République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou

Faculté de : Génie électrique et d'informatique
Département : Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de

Master en Informatique

Spécialité : Conduite de projets informatiques

Présenté par :

OTMANE-CHERIF Dhya et **OULD-SAID Cynthia**

Thème : Mise en œuvre des réseaux de neurones pour une recommandation basée sur du contenu

Proposé et dirigé par : Mr. SADI Samy

Soutenu le 17 juillet 2019 devant le jury composé de:

Mme. AMIROUCHE Fatiha	Présidente du Jury
Mme. BELKACEMI Lila	Membre du Jury
Mr. SADI Samy	Directeur de mémoire

Dédicaces

Je dédie ce modeste travail :

À mes très chers parents

Qui m'ont inculqué le sens de la responsabilité, du travail et de l'optimisme. Je ne saurais exprimer la gratitude, le respect et la reconnaissance que j'ai pour eux.

À ma très chère sœur

Pour son soutien moral et ses précieux conseils tout au long de mes études. « Loin des yeux mais près du cœur ».

À ma chère grand-mère

À qui je souhaite une bonne santé et une longue vie.

À mes amies

Pour leurs aides et supports dans les moments difficiles.

Cynthia

Dédicaces

Je dédie ce modeste travail :

À mes très chers parents

Qui n'ont jamais cessé de me soutenir et de m'épauler tout au long de mes études afin que je puisse atteindre mes objectifs.

Que ce travail traduit ma gratitude et mon plus profond respect.

À mon très cher frère

Pour son soutien infini et ses encouragements incessants, qui m'ont toujours donné de la force pour persévérer.

À mes chers grands-parents

À qui je souhaite une bonne santé et une longue vie.

À mes amies

Pour leurs aides et supports dans les moments difficiles.

Dhya

Remerciements

Nous tenons à exprimer d’abord toute notre gratitude à notre encadreur Dr. **Sadi Samy** pour ses conseils, ses directives, ses encouragements, son aide et son soutien sans faille.

Nos remerciements s’adressent aux membres du jury qui ont accepté d’évaluer notre travail.

Nous tenons à exprimer nos sincères remerciements à tout le corps professoral et administratif de l’université Mouloud Mammeri de Tizi-Ouzou.

Enfin, nous remercions tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Résumé

Avec l'avènement du web et les évolutions technologiques, entre autres, la masse de données à exploiter ou analyser est devenue très volumineuse. Il est devenu donc difficile de savoir quelles sont les données à rechercher et où les trouver. Des techniques informatiques ont été développées pour faciliter cette recherche ainsi que l'extraction des informations pertinentes. Celle sur laquelle nous nous concentrons dans ce mémoire est la recommandation. Il s'agit de guider l'utilisateur dans son exploration des données afin qu'il trouve des informations pertinentes. Bien que les systèmes de recommandation ont été étudiés dans des domaines divers et variés comme le web, le e-commerce et bien d'autres. En effet, plusieurs techniques de recommandation ont vu le jour. Nous nous intéressants dans notre travail aux techniques à base de l'apprentissage automatique, plus précisément celles basées sur les réseaux de neurones. Pour recommander des films à un utilisateur, nous proposons trois modèles de réseaux de neurones, qui prennent en entrée du contenu textuel représentant des descriptions de films extraites de la collection préexistante de l'Internet Movie Database (IMDb) et qui les classent selon leurs attributs descriptifs. Nous avons implémenté et évalué ces modèles. Par la suite, nous avons effectué une comparaison des résultats obtenus. Nous avons également réalisé une interface afin de visualiser la recommandation.

Mots clés: Systèmes de recommandation, apprentissage automatique, réseaux de neurones, Tensorflow, Keras, l'Internet Movie Database (IMDb).

Abstract

With the advent of the web and technological developments, among others, the mass of data to be exploited or analyzed has become very large. As a result, it has become difficult to know what data to look for and where to find them. Computer techniques have been developed to facilitate this research as well as the extraction of relevant information. The one we focus on in this memoir is the recommendation. This is to guide the user in their exploration of the data so that it finds relevant information. Although recommendation systems have been studied in various and varied fields such as the web, e-commerce and many others. Indeed, several recommendation techniques have emerged. We are interesting in our work on techniques based on machine learning, more precisely those based on neural networks. To recommend films to a user, we propose three models of neural networks, which take as input textual content representing movie descriptions extracted from the Internet Movie Database's (IMDb) pre-existing collection and classifying them according to their attributes descriptive. We have implemented and evaluated these models. Subsequently, we performed a comparison of the results obtained. We also made an interface to visualize the recommendation.

Key-words: Recommendation systems, machine learning, neural networks, Tensorflow, Keras, the Internet Movie Database (IMDb).

Table des matières

Table des tableaux et des figures

Introduction générale

1 Contexte du travail

La quantité d'information à disposition de chacun est en constante croissance, en majeure partie grâce à internet. Cette quantité, presque infinie dans certains domaines, a créé plusieurs nouveaux besoins notamment la nécessité de nouvelles techniques pour en faire le tri. Plutôt que de devoir filtrer manuellement ces quantités importantes d'offres changeant au fil du temps, des systèmes de recommandation ont été créés pour en présenter un petit nombre à l'utilisateur.

Un système de recommandation est un outil logiciel qui vise à prédire l'intérêt qu'un utilisateur pourrait avoir pour tel ou tel article et cela à partir d'un certain nombre d'informations. Afin d'y arriver, plusieurs méthodes ont été utilisées notamment l'apprentissage automatique qui comprend différents algorithmes. Parmi eux, les algorithmes à base de réseaux neuronaux qui ont permis de créer des représentations plus complexes et complètes en profitant d'importantes quantités de données. L'apparition récente de systèmes de recommandations à partir de ces réseaux neuronaux a montré que ceux-ci peuvent obtenir des résultats intéressants et ainsi améliorer la qualité des recommandations offertes aux utilisateurs.

2 Problématique

Plusieurs problématiques peuvent être mises en évidence dans le cadre de notre travail.

Tout d'abord, il faut déterminer une approche pour les systèmes de recommandation permettant d'entraîner des modèles utilisant des données textuelles dépendant uniquement de l'utilisateur pour faire des recommandations aussi complexes et diversifiées que celles faites par les filtres collaboratifs.

Par la suite, il faudra bien définir l'implémentation de notre système de recommandation en utilisant les réseaux de neurones, qui contrairement aux modèles de recommandation traditionnels, permettent de manipuler facilement des données massives et de mieux comprendre les demandes des utilisateurs et les caractéristiques des éléments, et ainsi fournir une meilleure qualité de recommandation.

3 Contribution

Notre contribution dans le cadre de la recommandation de films se situe à plusieurs niveaux :

1. Réalisation d'un état de l'art sur les systèmes de recommandation ainsi que l'apprentissage automatique.
2. Construction de trois modèles pour la recommandation de films.
3. Entrainement et évaluation de ces modèles après les avoir implémentés et réalisation d'une interface.

4 Organisation du mémoire

Ce mémoire est articulé en deux parties englobant quatre chapitres en plus de l'introduction générale et de la conclusion générale.

La première partie comporte le premier et deuxième chapitre, où nous présentons la notion de système de recommandation, ses objectifs ainsi que ses différentes classifications dans le chapitre 1. Par la suite, nous présentons l'apprentissage automatique tout en détaillant la notion de réseau de neurones et son application dans les systèmes de recommandation dans le chapitre 2.

La deuxième partie compte le troisième et quatrième chapitre. Nous décrivons les modèles proposés dans le troisième chapitre. Puis, dans le quatrième chapitre, nous décrivons l'environnement, les outils utilisés, les résultats obtenus ainsi que l'interface réalisée dans le cadre de la recommandation de films.

Chapitre 1: Les systèmes de recommandation

1 Introduction

Avec la très grande masse d'informations, aujourd'hui disponible sur l'Internet il est devenu primordial de concevoir des mécanismes qui permettent aux utilisateurs d'accéder à ce qui les intéresse le plus rapidement possible. En effet, leurs besoins sont difficiles à traiter d'une part parce qu'ils ne sont pas formulés explicitement et d'autre part parce qu'ils sont évolutifs. Les systèmes de recommandation apportent ainsi une solution à ce problème. Ce sont des outils puissants conçus dans le but de filtrer et d'adapter les informations pour chaque utilisateur. Ils deviennent ainsi indispensables dans différentes variétés d'industries, d'entreprises, de services financiers, dans le secteur de la musique et de la radio en ligne, de la télévision et des vidéos, des publications en ligne et dans d'innombrables autres domaines.

Dans ce chapitre, nous commençons par donner un historique retraçant les différents travaux et progrès réalisés dans le domaine des systèmes de recommandation. Après quoi, nous donnons quelques définitions d'un système de recommandation, ses différentes classifications et ses avantages et inconvénients. Ensuite, nous présentons les techniques adoptées par les systèmes de recommandation. Enfin, nous terminons le chapitre par une conclusion.

2 Historique

Le monde de l'informatique a vu naitre la recommandation en 1979 grâce à un système bibliothécaire primitif appelé Grundy [1]. Considéré comme la première étape vers les systèmes de recommandation, ce dernier effectue des suggestions de livres en se basant sur une classification en stéréotypes des utilisateurs.

Par la suite, « *Information Lens System* » [2] qui est apparu en 1987 a été considéré comme l'un des premiers vrais systèmes de recommandation. Depuis, d'autres systèmes sont apparus tels que Tapestry [3] en 1992 qui effectue des recommandations de documents, GroupLens [4] en 1994 qui recommande des articles d'actualités et des films et enfin en 1995, Ringo [5] et Bellcore [6] qui sont respectivement des systèmes de recommandation de musiques et de vidéos.

Depuis les années 2000 et grâce à l'évolution de l'apprentissage automatique, les systèmes de recommandation utilisent des algorithmes d'apprentissage artificiel notamment

dans des sites de ventes et de divertissement tels que Netflix et Amazon mais aussi dans des réseaux sociaux tels que Facebook et Instagram.

Aujourd'hui, les systèmes de recommandation sont devenus très populaires et incontournables sur internet, notamment pour des sites de e-commerce (vente en ligne).

3 Définition des systèmes de recommandation

Il existe plusieurs définitions qui ont été données pour les systèmes de recommandation dans l'état de l'art. Nous donnons quelques-unes dans cette section.

Saimadhu [7] définit un système de recommandation comme étant une boîte noire qui analyse un certain ensemble d'utilisateurs et présentes les *items* qu'un utilisateur peut aimer. Le terme « *item* » désigne des éléments qui peuvent être de formes très différentes : documents textuels, images, vidéos, lieux, produits commerciaux etc.

Burke [8] définit les systèmes de recommandation comme étant des systèmes capables de fournir des recommandations personnalisées permettant de guider l'utilisateur vers des ressources intéressantes et utiles au sein d'un espace de données important.

En définition plus simple, un système de recommandation est une forme spécifique de filtrage de l'information qui se base sur les préférences et le comportement de l'utilisateur afin de lui présenter des *items* susceptibles de lui plaire.
Ce dernier effectue :

- Des suggestions sous forme de listes personnalisées et classées des *items* qui peuvent être des films, livres ou musiques à des utilisateurs.
- Des renseignements pour aider ces utilisateurs à savoir quels *items* leurs convient.

4 Classification des systèmes de recommandation

Durant ces dernières années, plusieurs classifications sont apparues. Celles-ci se basent sur différents facteurs pour catégoriser un système de recommandation notamment : la connaissance des goûts de l'utilisateur, son positionnement par rapport aux autres et la connaissance des *items* à recommander.

4.1 Classification classique

Cette classification possède trois types d'approches :

- Approches basées sur le contenu ;
- Approches basées sur le filtrage collaboratif ;
- Approche hybride.

4.1.1 Les approches basées sur le contenu

4.1.1.1 Définition

Pazzani [9] définit les approches basées sur le contenu comme étant des méthodes qui émettent des recommandations en analysant la description des articles, objets ou *items* qui ont été évalués par l'utilisateur et la description de ceux qui sont destinés à être recommandés.

Cette approche permet d'associer des *items* à un profil utilisateur. Notamment, en utilisant des techniques d'indexation qui consistent à organiser un ensemble de documents et à faciliter ultérieurement la recherche de contenu dans cette collection. Un index est, en toute généralité, une liste de descripteurs auxquels est associée une liste des documents ou parties de documents.

La figure 1 résume le principe d'une approche basée sur le contenu.

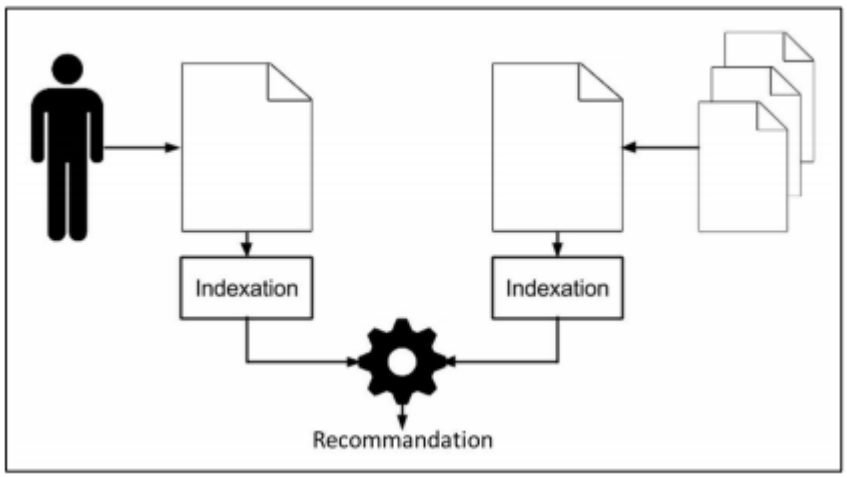

Figure 1: Approche basée sur le contenu

4.1.1.2 Processus des approches basées sur le contenu

Cette approche qui est aussi appelée filtrage cognitif s'appuie sur le contenu des *items* pour les comparer à un profil lui-même constitué de thèmes. Chaque utilisateur du système possède alors un profil qui décrit des centres d'intérêt. Par exemple, le profil peut contenir

une liste des thèmes ou préférences que l'utilisateur aime ou n'aime pas. Lors de l'arrivée d'un nouvel *item*, le système prédit l'utilité d'un *item* pour l'utilisateur en comparant son profil avec le descriptif de *l'item* en question.

