - val_loss: 0.7406
Epoch 5/15
5625/5625 [==============================] - ETA: 0s - loss: 0.3001
Epoch 00005: val_loss did not improve from 0.74056
5625/5625 [==============================] - 276s 49ms/step - loss: 0.3001 - val_loss: 0.7456
Epoch 6/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.2979
Epoch 00006: val_loss did not improve from 0.74056
5625/5625 [==============================] - 274s 49ms/step - loss: 0.2979 - val_loss: 0.7407
Epoch 7/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.2961
Epoch 00007: val_loss did not improve from 0.74056
5625/5625 [==============================] - 274s 49ms/step - loss: 0.2961 - val_loss: 0.7441
Epoch 8/15
5624/5625 [============================>.] - ETA: 0s - loss: 0.2944
Epoch 00008: val_loss did not improve from 0.74056
5625/5625 [==============================] - 343s 61ms/step - loss: 0.2944 - val_loss: 0.7434
Epoch 9/15
5625/5625 [==============================] - ETA: 0s - loss: 0.2923
Epoch 00009: val_loss did not improve from 0.74056
5625/5625 [==============================] - 279s 50ms/step - loss: 0.2923 - val_loss: 0.7435
```

d) Test de 20 époques d'apprentissage :

Nous remarquons d'après les résultats obtenus dans le test de 20 époques d'apprentissage que la fonction de perte a commencéà 0.2902et la précision à 0.7427 et après 20 époques d'apprentissage la perte est de0.2822et la précision est égale à 0.7470. La valeur deMAE = 0.7498.

```
Epoch 1/20
5624/5625 [============================>.] - ETA: 0s - loss: 0.2902
Epoch 00001: val_loss improved from inf to 0.74266, saving model to C:/Users/PC/Desktop/test\2020-09-21_recommender_movie.h5
5625/5625 [==============================] - 281s 50ms/step - loss: 0.2902 - val_loss: 0.7427
Epoch 2/20
5624/5625 [============================>.] - ETA: 0s - loss: 0.2898
Epoch 00002: val_loss did not improve from 0.74266
5625/5625 [==============================] - 283s 50ms/step - loss: 0.2898 - val_loss: 0.7441
Epoch 3/20
5625/5625 [==============================] - ETA: 0s - loss: 0.2888
Epoch 00003: val_loss did not improve from 0.74266
5625/5625 [==============================] - 277s 49ms/step - loss: 0.2888 - val_loss: 0.7430
Epoch 4/20
5624/5625 [============================>.] - ETA: 0s - loss: 0.2868
Epoch 00004: val_loss did not improve from 0.74266
5625/5625 [==============================] - 284s 50ms/step - loss: 0.2868 - val_loss: 0.7435
Epoch 5/20
5625/5625 [==============================] - ETA: 0s - loss: 0.2858
Epoch 00005: val_loss did not improve from 0.74266
5625/5625 [==============================] - 299s 53ms/step - loss: 0.2858 - val_loss: 0.7427
Epoch 6/20
5625/5625 [==============================] - ETA: 0s - loss: 0.2844
Epoch 00006: val_loss did not improve from 0.74266
5625/5625 [==============================] - 303s 54ms/step - loss: 0.2844 - val_loss: 0.7479
```


Le tableau III.8représente les principaux résultats obtenus pendant la phase de test pour différentes époques (5, 10, 15 et 20). Il illustre les résultats obtenus au début et à la fin des époques ainsi que le pourcentage de l'erreur MAE.

	Loss		**Val_loss**		**L'erreur MAE %**
5 époques	0.4061	0.3553	0.7430	0.7469	74.82
10époques	0.3471	0.3127	0.7391	0.7442	74.45
15époques	0.3101	0.2901	0.7420	0.7445	74.58
20époques	0.2902	0.2822	0.7427	0.7470	74.98

Tableau III.8. Principaux résultats avec 5 couches cachées

Discussions des résultats

Selon les différents résultats obtenus, on remarque que le taux d'erreur baisse tandis que la précision augmente cela signifie que le modèle proposé est capable de mieux s'adapter à nos données en augmentant le nombre de couches et d'époques. Notre système répond d'ailleurs à la définition du réseau de neurones qui disait que plus le réseau de neurones est profond meilleur sont ses performances. On peut donc constater que le nombre d'époques et la profondeur de réseaux sont des facteurs importants pour l'obtention de meilleurs résultats.