4.1.1.3 Types de recommandation basée sur le contenu

On distingue deux types de recommandation basée sur le contenu :

- **Recommandation basée sur les mots-clefs** : elle consiste à utiliser des mots-clés qui sont extraits grâce à une indexation automatique et qui sont représentatifs de chaque item ou profil. C'est une méthode qui est généralement utilisée pour la recommandation de films, de restaurants ou de page web.
- **Recommandation basée sur la sémantique** : ce sont des systèmes qui évoluent au rythme des nouvelles technologies du Web sémantique, afin de remédier à certains manques des systèmes basés sur le contenu classique.

4.1.2 Les approches basées sur le filtrage collaboratif

4.1.2.1 Définition

Les systèmes basés sur le filtrage collaboratif prennent en considération les évaluations des utilisateurs sur des *items* afin de calculer la similarité de leurs préférences et de recommander ces *items* sans pour autant effectuer une analyse de leurs contenus.

La figure 2 résume le principe d'une approche basée sur le filtrage collaboratif.

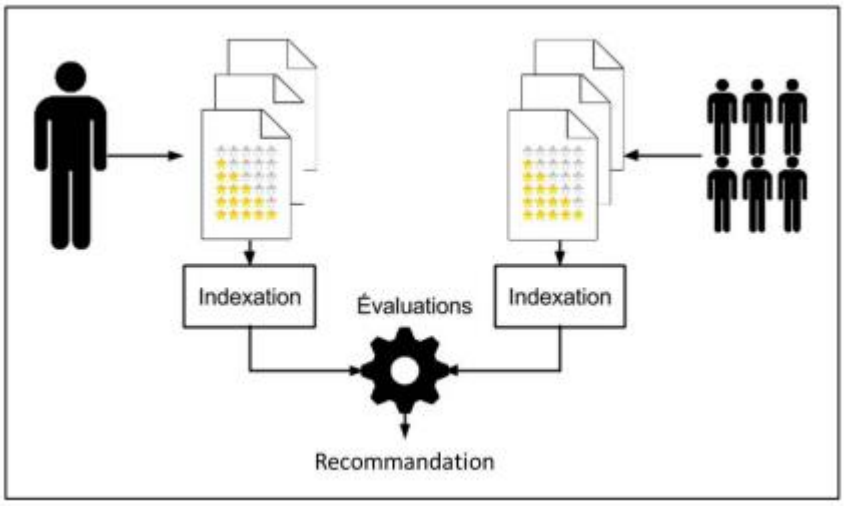

Figure 2 : Approche basée sur le filtrage collaboratif

4.1.2.2 Processus du filtrage collaboratif

Le processus du filtrage collaboratif se fait selon les étapes suivantes :

1. Évaluation des recommandations : l'utilisateur doit fournir une évaluation explicite (notes) ou implicite sur les *items*.
2. Formation des communautés : le système se base sur la similarité des préférences des utilisateurs, c'est-à-dire la proximité de leurs évaluations pour former des communautés.
3. Production des recommandations : le système va tout simplement s'appuyer sur l'évaluation de la communauté d'un utilisateur concernant un *item* pour le lui suggérer.

4.1.2.3 Types de recommandation basée sur le filtrage collaboratif

On distingue deux types de recommandation basée sur le filtrage collaboratif :

- **Filtrage collaboratif basé sur la mémoire** : consiste à prédire l'intérêt des *items* pour un utilisateur en se basant sur la totalité des évaluations des autres utilisateurs disponibles au moment du calcul de la recommandation.
- **Filtrage collaboratif basé sur un modèle** : se base aussi sur les évaluations des autres utilisateurs, cependant la prédiction ne se fera pas directement mais en utilisant une classification d'utilisateurs en groupe.

4.1.3 Les approches hybrides

4.1.3.1 Définition

Une approche est dites hybride quand celle-ci combine deux ou plusieurs approches de recommandation différentes afin de tirer parti de leurs avantages respectifs et d'en pallier les limites.

4.1.3.2 Processus des approches hybrides

Se fait selon ces deux étapes :

1. Effectuer de manière indépendante les filtrages des *items* via des méthodes collaboratives ou par le contenu (ou autres) pour générer des recommandations dites candidates.

2. Combiner ces ensembles de recommandations via des méthodes d'hybridations telles que des pondérations, commutations, cascade afin de produire les recommandations finales pour les utilisateurs.

4.2 Classification de Burke

Selon Burke [8] trois éléments constituent un système de recommandation :

- les informations que le système possède avant la recommandation ;
- les informations que l'utilisateur doit communiquer au système pour que ce dernier puisse faire des recommandations ;
- un algorithme qui combine ces deux informations pour parvenir à ses suggestions.

Il a ainsi proposé trois classes de recommandation:

- la recommandation basée sur les données démographiques ;
- la recommandation basée sur l'utilité ;
- la recommandation basée sur la connaissance.

4.2.1 Recommandation basée sur les données démographiques

C'est une recommandation simple qui se base sur les informations démographiques d'un utilisateur (l'âge, le métier, le pays, etc.). En effet, si ce dernier évolue dans un environnement similaire qu'un autre utilisateur, ils partageront des goûts communs.

4.2.2 Recommandation basée sur l'utilité

Consiste à faire des suggestions en calculant l'utilité de chaque *item* pour l'utilisateur. Bien entendu, le problème central est de créer une fonction d'intérêt pour chaque utilisateur [10].

4.2.3 Recommandation basée sur la connaissance

Reconnus comme étant une recommandation fiable, elle consiste à collecter le maximum d'informations sur un utilisateur pour pouvoir ensuite lui recommander des *items*.

4.3 Autres classifications

Il existe d'autres classifications des systèmes de recommandation [11].

4.3.1 Recommandation objet

Consiste à recommander des *items* (contenus) en corrélant leurs qualités et propriétés avec les préférences de l'utilisateur. Ce genre de système va tous simplement créer un profil pour chaque *item*, qui contiendra un certain nombre d'attributs propres à lui-même pour pouvoir recommander des items similaires. Cette méthode se base sur la similarité des propriétés.

4.3.2 Recommandation sociale

Cette approche se base sur la similarité du comportement et activités des utilisateurs. Par exemple, si deux personnes aiment un livre d'un auteur X et que l'un d'eux aime un autre auteur Y, le système va ainsi recommander l'auteur Y à l'autre personne.

4.3.3 Recommandation personnalisée

Cette approche consiste à recommander des *items* sur la base du comportement passé de l'utilisateur, c'est-à-dire sur la base de son historique.

4.3.4 Recommandation éditoriale

Lorsqu'un visiteur nouveau navigue sur un site et que le système ne possède aucune information sur ce dernier, il va tout simplement lui présenter les *items* populaires et ayant les meilleures notes pour attirer son attention et lui donner envie de parcourir le site.

4.3.5 Recommandation contextuelle

Cette recommandation se base sur les *items* déjà consultés. Prenons l'exemple de Youtube, si l'utilisateur regarde une vidéo d'un certain réalisateur, le système va ainsi lui recommander des vidéos du même réalisateur, grâce à des techniques de rapprochement *d'items* simples ou complexes basées sur les usages.

5 Avantages et inconvénients des systèmes de recommandation

Les différentes approches utilisées dans les systèmes de recommandation possèdent des avantages et des inconvénients.

5.1 Les avantages

Les avantages du filtrage basé sur le contenu :

- Recommande des *items* similaires à ceux que l'utilisateur a déjà aimés.

- Possibilité d'effectuer des recommandations sans une large communauté d'utilisateurs car leurs données sont inutiles.
- Seuls les goûts uniques de l'utilisateur comptent et non les informations sur les autres utilisateurs.
- Une liste de recommandations peut être générée même s'il n'y a qu'un seul utilisateur et elle devient plus précise avec le temps.
- Possibilité de recommander de nouveaux éléments ou ceux qui ne sont pas populaires.

Les avantages du filtrage collaboratif :

- Ne demande aucune connaissance ou analyse sur le contenu de *l'item*.
- La qualité de la recommandation peut être évaluée.
- Plus le nombre d'utilisateurs accroît, plus la recommandation est meilleure.

5.2 Les inconvénients

Les inconvénients du filtrage basé sur le contenu :

- Pour faire une bonne recommandation, il faut effectuer une analyse du contenu.
- Nécessite de connaitre le profil de l'utilisateur.
- Risque de sur-spécialisation, c'est-à-dire que les réponses soient trop homogènes et que l'on se limite aux éléments similaires seulement.

Les inconvénients du filtrage collaboratif :

- Démarrage à froid : quand un utilisateur est nouveau et que le système ne dispose d'aucune information sur lui, il est difficile de recommander des produits pertinents.
- Le problème du mouton gris (*gray sheep*) : Les utilisateurs qui ont des goûts étranges, c'est-à-dire des goûts qui varient de la norme ou qui sortent du commun, n'auront pas beaucoup d'utilisateurs voisins. Il sera donc difficile de faire des recommandations pertinentes pour ce genre d'utilisateurs.
- La complexité : due au grand nombre *d'items* et d'utilisateurs.

6 Techniques adoptées par les systèmes de recommandation

Les systèmes de recommandation utilisent différentes méthodes pour parvenir à des prédictions pertinentes aux yeux d'un utilisateur. Différents algorithmes d'apprentissage et mesures de similarité ont ainsi été développés dans ce sens.

6.1 Les algorithmes d'apprentissage

6.1.1 Pertinence de Rocchio (*Rocchio's relevance feedback*)

Pertinence de Rocchio (12] est un algorithme destiné à l'amélioration des systèmes de recherche documentaires. Il repose sur l'hypothèse que la plupart des utilisateurs ont une conception générale des *items* à désigner comme pertinents ou non pertinents. Les commentaires des utilisateurs sont utilisés pour affiner une requête de recherche. Le système exploite ainsi le jugement de l'utilisateur sur des *items* afin de perfectionner ses suggestions.

6.1.2 K-plus proche voisin (*K-nearest neighbor*)

Il représente l'un des algorithmes les plus couramment utilisés dans les systèmes de recommandation. Il consiste à effectuer une classification d'une entrée basée sur les propriétés de ses voisins les plus proches dans l'espace. C'est une version de base de l'algorithme. Très efficace pour de nombreux problèmes de classification, en particulier avec une dimensionnalité faible (petit nombre de caractéristiques).

6.1.3 RecTree

Algorithme de filtrage collaboratif appelé l'arbre de recommandation, son objectif principal est de fractionner les données dans des cliques d'utilisateurs semblables tout en maximisant les similarités entre les membres d'une même clique et minimisant celles entre les membres de deux cliques différentes.

Tout d'abord, l'algorithme K-Means sera utilisé afin de diviser l'ensemble des données. Il consiste à choisir k graines initiales comme les k centres provisoires de groupe. Ensuite, il assigne des utilisateurs au groupe dont ils sont les plus proches. Le centre de chaque groupe est pris comme le nouveau centre et les utilisateurs sont réassignés. Ces étapes sont répétées jusqu'à ce que le changement des positions des centres tombe au-dessous d'un seuil. Par la suite, l'arbre de recommandation sera calculé. Ainsi, plus on descend dans l'arbre et plus les clusters sont spécifiques à un certain groupe d'utilisateurs similaires. Par

conséquent, plus on parcourt l'arbre en profondeur, plus les individus partagent le même avis.

6.1.4 Les réseaux de neurones

L'apprentissage automatique est fréquemment utilisé pour créer des systèmes de recommandation. Particulièrement les réseaux de neurones qui sont des modèles de calcul dont la conception est très schématiquement inspirée du fonctionnement de vrais neurones. Ils sont généralement optimisés par des méthodes d'apprentissage de type statistique. Grâce à leurs capacités de classification et de généralisation, ils enrichissent avec un ensemble de paradigmes permettant de générer de vastes espaces fonctionnels, souples et partiellement structurés. Ils permettent ainsi d'effectuer des recommandations pertinentes pour un utilisateur. Nous pouvons citer, à titre d'exemple, la mise en œuvre des réseaux profonds dans le système de recommandation de YouTube [13]

6.2 Les mesures de similarité

Pour extraire une liste de suggestions, les algorithmes utilisent la notion de mesure de similarité. Cette dernière a pour but de donner une valeur à la ressemblance entre 2 *items* ou profils. Plus la ressemblance est forte, plus la valeur de la similarité sera grande. À l'inverse, plus la ressemblance est faible, plus la valeur de la similarité sera petite.

Il existe différentes mesures de similarité.

6.2.1 Mesure de cosine

Cette mesure utilise la représentation vectorielle complète, c'est-à-dire la fréquence des objets. Deux objets sont similaires si leurs vecteurs sont confondus. Ainsi, la mesure de cosine quantifie la similarité numérique entre ces deux vecteurs.

La formule est la suivante : $\mathrm{Sim}\,(\mathrm{X}, \mathrm{Y}) = \cos\,(\mathrm{X}, \mathrm{Y}) = \dfrac{X * Y}{||\mathrm{X}||_2 * ||\mathrm{Y}||_2}$

6.2.2 Corrélation de Pearson

La mesure de similarité de Pearson est basée sur le calcul de la corrélation (la similarité entre deux usagers). Elle prend une valeur entre -1 et 1. Deux vecteurs identiques sont parfaitement corrélés (corrélation = 1), alors que deux vecteurs opposés sont parfaitement anti corrélés (corrélation = -1).

7 Conclusion

Dans ce chapitre, nous avons présenté les systèmes de recommandation et abordé les différentes classifications de ces systèmes notamment la classification classique qui comprend le filtrage collaboratif, le filtrage contenu et le filtrage hybride. Par la suite, nous nous sommes intéressés aux différentes techniques utilisées dans les systèmes de recommandation. Parmi elles, les techniques à base d'apprentissage automatique, notamment à l'aide de réseaux de neurones sont les plus prometteuses. Nous avons donc choisi d'adopter cette approche dans notre travail.

Ainsi, nous présentons plus en profondeur l'apprentissage automatique dans le chapitre qui suit, particulièrement les réseaux de neurones qui ont apporté des succès majeurs dans le domaine de la recommandation.

Chapitre 2: Apprentissage automatique

1 Introduction

L'intelligence artificielle (IA) est apparue en 1956. C'est l'une des disciplines scientifiques et d'ingénierie les plus récentes et les plus florissantes. Elle représente la capacité d'un programme informatique à fonctionner comme un cerveau humain. Ce qui peut se révéler bien pratique pour différentes tâches telles que la recommandation qui fait l'objet de ce mémoire.

L'apprentissage automatique (ou *machine learning* en anglais) est une sous-partie de l'intelligence artificielle qui consiste à reproduire un comportement non pas en le programmant à la main dans un ordinateur, mais en concevant un système plus général capable d'apprendre à partir d'exemples.

La discipline de l'apprentissage automatique possède de riches fondements théoriques [14].En effet, elle a beaucoup puisé dans les mathématiques et les statistiques, et plus précisément des mathématiques pour l'informatique : algèbre linéaire, la probabilité et la logique.

Dans ce chapitre, nous examinons en premier lieu les différents types d'apprentissage automatique. Ensuite, nous introduisons la notion de réseau de neurones tout en expliquant quelques généralités telles que les fonctions d'activations, les algorithmes d'apprentissage et les différentes architectures de réseaux de neurones. Enfin, nous concluons par quelques exemples d'architectures de réseaux de neurones utilisées dans les systèmes de recommandation et par l'intégration de l'apprentissage par renforcement.

2 Généralités sur l'apprentissage automatique

2.1 Définition

L'apprentissage est le processus qui permet de construire un modèle général à partir de données (observations) particulières du monde réel.

Arthur Samuel, un pionnier dans le domaine de l'intelligence artificielle et des jeux informatiques, a inventé le terme « *Machine Learning* ». Il a défini l'apprentissage automatique comme « un domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés ».

D'une manière très profane, l'apprentissage automatique se concentre sur l'élaboration de modèles capables de représenter certaines caractéristiques du monde qui nous entoure, d'apprendre certaines propriétés statistiques des distributions des données qu'ils traitent, afin d'accomplir diverses tâches. Le rapport à l'intelligence vient de la capacité de ces modèles à généraliser, c'est-à-dire à extraire l'information pertinente de données étudiées au fil d'un processus de mise-à-jour appelé entraînement et de savoir la réutiliser avec efficacité sur de nouvelles données jamais rencontrées auparavant.

Il existe différents algorithmes et techniques d'apprentissage automatique qui se différencient par le type de données dont dispose le système pour apprendre.

2.1.1 Apprentissage supervisé

L'apprentissage est dit supervisé lorsque les données utilisées dans ce processus sont déjà catégorisées. En d'autres termes, le modèle d'apprentissage sera formé à l'aide de données bien étiquetées (classées). Les algorithmes tirent des enseignements à partir de ces dernières et déterminent ensuite l'étiquette à attribuer aux nouvelles données.

L'apprentissage supervisé utilise des variables d'entrée (x), une variable de sortie désirée (Y) et un algorithme pour apprendre la fonction de prédiction (f). La sortie est calculée comme suit: $Y = f(X)$

L'objectif de base est de définir la fonction de prédiction (mentionnée ci-dessus) de telle sorte à ce que pour de nouvelles données d'entrée (inconnues auparavant), la variable de sortie correspondante peut être prédite.

Sur la base des jeux de données donnés, le problème de l'apprentissage automatique est classé en deux types : la classification et la régression. Si les données fournies ont des valeurs de sortie discrètes (c'est-à-dire un ensemble d'étiquettes ou de classes de sortie), il s'agit d'un problème de classification. Si l'ensemble des données comporte des valeurs en sortie d'attributs numériques continues (sans étiquettes cibles), il s'agit alors d'un problème de régression.

Lors de l'apprentissage supervisé, les données utilisées pour construire le modèle sont appelées données d'apprentissage (ou d'entrainement). Elles comportent aussi bien les valeurs d'entrée que les valeurs de sorties correspondantes. Elles sont utilisées lors d'une première phase appelée phase d'apprentissage (ou d'entrainement) dont le but est de

construire et de générer un modèle entrainé pouvant faire des prédictions sur un jeu de données potentiellement nouveau. Ensuite, vient la phase de validation qui utilise un autre ensemble de données qui comporte toujours des valeurs d'entrée et des valeurs de sortie souhaitées. Ces données sont utilisées pour valider le modèle. Ainsi, les valeurs en entrées sont utilisées pour faire des prédictions qui permettent ensuite, de valider le modèle ou d'ajuster les paramètres d'apprentissage. Une fois, le modèle validé, un troisième ensemble, appelé ensemble de test, contenant toujours des valeurs d'entrée et des valeurs de sortie souhaitées, est utilisé pour évaluer les prédictions du modèle. Cette fois-ci, il n'y a pas d'ajustement des paramètres d'apprentissage. La séparation des étapes de validation et de test permet d'éviter de fausser les résultats d'évaluation en choisissant des paramètres d'apprentissage favorisant les prédictions sur un jeu de données restreint.

2.1.2 Apprentissage non-supervisé

L'apprentissage non supervisé consiste à détecter les similarités dans les données qu'il reçoit pour les organiser en fonction de ces dernières.

L'apprentissage non-supervisé (ou *Clustering*) est utilisé quand il n'y a pas de variables à prédire et que le nombre de classes et leur nature ne sont pas prédéterminés. L'algorithme doit découvrir par lui-même la structure des données et séparer les données en différents groupes (clusters) homogènes (grâce à une notion de similarité).

Dans ce type d'apprentissage, l'ensemble d'entrainement comprend des données d'entrées, mais pas des données de sorties. Ce qui signifie qu'il se fait de manière totalement autonome sans l'aide d'un superviseur.

2.1.3 Apprentissage par renforcement

L'apprentissage par renforcement (RL pour *Reinforcement Learning*) est un autre type d'apprentissage qui apprend en permanence à partir de données recueillies suite à des expériences successives qui ne sont pas forcément réalisées au même moment (elles sont étalées dans le temps) dans le but de trouver la meilleure solution (ou prédiction) à un problème donné (ou à un certain nombre de valeurs d'entrée données).

Ainsi, il permet à un agent de renforcement (l'algorithme) de déterminer automatiquement le comportement idéal dans un contexte spécifique, afin d'optimiser ses

performances. Un simple retour d'expérience sous forme d'une récompense ou d'une pénalité est alors utilisé pour qu'il puisse ajuster son comportement.

L'agent de renforcement interagit avec l'environnement, essaie plusieurs solutions (on parle d'exploration), observe la réaction de l'environnement et enfin adapte son comportement pour trouver la meilleure stratégie (il exploite le résultat de ses explorations). En retour, l'environnement procure à l'agent une récompense, qui peut être positive ou négative.

Formellement, la base du modèle d'apprentissage par renforcement consiste en un ensemble d'états S de l'agent dans l'environnement, un ensemble d'actions A que l'agent peut effectuer et un ensemble de valeurs scalaires (ou récompenses) R que l'agent peut obtenir. Ce modèle est utilisé dans un scenario d'apprentissage qui typiquement est construit ainsi :

1. L'agent effectue une action sur l'environnement ;
2. Cette action génère un résultat sous forme d'une récompense(ou d'une pénalité) qui sera une représentation du nouvel état ;
3. Ensuite, cette nouvelle représentation sera transmise à l'agent qui va ajuster ses paramètres (et donc son fonctionnement).

En conséquence, le modèle peut continuer d'augmenter ses performances tout au long de son utilisation (contrairement à un modèle d'apprentissage supervisé par exemple, qui n'apprend que lors de l'étape d'apprentissage).

La figure 3 résume le scénario de l'apprentissage par renforcement.

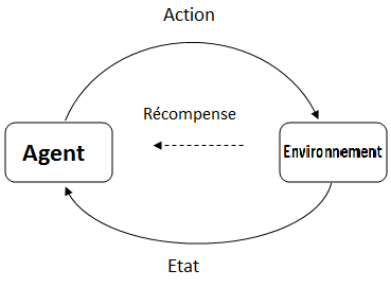

Figure 3 : Scénario de l'apprentissage par renforcement

2.1.4 Apprentissage par transfert

L'apprentissage par transfert (*transfer learning* en anglais) est une méthode d'apprentissage automatique dans laquelle un modèle développé pour une tâche est réutilisé comme point de départ d'un modèle pour une seconde tâche. En effet, ce qui a été appris dans un premier contexte sera exploité pour améliorer ou pour accélérer l'apprentissage dans un autre contexte.

L'apprentissage par transfert est une optimisation qui permet de progresser rapidement ou d'améliorer les performances d'apprentissage.

3 Généralité sur les réseaux de neurones

3.1 Neurone formel

Définition : Un neurone formel aussi appelé neurone de McCulloch et Pitts [16] est un opérateur mathématique très simple qui traduit la représentation mathématique d'un neurone biologique ayant plusieurs entrées (dendrites) et une sortie (axone). Il est utilisé en intelligence artificielle, en association avec ses semblables pour former un réseau de neurones.

Le neurone formel reçoit des signaux en provenance d'autres neurones. Ensuite, il calcule la somme pondérée de ces entrées reçues à laquelle il applique une fonction d'activation pour obtenir la sortie du neurone.

Les composants élémentaires d'un neurone formel sont :

- Entrée : représente le vecteur d'entrée à analyser.
- Poids : représente le paramètre $\mathbf{w_i}$ qui est le facteur multiplicateur qui affecte l'influence de chaque entrée sur la sortie du neurone.
- Biais : représente une unité fictive associée à un poids qui permet de contrôler si un neurone est actif.
- Fonction d'activation : c'est une fonction mathématique qui représente le seuil à partir duquel un neurone va émettre un signal.
- Sortie du neurone : représente l'application de la fonction d'activation à la somme pondérée des signaux d'entrée.

La figure 4 montre la structure d'un neurone formel.

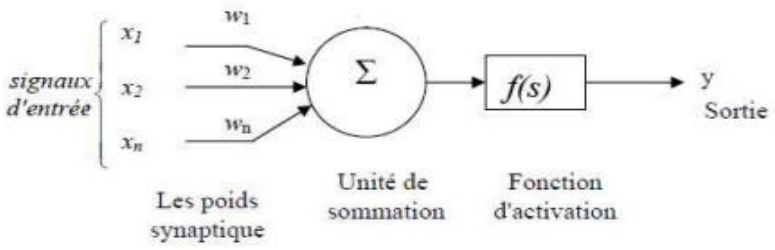

Figure 4 : Structure d'un neurone formel

3.2 Réseaux de neurones

Définition : un réseau de neurones artificiel (ANN) est défini par Hecht-Nielsen comme « un système informatique composé d'un certain nombre d'éléments de traitement simples et fortement interconnectés, qui traitent les informations en fonction de leur réponse dynamique aux entrées externes » [17].

Les réseaux de neurones sont généralement organisés en couches. Les couches sont constituées d'un certain nombre de nœuds (neurones) interconnectés contenant une fonction d'activation. Les données sont présentées au réseau via la couche d'entrée, qui communique avec une ou plusieurs couches cachées où le traitement a proprement dit est effectué via un système de connexions pondérées. Les couches masquées sont ensuite liées à une couche de sortie pour produire une réponse.

Considérée comme une méthode d'apprentissage automatique, les réseaux de neurones sont utilisés entre autres pour des problématiques de prédiction et de classification.

La figure 5 montre la structure d'un réseau de neurones :

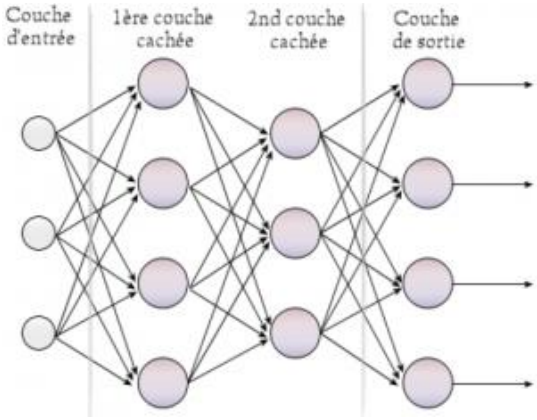

Figure 5 : Structure d'un réseau de neurones à 2 couches cachées

Fonctionnement de l'apprentissage dans un réseau de neurones :

Le réseau de neurones reçoit des entrées pondérées auxquelles il applique une fonction d'activation qui calcule une sortie pour chaque neurone. Par la suite, il modifie et ajuste ses poids avec un hyper-paramètre (paramètre configurable utilisé pendant l'entrainement) appelé taux d'apprentissage (*learning rate*) afin qu'une entrée donnée produise une sortie souhaitée. L'ajustement est calculé avec un algorithme d'apprentissage basé sur une erreur calculée entre la sortie prédite et celle souhaitée.

Ce processus est itératif. Nous appelons chaque itération époque (*epoch*). Dans chaque itération, nous passons sur toutes les données.

3.3 Fonctions d'activations usuelles

La fonction d'activation ou fonction de transfert est une porte mathématique entre l'entrée alimentant le neurone actuel et sa sortie allant à la couche suivante. C'est l'une des caractéristiques les plus importantes des réseaux de neurones artificiels. Elle est considérée comme une fonction qui active et désactive la sortie du neurone. Sa représentation mathématique est comme suit : $Y = F_{activation}(b + \sum Wi * Xi)$

- x_i : sont les signaux d'entrées.
- w_i : les poids (Weight).
- b : le biais, un neurone qui a toujours comme entrée la valeur 1.

Nous présentons dans ce qui suit quelques-unes des fonctions d'activations les plus utilisées dans les réseaux de neurones.

3.3.1 Fonction linéaire

Une fonction linéaire est tout simplement une fonction en ligne droite où l'activation est proportionnelle à l'entrée (qui est la somme pondérée du neurone). Sa fonction est de la forme : $A = cx$

Sa courbe est présentée dans la figure 6.