V. CONCLUSION

Ce chapitre décrit l'implémentation d'un algorithme de filtrage collaboratif hybride qui capable de prendre en considération des informations contextuelles non observables. L'environnement de développement, la base de données utilisée et les détails d'implémentation sont alors présentés. Des tests sur la base de données *MovieLens* sont réalisés dans le but d'évaluer les performances du système mis en œuvre. En effet, les résultats obtenus confirment que l'algorithme proposé améliore la qualité des prédictions.

Conclusion et Perspectives

Dans ce mémoire, nous avons proposé un système de recommandation hybride combinant la méthode LDA avec un réseau de neurones profond pour la recommandation des films. LDA est utilisée pour trouver la structure sémantique latente, la distribution des mots sur les thèmes latents et le mélange des distributions des thèmes latents, à partir des descriptions textuelles des objets consultés. Le résultat provenant de LDA est ensuite intégré dans un réseau de neurones profond afin de suggérer des items plus pertinents aux usagers. Nous avons montré, à travers les différents tests expérimentaux effectués, que l'intégration du contexte implicite via un réseau neuronal profond améliore les performances des systèmes de recommandation.

Perspectives

Comme perspectives de recherches futures, nous envisageons de:

1. Tester le modèle proposé avec d'autres bases de données telles que Book-Crossing
2. Augmenter le nombre de couches et de neurones cachés
3. Utiliser d'autres métriques d'évaluation
4. Considérer d'autres informations contextuelles

Références

A. Références Bibliographiques

[1] Corentin HARDY, Contribution au développement de l'apprentissage profonddans les systems distribués », Thèse de doctorat, Universite de Rennes 1, 2019

[2]Unger, M., Shapira, B., Rokach, L., &Livne, A. Inferring contextual preferences using deep encoder-decoder learners.*New Review of Hypermedia and Multimedia*, *24*(3), 262–290.2018.

[3] Bazire, M., Brézillon, P. Understanding Context Before Using It. Vol. 3554, 29–40. 2005.

[4] Zheng, Y. Interpreting Contextual Effects By Contextual Modeling In Recommender Systems.ArXiv.2017.

[5] Burke R., "Hybrid Recommender Systems: Survey and Experiments", User modeling and user-adapted interaction, Vol. 12, N°. 4, pp. 331 - 370, 2002.

[6] Basu C., Hirsh H., Cohen W., Manning N. C., "Recommendation: A study in combining multiple information sources", Technical paper, 2001.

[7] Pazzani M. J., "A framework for collaborative, content-based and demographic filtering", Artificial Intelligence Review, Vol.13, N°. 5, pp. 393 - 408, 1999.

[8] Nguyen A.T., Denos N., Berrut C., "Modèle d'espaces de communautés basé sur la théorie des ensembles d'approximation dans un système de filtrage hybride", Conférence en Recherche Information et Applications, pp. 303 - 314, 2006.

[9] Candillier L., Jack K., Fessant F., Meyer F., "State-of-the-art recommender systems", Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling, 22 pages, 2009.

[10] Schilit B. N., Adams N. I., Want R., "Context-Aware Computing Applications", In: Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE Computer Society, Santa Cruz, CA, pp. 85 - 90, 1994.

[11] Brown P. J., "Triggering information by context", personnal Technologies, Vol. 2, N°.1, pp.19 - 29, 1998.

[12] Schein A. I., Popescul A., Ungar L. H., Pennock D. M., "Generative Models for Cold-Start Recommendations", In: Proceedings of the 2001 SIGIR Workshop on Recommender Systems, 2001.

[13] O'Donovan J., Smyth B., "Trust in recommender systems", In: Proceedings of the 10th international ACM conference on Intelligent user interfaces, New York, USA, pp. 167 - 174, 2005.

[14] Gasmi I., Seridi-Bouchlaghem H., Labar H., "Collaborative filtering recommendation based on research habit of users", In: Proceedings of the third IEEE International Conference on Multimedia Computing and Systems, Tangier, Morocco, 2012.