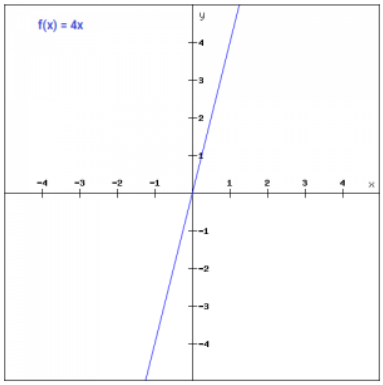

Figure 6 : Courbe de la fonction linéaire

3.3.2 Fonction sigmoïde

La fonction Sigmoïde est une fonction d'activation non linéaire qui permet de propager facilement les erreurs en arrière. Elle est de la forme : $F(x) = \frac{1}{1+e^{-x}}$

L'avantage de cette fonction d'activation réside dans le fait que la sortie sera toujours comprise entre 0 et 1. Elle permet aussi de faire des prédictions claires pour X > 2 ou X < -2. Cependant, c'est une fonction couteuse en termes de calcul. Sa courbe est de la forme d'un « S » comme le montre la figure 7 ci-dessous.

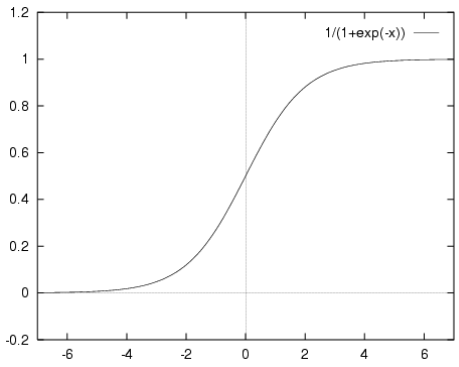

Figure 7 : Courbe de la fonction Sigmoïde

3.3.3 Fonction softmax

La fonction softmax est un type de fonction sigmoïde, elle est pratique pour gérer des problèmes de classification. Elle est généralement utilisée uniquement pour la couche de sortie, pour les réseaux de neurones devant classer les entrées en plusieurs catégories.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \quad Pour\ j = 1 \dots k$$

3.3.4 Fonction tanh

La fonction tanh (tangente hyperbolique) est très similaire à la fonction sigmoïde, elle est très populaire et largement utilisée. C'est une fonction non linéaire qui varie entre -1 et 1.

Sa forme est comme suit : $F(x) = \tanh(x) = \frac{2}{1+e^{-2x}}$

La figure 8 nous montre sa courbe.

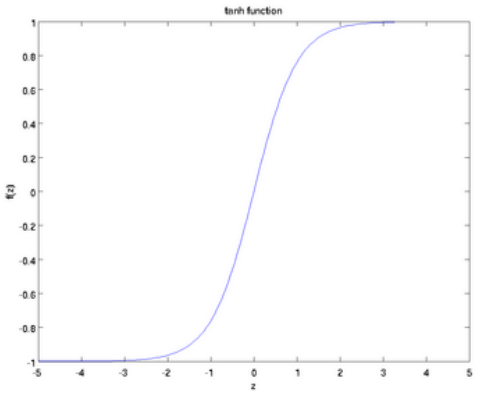

Figure 8 : Courbe de la fonction tanh

3.3.5 Fonction Relu et Leaky Relu

3.3.5.1 Relu

La fonction ReLu (en anglais *Rectified Linear Units*) signifie «unités linéaires rectifiées », est définie par : $A(x) = \max(0, x)$

C'est une fonction non linéaire qui donne une sortie x si x est positif et 0 sinon.

La figure 9 montre la courbe de la fonction Relu.

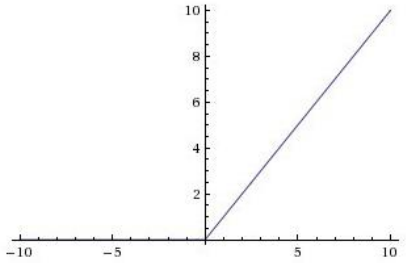

Figure 9 : Courbe de la fonction Relu

3.3.5.2 *Leaky Relu*

La fonction Leaky Relu n'est autre qu'une version améliorée de la fonction Relu. En effet, le gradient[1] de relu est à 0 pour x < 0. Ceci provoque l'inactivité des neurones. La fonction Leaky Relu permet de résoudre ce problème. Au lieu de définir la fonction Relu à 0 pour x < 0, nous la définissons comme une petite composante linéaire de x comme suit :

$$F(x) = \begin{cases} ax & x < 0 \\ x & x \geq 0 \end{cases}$$

Cependant, la fonction Leaky Relu ne fournit pas de prédictions cohérentes pour les valeurs d'entrée négatives.

Sa courbe est de la forme suivante (figure 10).

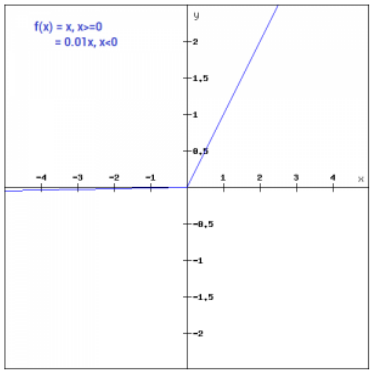

Figure 10 : Courbe de la fonction Leaky Relu

4 Réseaux de neurones et algorithmes d'apprentissage

Les réseaux de neurones sont basés sur des modèles informatiques pour la logique de seuil. La logique de seuil est une combinaison d'algorithmes et de mathématiques.

Un algorithme est une suite d'actions (dont des calculs) à réaliser pour parvenir à un résultat. Les algorithmes d'apprentissage sont considérés comme étant le mécanisme central par lequel les réseaux de neurones apprennent. En effet, ils indiquent au réseau si ce dernier a commis une erreur ou pas lorsqu'il a fait une prédiction.

Le réseau de neurones va ainsi être modifié en fonction des exemples d'apprentissage, incluant les poids de ses connexions, son architecture, etc. La problématique de

[1] Gradient : désigne un vecteur représentant la variation d'une fonction par rapport à la variation de ses différents paramètres.

l'apprentissage se présente souvent comme la minimisation de l'erreur générée par le modèle.

Parmi ces algorithmes d'apprentissage, nous avons : la rétro-propagation du gradient, adagrad et adam.

4.1 La rétro-propagation du gradient

L'algorithme de rétro-propagation a été introduit dans les années 1970. Il est le fer de lance de l'apprentissage dans les réseaux de neurones.

La rétro-propagation (en anglais *back-propagation*) est une méthode de calcul des poids qui consiste à minimiser l'erreur quadratique de sortie (somme des carrés de l'erreur de chaque composante entre la sortie obtenue et la sortie désirée).

La propagation est précédée par la propagation en avant (en anglais *forward-propagation*) où les entrées se propagent en avant dans le réseau pour calculer une sortie qui sera comparée à la sortie désirée pour obtenir une erreur. Ensuite, cette dernière, sera propagée dans le sens inverse de la propagation des activations (de la couche de sortie vers la couche d'entrée), en calculant l'erreur individuelle de chaque neurone puis en ajustant les poids du réseau afin de minimiser l'erreur globale. Dans ce cas, nous parlons de rétro-propagation (en anglais *backward-propagation*).

La rétro-propagation du gradient consiste donc à effectuer une rétro-propagation. C'est un algorithme itératif qui s'applique autant de fois que nécessaire pour obtenir une meilleure prédiction et ainsi diminuer l'erreur dans le réseau.

La figure 11 résume le principe de la rétro-propagation du gradient.

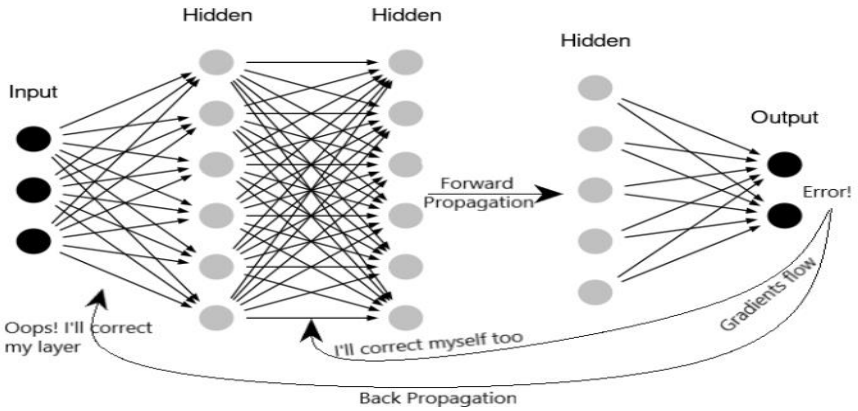

Figure 11 : Principe de la rétro-propagation du gradient

4.2 Adagrad

Adagrad est un algorithme de gradient adaptatif qui donne de bons résultats dans des tâches d'apprentissage automatique à grande échelle [18].

Adagrad a été conçu pour être utilisé dans des problèmes où il est impossible de choisir manuellement différents taux d'apprentissage pour chaque paramètre. Afin de s'assurer que le processus d'apprentissage n'est ni trop lent, ni trop volatile et imprécis, l'algorithme d'Adagrad ajuste de manière adaptative le paramètre de taux d'apprentissage.

L'algorithme se déroule ainsi :

1. Il intègre de manière dynamique la représentation des données observées lors des itérations précédentes.
2. Il applique ces représentation pour définir des taux d'apprentissage plus faibles pour les données présentant des caractéristiques les plus fréquentes et des taux d'apprentissage plus élevés pour les données présentant des caractéristiques relativement peu fréquentes.
3. Il offre ainsi la possibilité aux caractéristiques peu fréquentes de se démarquer, ce qui permet d'identifier des caractéristiques rares, mais très prédictives.

4.3 Adam

Adam (nom dérivé de : *Adaptive Moment Estimation*) est un algorithme qui calcule des taux d'apprentissage adaptatifs.

L'algorithme se déroule ainsi :

1. Tout d'abord, il calcule une moyenne exponentielle pondérée des gradients passés.
2. Ensuite, il calcule une moyenne exponentielle pondérée des carrés des gradients passés.
3. Enfin, il met à jour les poids dans une direction basée sur les informations précédentes.

Adam est basé sur l'algorithme Adagrad. La différence entre ces deux algorithmes réside dans la fonction abstraite qui modifie le taux d'apprentissage des paramètres. En effet, dans

Adagrad, la fonction abstraite est la somme de tous les gradients passés et actuels contrairement à Adam où c'est la moyenne mobile des gradients carrés antérieurs [19].

5 Architectures de réseaux de neurones

Les réseaux de neurones présentent différentes architectures. On distingue deux types de réseaux de neurones : les réseaux non bouclés et les réseaux bouclés (récurrents), que nous présentons dans ce qui suit. Il y a également un autre type de réseaux de neurones, dit convolutionnel. Bien que celui-ci soit a priori, un réseau non bouclé, il présente des mécanismes de fonctionnement différents de ceux d'un réseau de neurone non bouclé classique. Nous présentons donc ce type de réseaux séparément dans une autre section.

5.1 Réseaux de neurones non bouclés

Un réseau de neurones non bouclé est représenté graphiquement par un ensemble de neurones connectés entre eux où l'information circule des entrées vers les sorties sans «retour en arrière» [20]. Le graphe d'un réseau non bouclé est acyclique. En effet, dans ce type de réseau le déplacement à partir d'un neurone quelconque en suivant les connexions, ne permet pas le retour au neurone de départ. Les neurones qui effectuent le dernier calcul de la composition de fonctions sont les neurones de sortie. Ceux qui effectuent des calculs intermédiaires sont les neurones cachés.

La figure 12 illustre la structure d'un réseau de neurones non bouclé [20].

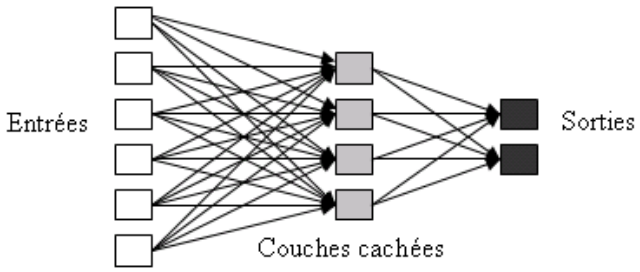

Figure 12 : Structure d'un réseau de neurones non bouclé

Les réseaux non bouclés à couches sont structurés de telle sorte à ce que les neurones qui appartiennent à une même couche ne soient pas connectés entre eux. Chacune des couches reçoit des signaux de la couche précédente et transmet le résultat de ses traitements à la couche suivante. Les deux couches extrêmes correspondent à la couche

d'entrée qui reçoit ses entrées du milieu extérieur d'une part et à la couche de sortie qui fournit le résultat des traitements effectués d'autre part. Les couches intermédiaires sont appelées couches cachées, leur nombre est variable.

5.1.1 Le perceptron

Frank Rosenblatt, un psychologue américain, a proposé le modèle classique du perceptron en 1958. Raffiné et analysé avec soin par Minsky et Papert (1969), ce modèle est représenté dans la figure 13 ci-dessous.

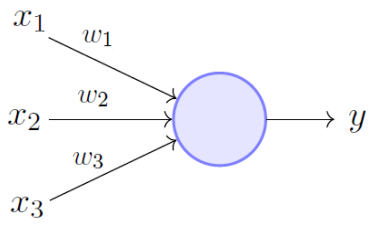

Figure 13 : Modèle de perceptron de Minsky –Papert (1969)

Le perceptron est un réseau acyclique dont la dynamique (l'activité) est déclenchée par la réception en entrée d'information. C'est un réseau dit simple car il ne se compose que de deux couches : une couche d'entrée (X_i) et une couche de sortie (Y) ce qui implique une seule matrice de poids (W_i). Les unités de la couche d'entrée sont connectées à celles de la couche de sortie. Ces structures lui permettent d'être considéré comme un classifieur linéaire. Autrement dit, le perceptron est capable de classer des entrées et décider si elles correspondent à un côté d'un hyper- plan ou bien à un autre.

5.1.2 Couches entièrement connectées

Les réseaux de neurones entièrement connectés sont des réseaux profonds. En effet, plus ils contiennent de couches cachées, plus leur profondeur est grande. Ils se composent d'une couche d'entrée, d'une couche de sortie et d'une ou plusieurs couches cachées. Chaque neurone d'une couche est connecté à tous les neurones de la couche qui le précèdent et qui le succèdent directement. Par contre, il n'y a pas de connexions entre les neurones d'une même couche.

Contrairement au perceptron à couche unique qui ne peut apprendre que des fonctions linéaires, un réseau entièrement connecté peut apprendre des fonctions non

linéaires. Leur apprentissage permet d'ailleurs d'obtenir des résultats très précis notamment dans la reconnaissance d'image ou encore la traduction de textes.

La figure 14 illustre la structure d'un réseau de neurones entièrement connecté à deux couches cachées.

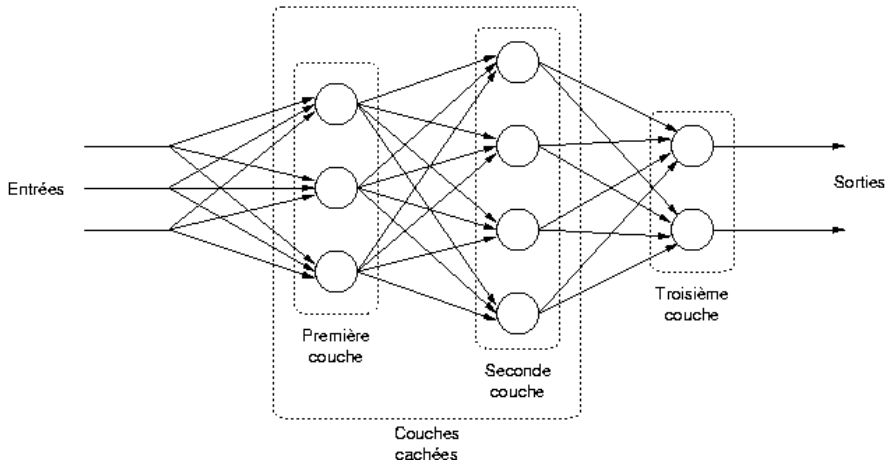

Figure 14 : Structure d'un réseau de neurones entièrement connecté

5.2 Réseaux récurrents

Un réseau de neurones récurrent (en anglais *Recurrent Neural Network*) ou RNN est un réseau de neurones bouclé qui est schématisé par un graphe cyclique. Il peut être considéré comme plusieurs copies du même réseau, chacune transmettant un message à un successeur.

Le réseau de neurones récurrent est un type de réseau de neurones dans lequel les sorties de l'entrée précédente sont alimentées par des entrées de l'étape en cours. Dans les réseaux de neurones traditionnels, toutes les entrées et toutes les sorties sont indépendantes les unes des autres, mais dans certains cas de prédiction (par exemple faisant intervenir du texte avec une suite de mots), il est nécessaire de se rappeler de l'état précédent du réseau (calculé pour le mot précédent dans un texte). C'est ainsi que sont nés les RNNs, apportant ainsi une solution à ce problème à l'aide d'une couche cachée bouclée. Sa caractéristique principale est l'état caché, qui garde en mémoire certaines informations sur ce qui a été calculé pour une entrée précédente.

Le RNN utilise les mêmes paramètres pour chaque entrée car il effectue la même tâche sur toutes les entrées ou les couches masquées pour produire la sortie. Cela réduit la

complexité et le nombre de paramètres (de poids) à entrainer, contrairement à un réseau de neurones équivalent utilisant uniquement des couches entièrement connectées.

La figure 15 ci-dessous illustre l'architecture d'un RNN.

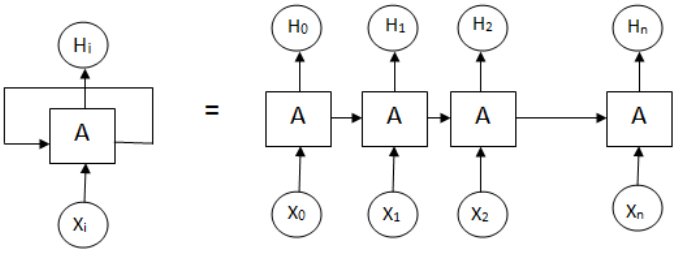

Figure 15 : Structure d'un RNN déroulé

Dans la figure 15, un bloc de réseau de neurones A, examine une entrée X_i et génère une valeur H_i. Une boucle permet aux informations d'être transmises d'une étape du réseau à la suivante. Un réseau de neurones récurrent peut être considéré comme plusieurs copies du même réseau, chacune transmettant un message à un successeur.

Parmi les types de réseaux de neurones récurrents, nous citons : les réseaux de mémoire à long terme à court terme (LSTM) et leur variante appelée *Gated Recurrent Unit* (GRU).

5.2.1 LSTM

Les réseaux récurrents à mémoire court et long terme (en anglais *Long short-term memory*) forment un type particulier de RNNs, capable d'apprendre des dépendances à long terme, c'est-à-dire se souvenir des informations pendant de longues périodes. Ils ont été introduits par Hochreiter et Schmidhuber (1997) [21] et ont été affinés et popularisés par de nombreuses personnes dans des travaux ultérieurs.

Les LSTMs ont été créés comme solution à la mémoire à court terme des RNNs. En effet, ces derniers peuvent laisser de côté des informations importantes dès le début. Si une séquence est suffisamment longue, ils auront du mal à transmettre les informations des étapes précédentes aux étapes suivantes.

La figure 16 illustre la structure d'un réseau LSTM :

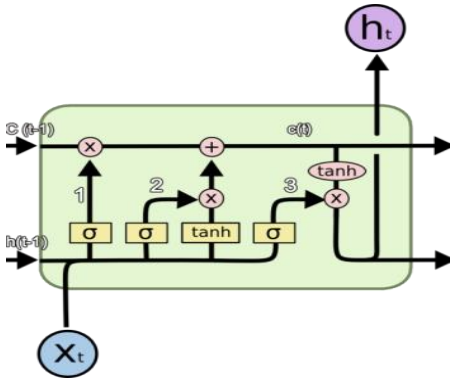

Figure 16 : Structure d'un réseau de neurones LSTM

Les LSTM contiennent des informations qui peuvent être stockées, écrites ou lues dans une cellule, de la même manière que les données stockées dans la mémoire d'un ordinateur. La cellule prend des décisions sur ce qu'il faut stocker et quand autoriser les lectures, écritures et effacements, via des portes s'ouvrant et se fermant. Ces portes agissent sur les signaux qu'elles reçoivent et bloquent ou transmettent des informations.

Les trois principales portes d'un LSTM sont :

1. **Porte d'oublie (*Forget gate*)** : Cette porte décide quelles informations doivent être jetées ou conservées. Les informations de l'état masqué précédent et les informations de l'entrée de la cellule en cours x(t) marquée par un cercle bleu sont transmises à la fonction Sigmoïde, la sortie est multipliée par la sortie de la cellule précédente c(t-1). Cette partie de la cellule détermine si l'entrée précédente sera prise en compte ou non lors de la prochaine itération.
2. **Porte d'entrée (*Input gate*)** : Pour mettre à jour l'état de la cellule, nous avons la porte d'entrée. La somme de l'entrée x(t) et de l'entrée d'état précédemment masquée h(t-1) est transmise à deux fonctions différentes. La fonction sigmoïde décide quelles valeurs seront mises à jour en transformant les valeurs entre 0 et 1. Le 0 signifie pas important, et le 1 signifie important. La fonction tanh quant à elle écrase les valeurs comprises entre -1 et 1 afin de contribuer à la régulation du réseau qui consiste à éviter le surapprentissage et s'assurer du fonctionnement correct du réseau. Le produit de ces deux sorties est additionné à la sortie de la porte d'oublie. La somme indiquée par c(*t*) est l'état actuel de la cellule.

3. **Porte de sortie** : décide quel devrait être le prochain état caché. La somme de l'entrée x(t) et de l'entrée h(t-1) précédemment masquée est transmise à une troisième fonction sigmoïde. La sortie de ceci est multipliée par la tanh de (c(t)), qui forme la sortie d'état cachée de la cellule LSTM en cours.

5.2.2 GRU

GRU (en anglais *Gated recurrent unit*) est une variante d'un LSTM introduite par Kyunghyun Cho [22]. Ainsi, le réseau LSTM a trois portes : entrée, sortie et oublie. Par contre, un réseau GRU ne possède que deux portes, une porte de mise à jour **z** et une porte de réinitialisation **r** :

1. **Porte de mise à jour :** détermine la quantité de mémoire à conserver. Son fonctionnement consiste à multiplier l'entrée x(t) et l'entrée h(t-1) avec leurs poids respectifs. Par la suite, le résultat sera additionné et une fonction d'activation sigmoïde lui sera appliquée.
2. **Porte de réinitialisation :** définit comment combiner une nouvelle entrée avec la valeur précédente. C'est la première à être utilisée, elle fonctionne de la même façon que la porte de mise à jour. En effet, les entrées x(t) et h(t-1) seront multipliées à leurs poids correspondants. Ensuite le résultat sera additionné et une fonction sigmoïde lui sera appliquée. La différence entre les deux portes réside dans les poids et l'utilisation de la porte de réinitialisation.

Fondamentalement, ce sont deux vecteurs qui décident quelles informations doivent être transmises à la sortie. Leurs particularité est qu'ils peuvent être formés pour conserver des informations de longue date, sans les perdre dans le temps ou pour supprimer des informations qui ne sont pas pertinentes pour la prédiction. GRU est considéré comme un réseau de neurones rapide à former et nécessite moins de données à généraliser (qu'un réseau LSTM) en général.

La figure 17 montre la structure d'un réseau de neurones GRU.

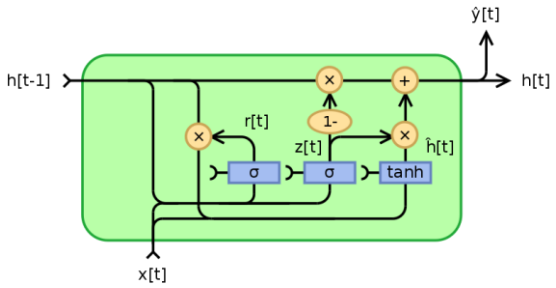

Figure 17 : Structure d'un réseau de neurones GRU

5.3 Réseaux convolutionnels (CNN)

Les réseaux de neurones convolutifs désignés par l'acronyme CNN, de l'anglais *Convolutional Neural Networks* sont un type particulier de réseaux de neurones multicouches dans lesquels le motif de connexion entre les neurones est inspiré par le cortex visuel [2] des animaux. Ce sont des réseaux convolutifs conçus pour nécessiter un prétraitement minimal. Leur nom provient d'une des opérations les plus importantes du réseau à savoir la convolution.

En 1990, le premier réseau de neurone convolutif s'appelait LeNet-5 [23] qui était l'œuvre pionnière de Yann LeCun et permettait de classer les chiffres à partir de nombres écrits à la main.

Une architecture CNN est formée par un empilement de couches de traitement indépendantes organisées comme suit :

(Convolution * m + pooling + correction) * n + couche entièrement connectées * p + couche de sortie.

Les CNNs sont ainsi caractérisés par deux nouvelles couches supplémentaires : une couche de convolution et une couche de pooling.

5.3.1 Couche de convolution

La couche convolutionnelle (en anglais *convolution layer*) est le bloc de construction de base d'un CNN qui traite les données d'un champ récepteur. Le but principal de la convolution est d'extraire des caractéristiques des données en entrées (une image ou un texte par exemple).

[2] Cortex visuel : occupe le lobe occipital du cerveau et est chargé de traiter les informations visuelles.

Les trois hyperparamètres qui permettent de dimensionner le volume de la couche de convolution sont :

1. Le pas (*stride*) : lorsque l'on effectue une convolution, on choisit ce qu'on appelle le pas, c'est-à-dire le pas avec lequel on déplace le noyau à travers le volume d'entrée.
2. La profondeur : correspond au nombre de filtres (noyau de convolution) utilisé pour l'opération de convolution, le nombre de filtres est en fait le nombre de noyaux (ou nombre de neurones associés à un même champ récepteur).
3. La marge à zéro (*zero padding*) : il est commun de mettre des zéros à la frontière du volume d'entrée afin de contrôler la dimension du volume de sortie. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée.

5.3.2 Couche de pooling

La couche de pooling (en anglais *pooling layer*) est une opération de sous-échantillonnage typiquement appliquée après une couche convolutionnelle. Elle permet de compresser l'information en réduisant la dimension, c'est-à-dire la hauteur et la largeur du volume d'entrée sans toutefois affecter sa profondeur. La couche de pooling fonctionne indépendamment sur chaque tranche de profondeur de l'entrée et la redimensionne uniquement au niveau de la surface.

Les types de pooling les plus populaires sont :

1. Le *Max-Pooling* : Chaque opération de pooling sélectionne la valeur maximale de la surface
2. L'*Average-Pooling*: Chaque opération de pooling sélectionne la valeur moyenne de la surface

6 Réseaux de neurones et systèmes de recommandation

6.1 Architectures utilisées

Les réseaux de neurones ont suscités un intérêt considérable dans de nombreux domaines de recherche en raison non seulement de leurs performances exceptionnelles, mais également de leurs capacités à résoudre de nombreuses tâches complexes tout en

fournissant des résultats novateurs. Ils ont récemment révolutionnés les architectures de recommandation de façon spectaculaire et offrent plus de possibilités d'améliorer la qualité des recommandations. Ils sont ainsi capables d'attraper les relations complexes dans les données elles-mêmes, à partir d'abondantes sources de données accessibles telles que des sources contextuelles, textuelles et informations visuelles.

Différentes architectures peuvent être déployées dans les systèmes de recommandation, utilisant les réseaux de neurones LSTM, GRU ou encore CNN. Parmi les réalisations existantes, nous citons :

- Le système de recommandation de vidéos sur Youtube qui est basé sur un réseau neuronal profond. Il présente une architecture à deux niveaux, la recommandation intra-session et la connexion intersessions utilisant des réseaux de neurones récurrents hiérarchiques [24].
- Le système de recommandation d'applications de Google Play avec un modèle large et profond [25].
- Le système de recommandation de nouvelles de Yahoo news basé sur les RNN [26].

6.2 Intégration de l'apprentissage par renforcement

Différentes techniques ont été proposées dans la recommandation telle que les méthodes basées sur le contenu ou basées sur le filtrage collaboratif, la régression logistique ou encore l'apprentissage en profondeur. Cependant, ces techniques considèrent la procédure de recommandation comme un processus statique, c'est-à-dire qu'ils supposent que la préférence reste inchangée.