[15] Polat H., Du W., "SVD-based collaborative filtering with privacy", In: Proceedings of the ACM Symposium on Applied Computing, 2004.

[16] Jamali M., Ester M., "Using a trust network to improve top-n recommendation", In: Proceedings of the third ACM conference on Recommender systems (RecSys '09), New York, USA, pp. 181 - 188, 2009.

[17] Kriesel D., "A Brief Introduction to Neural Networks", Publisher: dkriesel.com, 244 pages, 2011.

[18] Boughaba M et Boukhris B "L`apprentissage profond (Deep Learning) pourla classification et la recherche d'images par le contenu" , p 18 , 23, 2017

[19] Edwin Simonnet, Réseaux de neurones profonds appliqués à la compréhension de la parole, Thèse de doctorat, Université du Maine, 2019

[20] Merzougui D et Kermi M "Modélisation du problème de Partitionnement Materiel/Logiciel à l'aide du MLP" universite badji Moukhtar Annaba . 2009

[21] Sebastian Ruder An overview of gradient descent optimisation algorithms. arXiv preprint arXiv:1609.04747. (2016).

[22] Lefebvre-Brossard Antoine "Sur l'utilisation de réseaux de neurones dans un système de recommandations réciproques", mémoire Université de Montréal École Polytechnique De Montréal (2018).

[23] Chang C. C., Chen P. L., Chiu F. R., Chen Y. K.,"Application of neural networks and Kano's method to content recommendation in web personalization", Expert Systems With Applications, Vol. 36, N°. 3, pp. 5310 - 5316,2009.

[24] Chou P. H., Li P. H., Chen K. K.,Wu M. J.,"Integrating web mining and neural network for personalized e-commerce automatic service", Expert Systems With Applications, Vol. 37, N°. 4, pp. 2898 - 2910, 2010.

[25] Biancalana C., Gasparetti F., Micarelli A., Miola A., SansonettiG.,"Context-aware movie recommendation based on signal processing and machine learning", In: Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation, pp. 5 -10, 2011.

[26] KavithaDeviM. K., ThirumalaiSamyR., VinothKumarS., VenkateshP., "Probabilistic neural network approach to alleviate sparsity and cold start problems in collaborative Recommender Systems", In: Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, pp. 245-248, 2010.

[27] SKrstic M., BjelicaM., "Context-aware personalized program guide based on neural network", IEEE Transactions on Consumer Electronics, Vol. 58, N°.4, pp. 1301 - 1306, 2012.

[28] S.Zhang,L.Yao,et A.Sun,"Deeplearningbasedrecommendersystem:Asurveyandnew perspectives", arXivpreprintarXiv :1707.07435, 2017.

[29] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, et T.-S. Chua, "Neural collaborative filtering", dans Proceedings of the 26th International Conferenceon World Wide Web. International World Wide Web ConferencesSteeringCommittee, , pp. 173–182. 2017

[30] X. Wang, X. He, L. Nie, et T.-S. Chua, "Item silk road : Recommendingitfrom information domains to social users", dans Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2017, pp. 185– 194.

[31] J. Lian, F. Zhang, X. Xie, et G. Sun, "Cccfnet : a content-boosted collaborative filtering neural network for cross domain recommender systems", dans Proceedings of the 26th International Conferenceon World Wide Web Companion. International World Wide Web Conferences Steering Committee, , pp. 817–818. 2017

[32] D. Liang, J. Altosaar, L. Charlin, et D. M. Blei, "Factorizationmeets the item embedding : Regularizing matrix factorizationwith item co-occurrence", dans Proceedings of the 10th ACM conference on recommendersystems. ACM, , pp. 59–66. 2016

[33] S. Sedhain, A. K. Menon, S. Sanner, et L. Xie, "Autorec : Autoencoders meet collaborative filtering", dans Proceedings of the 24th International Conferenceon World Wide Web. ACM, , pp. 111–112. 2015

[34] F. Strub, R. Gaudel, et J. Mary, "Hybridrecommender system based on autoencoders", dans Proceedings of the 1st Workshop on Deep Learning for RecommenderSystems. ACM, , pp. 11–16. 2016

[35] T. Alashkar, S. Jiang, S. Wang, et Y. Fu, "Examples-rules guided deep neural network for make up recommendation." dans AAAI, , pp. 941–947. 2017

[36] H. Guo, R. Tang, Y. Ye, Z. Li, et X. He, "Deepfm : afactorization-machine based neural network for ctrprediction", arXivpreprintarXiv:1703.04247, 2017.