L'apprentissage par renforcement a ainsi été introduit afin de palier à ce problème. Des méthodes sont proposées pour modéliser la procédure de recommandation, telle que *Q-learning* [27] qui est un algorithme qui cherche à trouver la meilleure action à prendre compte tenu de l'état actuel. Plus spécifiquement, il cherche à apprendre une politique qui maximise la récompense totale. Le «q» dans *q-learning* est synonyme de qualité. La qualité dans ce cas, montre à quel point une action donnée est utile pour obtenir une récompense future.

7 Conclusion

Dans ce chapitre, nous avons présenté des généralités sur l'apprentissage automatique et sur les réseaux de neurones. Nous avons commencé par présenter les types d'apprentissage automatique. Puis, nous avons introduit la notion de réseau de neurones en détaillant ses différentes propriétés. Après quoi, nous avons présenté les architectures des réseaux de neurones où nous avons distingué les réseaux non bouclés, les réseaux récurrents et les réseaux convolutionnels. Pour finir, nous nous sommes intéressés à l'utilisation des réseaux de neurones dans les systèmes de recommandation.

Le prochain chapitre s'articule sur la conception d'une architecture de réseaux de neurones profonds (à plusieurs couches cachées) permettant d'effectuer des recommandations basées sur du contenu textuel, en l'occurrence des descriptions et en tenant compte des préférences d'un utilisateur.

Chapitre 3 : Modèles proposés

1 Introduction

Le domaine de la recommandation est une tâche qui consiste à prédire l'intérêt qu'aura un utilisateur pour un *item* (objet) donné, dans notre cas l'*item* représente un film.

Le problème de cette recommandation peut être traduit par une classification d'un ensemble de films selon leurs attributs descriptifs. En effet, le système sera capable de regrouper des films similaires en termes de scores en prenant en entrées leurs descriptions afin de suggérer à l'utilisateur des films de la même catégorie (donc ayant un score élevé).

Dans ce chapitre, nous présentons les méthodes de représentation et codage des données textuelles d'entrée. Par la suite, nous détaillons les trois modèles d'architectures réalisés pour une recommandation de films basée sur le contenu.

2 Apprentissage automatique et traitement de texte

Avant de développer des modèles d'apprentissage automatique pour nos besoins, il faut tout d'abord représenter nos données sous un format plus facile à manipuler.

Dans le cas de notre recommandation, nos données d'entrée sont les descriptions de chaque film. Elles sont représentées par du texte qui n'est pas exploitable tel quel dans un réseau de neurones. Il faut donc le convertir en un format numérique. Pour cela, deux étapes sont effectuées.

Tout d'abord, un prétraitement est effectué sur le texte pour éliminer l'information considérée comme peu importante lors de la recommandation. Cette étape permet de réduire la taille du texte et donc de faciliter l'entrainement d'un réseau de neurones sur ces données. Ensuite, la deuxième étape consiste à représenter le texte par une suite de nombres. Nous détaillons ces étapes dans ce qui suit.

2.1 Prétraitement du texte

Le prétraitement consiste à effectuer des transformations sur les données d'entrée textuelles afin que ces données soient compréhensibles par nos modèles de réseaux de neurones. Ces entrées sont en fait des vecteurs ou des matrices qui définissent le poids de chaque descripteur (mots ou groupe de mots) dans le texte où ils apparaissent. Le prétraitement du texte comprend plusieurs étapes.

2.1.1 Récupération du corpus de texte

La première étape est la récupération du texte. Il existe plusieurs manières de récupérer du texte : soit depuis une base de données disponible, soit depuis des fichiers XML ou autres.

2.1.2 Exploration du texte

Après avoir récupéré le corpus de texte, vient l'étape de l'exploration. Celle-ci est réalisée à l'aide de plusieurs méthodes.

2.1.2.1 Segmentation (Tokenization)

Dans le cadre du traitement automatique du langage, un mot peut être représenté par des caractères comme ".", ",", "l'", etc. Le texte dans son format brut peut être vu comme une simple séquence de caractères. La segmentation consiste à identifier les mots de cette séquence pour la convertir ensuite en une liste de mots.

2.1.2.2 Elimination des mots vides (stop words)

Les mots vides (en anglais *stop words*) sont les mots qui n'ont pas de sens significatif et qui souvent sont les plus communs dans des textes, par exemple "le", "la" ou "des" en français. Ils doivent ainsi être supprimés de la représentation des textes. Cette étape sert à nettoyer le corpus.

2.1.2.3 Lemmatisation et Racinisation

La racinisation permet de considérer que la racine des mots plutôt que les mots en entier sans pour autant se soucier de l'analyse grammaticale. Quant à La lemmatisation, c'est tout simplement une forme de racinisation qui permet de ramener les termes à leurs formes canoniques en se basant sur l'analyse grammaticale. L'idée étant de supprimer les suffixes, préfixes et autres des mots afin de ne conserver que leur origine. Cette technique permet de réduire la dimension des descripteurs et ainsi fournir une meilleure représentation.

2.1.2.4 Création d'un vocabulaire

Il s'agit de créer une liste de tous les mots que l'on désire garder pour représenter un texte. Le vocabulaire sert à donner un identifiant (ID) unique à chacun de ses mots, correspondant généralement à l'index dans la liste. On obtient par la suite une liste de chiffres pour représenter chaque exemple d'un corpus.

2.2 Représentations de texte

La représentation de texte consiste à interpréter les textes et les représenter sous une forme exploitable par les modèles de réseaux de neurones. Nous allons présenter dans ce qui suit quelques méthodes utilisées pour la représentation de texte.

2.2.1 Représentation en sac de mots (*bag of words*)

Cette représentation qui est appelée sac de mots ou en anglais *bag of word* consiste à représenter un texte dans l'espace des mots par un vecteur dont chaque composante correspond au nombre d'apparition d'un mot dans le document.

2.2.2 Représentation par des phrases

Cette technique propose d'utiliser les phrases comme unité de représentation. En effet, les phrases sont plus informatives que les mots seuls. Ce type de représentation permet d'obtenir de meilleurs résultats que ceux obtenus via la représentation en sac de mots en raison de la richesse sémantique de la phrase.

2.2.3 Représentation par des racines lexicales

La racinisation (*stemming* en anglais) permet de considérer que la racine des mots plutôt que les mots en entier. C'est une représentation bien meilleure que le codage de type « sac de mots » et permet de réduire la dimensionnalité des descripteurs.

2.2.4 Représentation avec des lemmes

La lemmatisation décrite auparavant, consiste à utiliser l'analyse grammaticale afin de remplacer les verbes par leur forme infinitive et les noms par leur forme au singulier. La substitution des mots par leur lemme réduit également l'espace des descripteurs comme pour les racines, et permet de représenter par un même descripteur des termes de même signification. Cette représentation est dites simple mais peut causer une perte d'informations.

2.3 Prétraitement et représentation utilisés

Dans le cadre de notre travail, nous avons effectué un prétraitement et une représentation de nos données textuelles, afin de les rendre utilisables comme entrée dans notre réseau de neurones. Nous décrivons dans ce qui suit les étapes par lesquelles nous sommes passées.

Tout d'abord, une récupération de nos données a été faite depuis une base de données préexistante. Par la suite, nous avons effectué les étapes d'exploration de texte suivantes : la segmentation, l'élimination des mots vides et la création d'un vocabulaire.

Enfin, chaque description est représentée par un vecteur numérique de taille égale, où chaque chiffre correspond à l'identifiant du mot dans le vocabulaire que nous avons créé.

3 Motivations et objectifs

L'objectif étant de réaliser un système de recommandation, nous proposons d'utiliser du contenu textuel. L'idée est de pouvoir exploiter au mieux des descriptions de films, sans prendre en considérations les autres attributs qui sont susceptibles de décrire un film donné.

Pour ce faire, nous avons opté pour les réseaux de neurones qui ont montrés leurs efficacités dans de nombreux domaines d'applications tels que le traitement d'image, la catégorisation de textes et le diagnostique médicale et cela grâce à leur capacité de classification et de généralisation.

Dans notre cas, nos modèles de réseaux de neurones déployés, en s'appuyant sur différentes architectures, sont utilisés comme méthode de classification de films selon un score de pertinence (un score définissant la probabilité que le film soit aimé par l'utilisateur). Par la suite, nous effectuons une recommandation en se basant sur les modèles de classification ainsi réalisés.

4 Architectures des modèles proposés

Au cours de nos expérimentations, nous avons créé trois modèles (modèle 1, modèle 2 et modèle 3) avec différentes architectures. Ils représentent des modèles de classifications. Ils prennent en entrée les descriptions de films prétraitées et produisent en sortie onze classes. Chaque classe correspond à un score de 0 à 10 et regroupe un ensemble de films ayant un score similaire.

Nous présentons dans ce qui suit ces trois modèles.

4.1 Architecture du modèle 1

Le premier modèle que nous présentons dans la figure 18 est une architecture de réseau convolutionnel (CNN). Elle est composée de deux couches de convolution, deux couches de *maxpooling* et cinq couches entièrement connectées.

La première couche de notre réseau est une couche d'incorporation qui code le texte en entrée qui représente une description de taille 30, en une séquence de vecteurs denses de dimension 300. Nous avons fixé la taille des descriptions à 30. Celle-ci représente la taille de la plus longue description dans notre jeu de données.

La deuxième couche est la couche de convolution avec 300 neurones, utilisant une fenêtre de convolution de taille 5 et une fonction d'activation *Leaky ReLu*. Le résultat est ensuite passé à une couche de *maxpooling* afin de compresser l'information en réduisant la taille de notre entrée. Elle utilise 300 neurones et une fenêtre de *pooling* de taille 2.

Nous réitérons la même chose avec une couche de convolution et une couche de *maxpooling* qui seront composées de 150 neurones.

Après cela, nous avons un réseau de neurones composé de cinq couches entièrement connectées. Les quatre premières couches ont respectivement 1000 neurones, 300 neurones, 150 neurones et 50 neurones où la fonction d'activation utilisée est Relu. La dernière couche utilise la fonction d'activation softmax qui permet de calculer la distribution de probabilité des 11 classes. Le nombre de neurones est décroissant, cela permet d'avoir un passage progressif d'un nombre élevé de neurones en entrée, vers 11 neurones en sortie, pour faciliter l'entrainement et sans pour autant rendre le réseau trop complexe.

La figure 18 résume l'architecture du modèle 1.

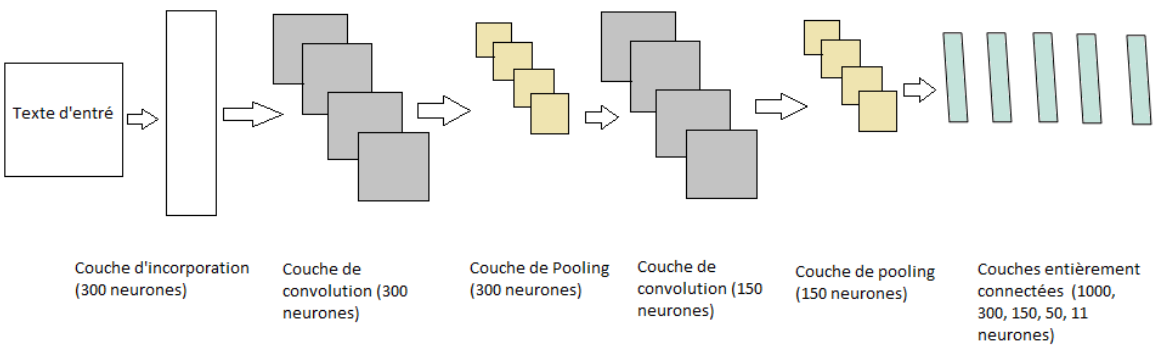

Figure 18 : Architecture du modèle 1

4.2 Architecture du modèle 2

Le deuxième modèle que nous présentons dans la figure 19 est une architecture simple d'un réseau LSTM.

La première couche de notre réseau est une couche d'incorporation qui code le texte en entrée qui représente une description de taille 30 en une séquence de vecteurs denses de dimension 300.

La couche suivante est une couche LSTM qui transforme notre entrée en un seul vecteur contenant des informations sur la séquence entière. Elle utilise 100 neurones.

Le résultat est ensuite passé à une couche de sortie utilisant 11 neurones où la fonction d'activation est softmax.

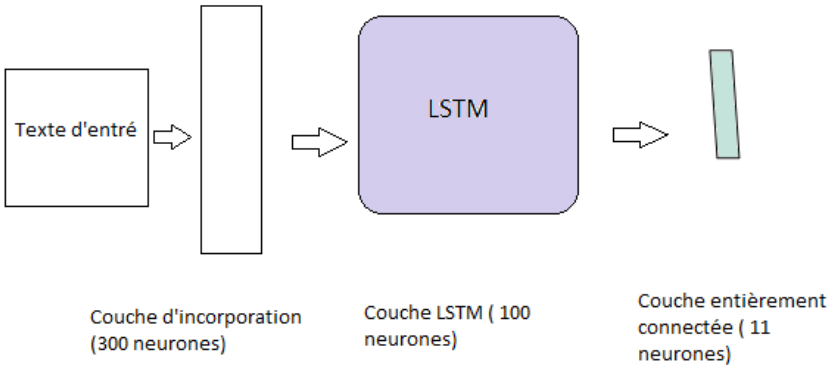

Figure 19 : Architecture du modèle 2

4.3 Architecture du modèle 3

Le troisième modèle que nous présentons dans la figure 20 est une architecture qui combine un réseau LSTM et un réseau CNN.

La première couche de notre réseau est une couche d'incorporation qui code le texte en entrée qui représente une description de taille 30 en une séquence de vecteurs denses de dimension 300.

Les autres couches sont les couches d'un CNN à savoir une couche de convolution et une couche de maxpooling. Elles sont utilisées comme moyen d'accélérer le temps d'entrainement de notre réseau. La couche de convolution avec 300 neurones, utilisant une fenêtre de convolution de taille 3 et une fonction d'activation Leaky ReLu. Le résultat est

ensuite passé à une couche de maxpooling afin de compresser l'information en réduisant la taille de notre texte d'entrée. Elle utilise 300 neurones et une fenêtre de pooling de taille 2.

La couche suivante est la couche LSTM avec 100 neurones. Et pour finir, la couche de sortie utilisant la fonction d'activation softmax, qui aura 11 neurones (classes) en sortie (figure 20).

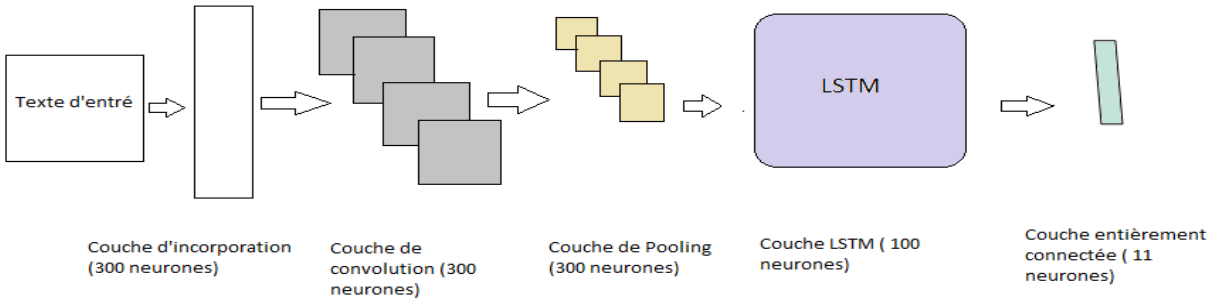

Figure 20 : Architecture du modèle 3

5 Mécanisme de recommandation

Les systèmes de recommandation sont une bonne alternative au système de recherche simple. En effet, un système de recommandation est destiné à proposer à un utilisateur des *items* (objet) susceptibles de l'intéresser. Dans notre cas, afin de suggérer à un utilisateur des films qui sont susceptibles de lui plaire, nous procédons d'abord à une classification de films selon leurs scores attribués et cela en utilisant les modèles proposés. En effet, La méthode de classification par réseaux de neurones a pour but d'identifier les classes auxquelles appartiennent des objets à partir de certains paramètres descriptifs. Il s'agit d'extraire une règle générale à partir des données observées (données d'entrainement).

Pour effectuer la recommandation, notre système récupère les scores que l'utilisateur attribuera aux films qu'ils lui seront proposés. Ensuite, notre réseau de classification s'entrainera sur ces données observées (descriptions et score), pour générer une règle qui devra classifier l'ensemble des films de notre base de données. En d'autres termes, le système va pouvoir classifier correctement les nouvelles descriptions et prédire le score de chaque film. Enfin, des films ayant des scores élevés seront suggérés à l'utilisateur.

6 Conclusion

Dans ce chapitre, nous avons présenté les prétraitements nécessaires au développement de modèles d'apprentissage. Par la suite, nous avons décrit les architectures que nous avons réalisées et utilisées dans le cadre de la recommandation de films tout en démontrant leurs objectifs.

Le prochain chapitre décrit notre implémentation, les expérimentations réalisées sur les modèles précédents ainsi que les interfaces de notre système de recommandation.

Chapitre 4 : Implémentation et expérimentations

1 Introduction

Dans ce chapitre, nous présentons les outils logiciels ainsi que les différentes librairies que nous avons utilisées. Parmi elles Tensorflow et Keras qui sont les deux principales bibliothèques déployées pour la réalisation et l'apprentissage de nos modèles.

Nous détaillons le jeu de données que nous avons utilisé afin d'entrainer, évaluer et tester nos modèles. Par la suite, nous montrons les résultats obtenus pour chaque architecture proposée et effectuons une comparaison entre elles afin de définir la mieux adaptée à notre système de recommandation. Enfin, nous présentons l'interface que nous avons réalisée pour la suggestion de films.

2 Outils et environnement de développement

2.1 Logiciels

2.1.1 Pycharm

Pycharm [1] est un environnement de développement intégré IDE (*Integrated Development Environment*). C'est un logiciel permettant d'intégrer dans une même fenêtre tous les éléments utiles à la programmation en python : un éditeur de texte pour écrire des scripts, une console pour exécuter des programmes, ainsi qu'un explorateur de fichiers pour parcourir le projet en cours. Il est développé par la société *JetBrains* qui est une entreprise informatique éditant des logiciels pour les développeurs de logiciels.

Pycharm permet l'analyse de code et contient un déboguer (*debugger*) graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement Web avec Django (un *framework* Web Python).

2.2 Bibliothèques logicielles

2.2.1 Tensorflow

Tensorflow[2] est un *framework* de programmation pour le calcul numérique qui a été rendu *Open Source* par Google en Novembre 2015. C'est l'un des *framework* les plus utilisés pour l'apprentissage profond et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont

[1] https://www.jetbrains.com/pycharm/
[2] https://www.tensorflow.org/

principalement faites via des tables de données multidimensionnelles, appelées Tenseurs (*Tensor*). Un tenseur à deux dimensions est l'équivalent d'une matrice.

2.2.2 Keras

Keras[3] est une API de réseaux de neurones de haut niveau, écrite en Python et capables de fonctionner sur Tensorflow. Elle présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilitent la configuration des réseaux neuronaux indépendamment de la bibliothèque informatique de *backend*.

2.2.3 Pandas

Pandas[4] est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Pandas est un logiciel libre sous licence BSD, ce qui permet sa réutilisation avec peu de restrictions.

2.2.4 Numpy

Numpy[5] est la librairie fondamentale du calcul scientifique avec Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux (*array*).Les calculs avec Numpy sont particulièrement optimisés car les tableaux sont homogènes (ils ne contiennent que des valeurs d'un même type) et de taille fixée à la création.

2.2.5 Tensorboard

Tensorboard fournit une suite d'outils de visualisation facilitant la compréhension, le débogage et l'optimisation de programmes. Il est utilisé pour visualiser les graphiques Tensorflow, tracer des métriques quantitatives sur leur exécution et afficher des données supplémentaires, telles que des images qui les traversent.

2.2.6 Flask

Flask[6] est un *framework open-source* de développement Web en Python. Il est conçu pour permettre une mise en route rapide et facile, avec la possibilité d'évoluer vers des

[3] https://keras.io/
[4] https://pandas.pydata.org/
[5] https://www.numpy.org/
[6] https://flask.palletsprojects.com

applications complexes. Il offre des suggestions et la possibilité au développeur de choisir les outils et les bibliothèques qu'il souhaite utiliser.

2.3 Configuration matérielle utilisée

La configuration du matériel utilisé dans notre implémentation est comme suit :

- Un PC portable Dell i5 CPU 2.20 GHZ ;
- Carte graphique Nvidia GeForce 820M ;
- RAM de taille 4 GO ;
- Disque dur de taille 500 GO ;
- Système d'exploitation Windows 7 professionnel.

3 Jeux de données utilisés

Le jeu de données que nous avons utilisé est une collection préexistante extraite de l'*Internet Movie Database* (littéralement « Base de données cinématographiques d'Internet »), abrégé en IMDb[7], qui représente une base de données en ligne sur le cinéma mondial, sur la télévision, et plus secondairement les jeux vidéo. IMDb restitue un grand nombre d'informations concernant les films, les acteurs, les réalisateurs, les scénaristes et toutes personnes et entreprises intervenant dans l'élaboration d'un film, d'un téléfilm, d'une série télévisée ou d'un jeu vidéo.

Dans notre cas, nous avons utilisé une collection sous format CSV (*Comma-separated values*) qui est un format texte ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules. Elle contient des informations sur dix mille (10 000) films représentées sous formes de colonnes à savoir :

- Le rang du film (rank);
- Le titre du film *(title)*;
- L'année de sortie du film (*year) ;*
- Le score qui représente la note attribuée au film ;
- Le genre du film (*Drama*, *Romance*, etc) ;
- Le nom du directeur du film (*director)*;
- La durée du film (*runtime)*;

[7] https://www.imdb.com/

- La description qui représente le résumé (synopsis) du film.

Dans le cadre de nos travaux, nous nous sommes intéressés au score de chaque film ainsi qu'à sa description. Nous avons ainsi, divisé notre jeu de données en trois parties :

- **Données d'entrainement** : qui constituent 80% de notre jeu de données à savoir 8000 films. Ces données sont utilisées pour générer le modèle d'apprentissage.
- **Données de validation** : qui constituent 10% de notre jeu de données à savoir 1000 films. Ces données seront utilisées lors de l'apprentissage afin de valider le modèle.
- **Données de test** : qui constituent les 10% restantes de notre jeu de données à savoir 1000 films. Le modèle d'apprentissage sera alors appliqué à ces données à des fins de tests.

4 Mesures d'évaluation

Lorsqu'un système de recommandation est développé, il est important d'être en mesure d'évaluer son fonctionnement et sa capacité à répondre aux objectifs qui lui ont été fixés. Pour ce faire, nous avons utilisé deux mesures d'évaluation : la précision et la perte.

4.1 Précision (*accuracy*)

La précision est la mesure la plus utilisée pour juger et évaluer la performance d'un modèle. Elle est définie par la formule suivante :

$$\text{Accuracy} = \frac{\text{Nombre de Prédictions correctes}}{\text{Nombre total de prédictions faites}}$$

4.2 Perte (*loss*)

La perte indique dans quelle mesure un modèle donné se comporte bien après chaque itération d'optimisation. Idéalement, il doit y avoir une réduction des pertes après chaque ou plusieurs itérations.

5 Entrainement des modèles et résultats obtenus

Nous avons entrainé nos trois modèles décrits dans la section précédente sur l'ensemble des données d'entrainement. Par la suite, nous avons testé les trois modèles sur l'ensemble de données de test.

Nous montrons dans ce qui suit les résultats obtenus en effectuant une évaluation en termes de précision et de perte.

5.1 Résultats obtenus pour le modèle 1

Nous présentons ci-dessous les courbes de précision et de perte montrant l'entrainement et la validation du premier modèle (figure 21 et figure 22).

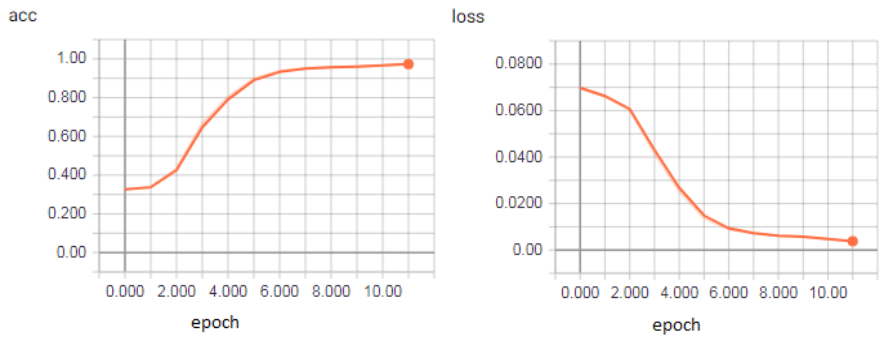

Figure 21 : Précision et perte du modèle 1 sur les données d'entraînement

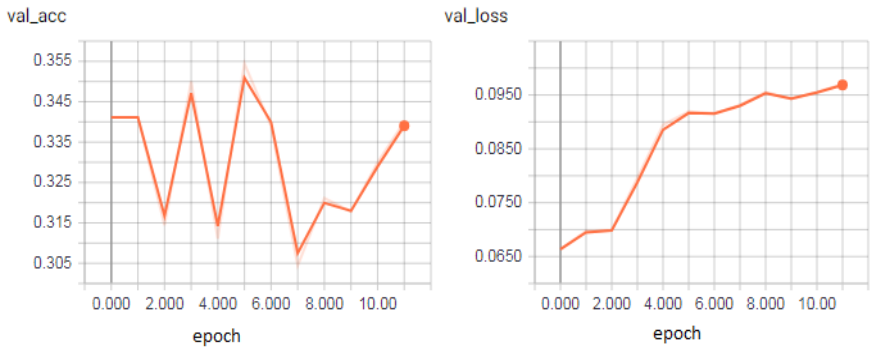

Figure 22 : Précision et perte du modèle 1 sur les données de validation

D'après la figure 21, la précision (acc) lors de la phase d'apprentissage augmente avec le nombre d'époques. Ceci reflète la capacité du modèle à apprendre. En effet, plus le nombre d'époques augmente, plus le modèle apprend et plus l'erreur d'apprentissage diminue.

D'après la figure 22, on remarque que la précision sur l'ensemble des données de validation augmente et diminue au fil des itérations jusqu'à atteindre 34% à l'époque 11.

Le tableau 1 montre les résultats obtenus pour le modèle 1.

Précision finale sur l'ensemble d'entrainement	Précision finale sur l'ensemble de test
97,30%	33,40%

Tableau 1 : Tableau de la précision finale sur l'ensemble d'entrainement et de test obtenue pour le modèle 1

5.2 Résultats obtenus pour le modèle 2

Les courbes ci-dessous (figure 23 et figure 24) représentent la précision et la perte sur l'ensemble des données d'entrainement et de validation avec le deuxième modèle.

Figure 23 : précision et perte du modèle 2 sur les données d'entrainement

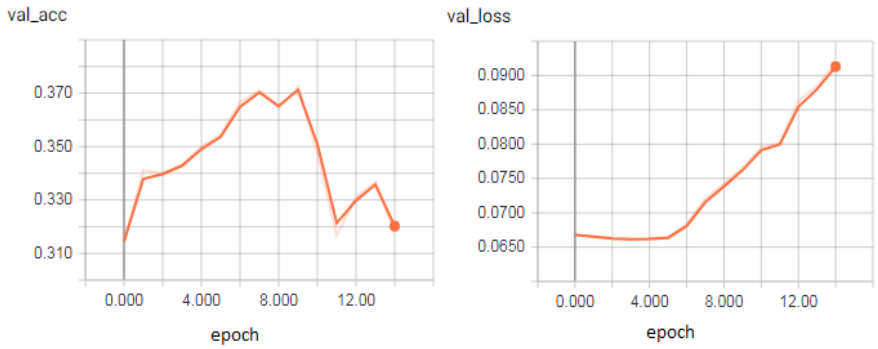

Figure 24 : Précision et perte du modèle 2 sur les données de validation

D'après la figure 23, nous pouvons remarquer que la précision lors de l'apprentissage augmente avec le nombre d'époques. Contrairement à l'erreur d'apprentissage qui elle diminue.