[37] H. Ying, L. Chen, Y. Xiong, et J. Wu, "Collaborative deepranking : A hybrid pair-wise recommendation algorithm with implicit feedback",dans Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, , pp. 555–567. 2016

[38] C. Chen, P. Zhao, L. Li, J. Zhou, X. Li, et M. Qiu, "Locally connected deeplearning frame work for industrial-scale recommender systems", dans Proceedings of the 26th International Conferenceon World Wide Web Companion. International World Wide Web Conferences Steering Committee, , pp. 769–770. 2017

[39] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, et T.-S. Chua, "Attentive collaborative filtering : Multimedia recommendation with item-and component-level attention",dans Proceedings of the 40th International ACM SIGI R conference on Research and Development in Information Retrieval. ACM, , pp. 335–344. 2017

[40] L.Zheng,V.Noroozi,etP.S.Yu,"Joint deep modeling of users and items using reviews for recommendation", dans Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. ACM, , pp. 425–434. 2017

[41] Y.Gong et Q.Zhang,"Hashtagrecommendationusing attention-basedconvolutional neural network." dans IJCAI, , pp. 2782–2788. 2016

[42] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, et H. Jing, "Recurrentrecommender networks", dans Proceedings of the tenth ACM international conference on web search and data mining. ACM, , pp. 495–503. 2017

[43] B. Hidasi, M. Quadrana, A. Karatzoglou, et D. Tikk, "Parallelrecurrent neural etwork architectures for feature-rich session-basedrecommendations", dans Proceedings of the 10th ACM Conference on RecommenderSystems. ACM, , pp. 241–248. 2016

[44] B. Hidasi, A. Karatzoglou, L. Baltrunas, et D. Tikk, "Session-basedrecommendationswithrecurrent neural networks", arXivpreprintarXiv:1511.06939, 2015.

[45] E.Smirnova et F.Vasile ,"Contextual sequence modeling for recommendation with recurrent neural net works",dans Proceedings of the 2nd Workshop on Deep Learning for RecommenderSystems. ACM, , pp. 2–9. 2017

[46] Y. Liu, S. Wang, M. S. Khan, et J. He, "A noveldeephybridrecommender system based on auto-encoder with neural collaborative filtering", Big Data Mining and Analytics, vol. 1, no. 3, pp. 211–221, 2018.

[47] Y.Li,T.Liu,J.Jiang,etL.Zhang,"Hashtagrecommendationwithtopicalattention-based lstm". Coling, 2016.

[48] A.Beutel ,P.Covington,S.Jain,C.Xu,J.Li,V.Gatto,etE.H.Chi,"Latentcross:Making use of context in recurrentrecommendersystems", dans Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. ACM, , pp. 46–54. 2018

[49] Delporte J., Karatzoglou A., Matuszczyk T., Canu S., "Socially Enabled Preference Learning from Implicit Feedback Data", In: Proceedings of the 13th European Conference on Machine Learning and Principles and Practive of Knowledge Discovery in Databases, pp. 145 - 160, Springer, 2013.

[50] Amatriain X., Pujol J. M., Oliver N., "I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems", In: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP '09), Berlin, Heidelberg, pp. 247 - 258, 2009.

[51] TAOULI Sarah et BENACHENHOU Walid , Utilisation de factorisation matricielle dans les Systèmes de recommandation sensible au contexte Université Abou BakrBelkaid– Tlemcen p 50. 2017

[52] Josiane mothe , Ambinintsoa Jocelyn Rakotonirina , Filtrage collaboratif sensible au contexte – une approche basée sur LDA ,(INFORSID 2017) Toulouse France . 2017.

[53] Document Embedding Models - A Comparisonwith Bag-of-Words Master university of zurich P 6. 7 sep 2018)
[54] Blei, D. M., Ng, A. Y., Jordan, M. I. Latent dirichlet allocation. Journal of machine Learning research, 3(Jan) :993-1022. 2003.

[55] Kingma, D. P. and Ba, J. Adam : A method for stochastic optimization. CoRR, abs/1412.6980. 2014.

[56] Nitish Srivastava et al. « Dropout : A Simple Way to Prevent Neural Networks fromOverfitting ». Dans : Journal of Machine Learning Research 15 .p. 1929-1958. 2014.

[57] François Chollet:Deep learning with python Manning publications shilter island 2018.

[58] Piton T., "Une Méthodologie de Recommandations Produits Fondée sur l'Actionnabilité et l'Intérêt Économique des Clients Application à la Gestion de la Relation Client du groupe VM Matériaux", Thèse de doctorat, Université de Nantes, France, 2011.

B. Références Web (Techniques)

[W1]https://docplayer.fr/1209466-Etat-de-l-art-sur-les-systemes-de-recommandation.html

[W2]
https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_francais.png?uselang=fr

[W3] https://github.com/wikistat/Apprentissage

[W4] https://en.wikipedia.org/wiki/Autoencoder

[W5] http://www.nltk.org/install.html

[W6] https://pypi.python.org/pypi/stop-words

[W7] https://radimrehurek.com/gensim/install.html

[W8] https://radimrehurek.com/gensim/models/ldamodel.html

Résumé

Résumé

Les systèmes de recommandations proposent aux utilisateurs des objets en relation *avec leurs centres d'intérêt.Les intérêts des usagers dépendent étroitement du contexte dans lequel ils se trouvent.*Ainsi, avec l'apparition du concept de l'apprentissage en profondeur (Deep Learning) et les bases de données volumineuses, un nouvel axe de recherche est développé. Notre projet consiste donc à proposer un système de recommandation sensible au contexte, qui est basé sur le deeplearning et plus particulièrement les réseaux de neurones multicouches. Le modèle proposé considère le contexte à partir d'informations contextuelles non observables. Nous avons choisi de capturer l'intention implicite de l'utilisateur par les thèmes associés aux items qu'il a consultés en utilisant la méthode LDA (Latent Dirichlet Allocation). Les résultats expérimentaux obtenus sur le jeu de données de *MovieLens100k* sont très encourageants. Ils ont montré l'efficacité de la méthode proposée.

Mots clés

Apprentissage en profondeur, Filtrage Collaboratif,LDA,Réseaux de neurones, Systèmes de Recommandation .

Abstract

Abstract

Recommander Systems provide to users the items which are related to their preferences. The user's interests highly depend on the context. Thus, with the emergence of Deep Learning and the large data sets, a new search area is emerging. Our purposeis to propose acontext-aware recommender system, which is based on Deep Learning, specially Multilayer Neural Network. We have chosen to capture the user's implicit context through the topics of items using the LDA (Latent Dirichlet Allocation) method. Experimental results from Movielens 100K dataset data set are very promising; they have shown the effectiveness of the proposed method.

Keywords

Collaborative filetring ,Deep learning ,LDA, Neural networks, Recommandation systems

ملخـــص

يوفر نظام التوصيات للمستخدمين أشياء تتعلق بمجالات اهتمامهم. يعتمد هذا النوع من الأنظمة بشكل كبير على اهتمامات المستخدمين. سمح ظهور مفهوم "التعلم العميق" وقواعد البيانات الضخمة بتطوير هذا المجال البحثي الجديد. يعتمد مشروعنا أساسا على اقتراح نظام توصية حساس للسياق. زيادة إلى مفهوم التعلم العميق، يرتكز هذا النظام أيضًا على الشبكات العصبية متعددة الطبقات. يُستخرج السياق المُعالج في هذا النموذج من المعلومات السياقية غير المعلنة. اخترنا استخراج النوايا الضمنية للمستخدم من خلال الموضوعات المرتبطة بالعناصر التي اطلع عليها و ذلك باستخدام طريقة LDA (Allocation Latent Dirichlet). النتائج التجريبية التي تم الحصول عليها من خلال مجموعة بيانات MovieLens100K مشجعة للغاية ؛فقد أظهرت فعالية الطريقة المقترحة.

الكلمات المفتاحية

أنظمة التوصية ،تصفية تعاونية ، التعلم العميق، LDA ، الشبكات العصبية