Concernant la validation (figure 24), nous constatons que la précision augmente jusqu'à l'époque 9 pour ensuite diminuer jusqu'à 32% à partir de l'époque 11. Quant à la perte, celle-ci augmente avec le nombre d'époques.

Le tableau 2 illustre les résultats obtenus pour le modèle 2.

Précision finale sur l'ensemble d'entrainement	Précision finale sur l'ensemble de test
81,20%	34,30%

Tableau 2 : Tableau de la précision finale sur l'ensemble d'entrainement et de test obtenue pour le modèle 2

5.3 Résultats obtenus pour le modèle 3

Nous présentons dans ce qui suit les courbes (figure 25 et figure 26) de précision et de perte montrant l'entrainement et la validation avec le troisième modèle.

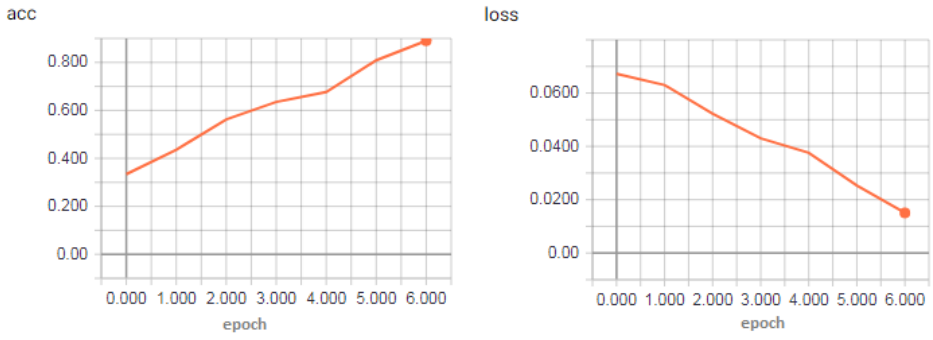

Figure 25 : précision et perte du modèle 3 sur les données d'entrainement

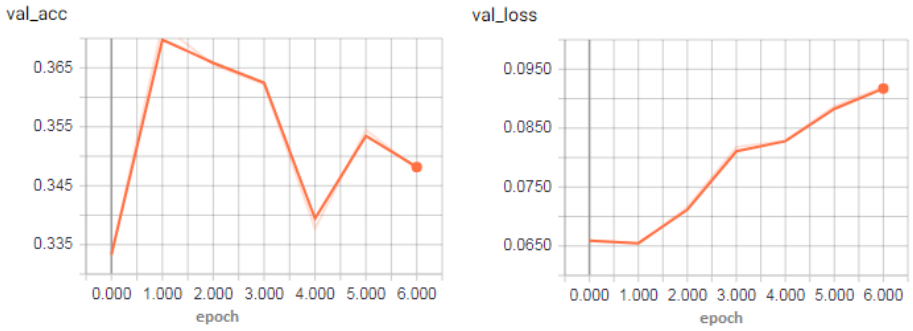

Figure 26 : précision et perte du modèle 3 sur les données de validation

D'après la figure 25, nous pouvons remarquer qu'à chaque époque la précision augmente lors de l'apprentissage et l'erreur quant à elle diminue. Ceci reflète que le modèle apprend plus d'informations.

Lors de la validation, nous constatons d'après la figure 26, que la précision atteint la valeur 34,9 % à l'époque 6. La perte quant à elle augmente avec le nombre d'époque.

Le tableau 3 montre les résultats obtenus pour le modèle 3.

Précision finale sur l'ensemble d'entrainement	Précision finale sur l'ensemble de test
85,26%	35,2%

Tableau 3 : Tableau de la précision finale sur l'ensemble d'entrainement et de test obtenue pour le modèle 3

5.4 Discussion

Les résultats obtenus pour les trois modèles sont comparés dans le tableau 4 ci-dessous.

	Nombre d'époques	Précision finale sur l'ensemble d'entrainement	Précision finale sur l'ensemble de test	Temps d''exécution
Modèle 1	11	97,30%	33,40%	10 min 36
Modèle 2	13	81,20%	34,30%	6 min 02
Modèle 3	7	85,26%	35,20%	5 min 24

Tableau 4 : Tableau comparatif des résultats obtenus par les trois modèles

D'une manière générale, les résultats de nos trois modèles sont similaires. Cependant, nous pouvons dire que le troisième modèle a présenté de meilleurs résultats en prenant en compte tout les facteurs comparatifs. Ce qui signifie que l'ajout de couches de convolution et de pooling au réseau de neurones LSTM permet une certaine amélioration.

Ces résultats obtenus peuvent être optimisés à mesure que nous approfondissons notre réseau et augmentons le nombre d'époques. La base d'apprentissage est également un élément déterminant dans les réseaux de neurones, il faut avoir une base d'apprentissage de grande taille pour aboutir à de meilleurs résultats.

6 Mise en œuvre de la recommandation

Dans cette section, nous présentons l'interface que nous avons réalisée afin de recommander à un utilisateur des films susceptibles de l'intéresser. Nous illustrons dans ce qui suit le fonctionnement de notre système de recommandation.

6.1 Page d'accueil

En premier lieu, nous présentons à un utilisateur dix films auxquels il devra attribuer des notes (scores). La figure 27 ci-dessous illustre la page d'accueil.

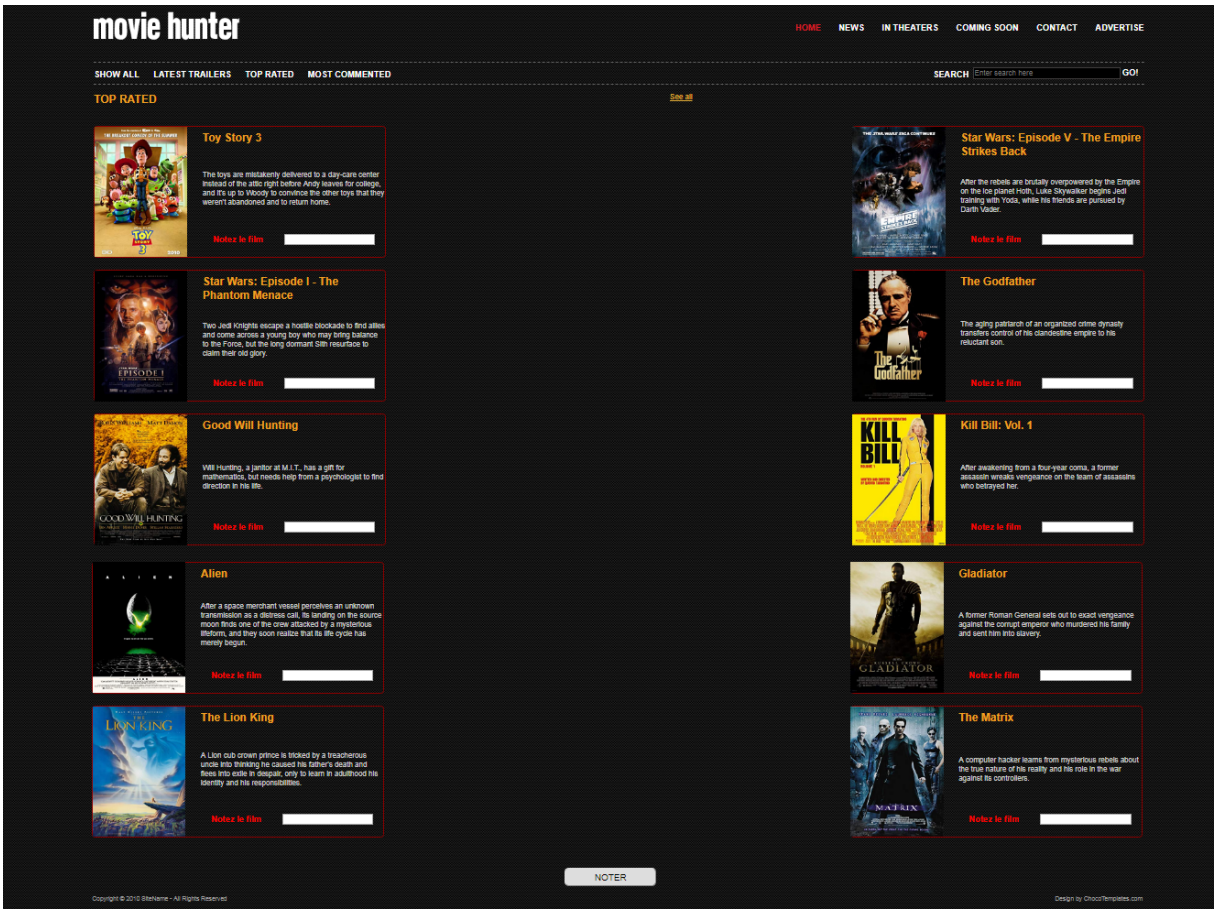

Figure 27 : Page d'accueil

6.2 Page de recommandation

Une fois les films notés, le système suggère à l'utilisateur une nouvelle liste contenant dix nouveaux films ayant le score le plus élevé. La figure 28 illustre la page de recommandation.

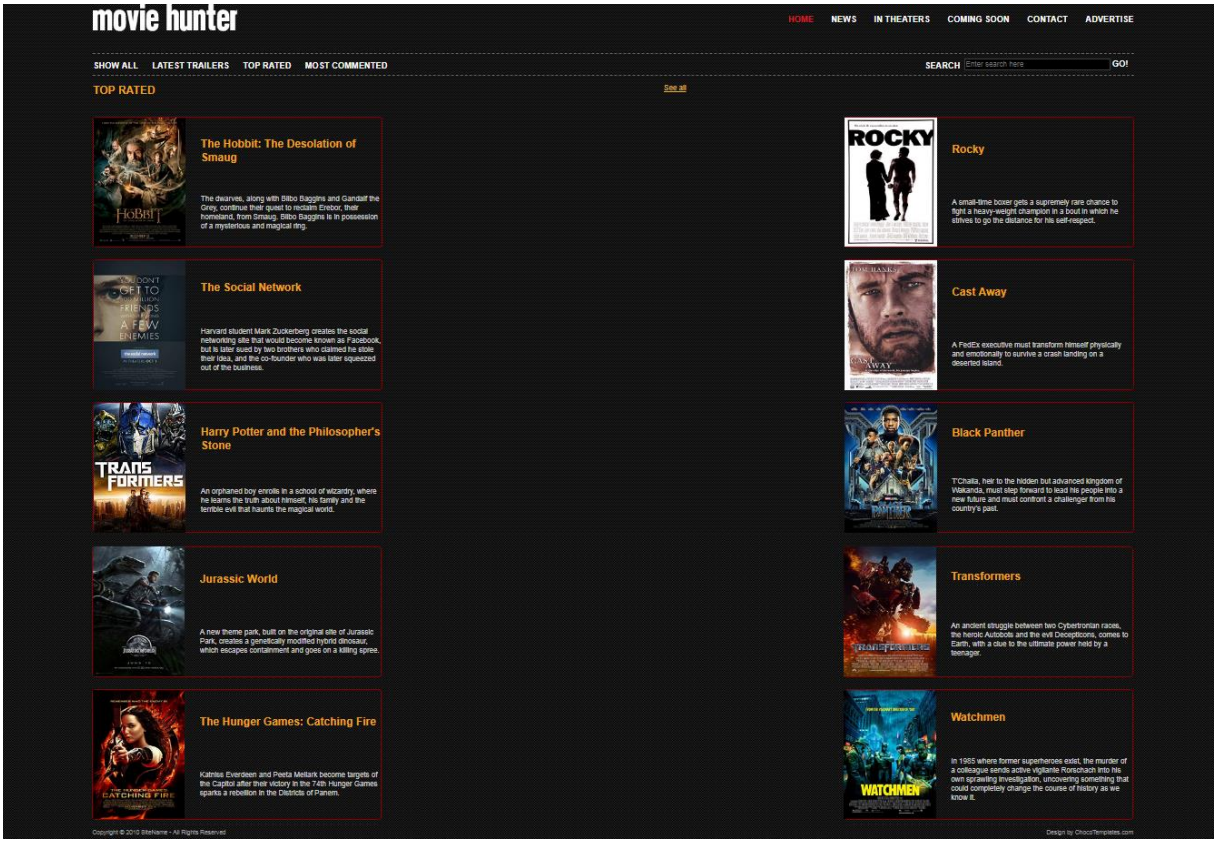

Figure 28 : Page de recommandation

7 Conclusion

Nous avons présenté dans ce chapitre les logiciels, librairies et configuration utilisés pour notre implémentation ainsi que les ensembles d'apprentissage, de validation et de test que nous avons utilisés. Par la suite, nous avons montré l'évaluation des modèles proposés. Pour finir, nous avons présenté l'interface de notre système de recommandation.

La comparaison des résultats trouvés a montré que le nombre d'époques, la taille de la base d'apprentissage et la profondeur des réseaux, sont des facteurs importants pour l'obtention de meilleurs résultats.

Les approches que nous avons proposées doivent être améliorées pour une utilisation future.

Conclusion générale

1 Synthèse

Les systèmes de recommandation visent à apporter des réponses pertinentes aux besoins des utilisateurs. En effet, les usagers ont des attentes différentes vis-à-vis de l'information. Pour répondre au mieux à cette pluralité d'intérêts, nos travaux se sont ainsi orientés vers la prise en compte de la diversité au sein des systèmes de recommandation tout en utilisant une approche basée sur le contenu. Il s'agit de proposer des *items* variés plutôt que similaires.

Dans ce travail, nous avons présenté d'une façon globale les systèmes de recommandation et l'apprentissage automatique, plus particulièrement les réseaux de neurones. Nous avons également présenté l'approche de notre système de recommandation appuyé par trois modèles d'architecture que nous avons évalués ainsi que la réalisation d'une interface afin de visualiser le processus de recommandation.

Les résultats obtenus démontrent un intérêt certain pour l'approche que nous avons adopté pour réaliser notre système de recommandation mais afin d'atteindre un niveau de recommandation plus satisfaisant, celle-ci nécessite une éventuelle amélioration.

2 Perspectives

Les perspectives découlant de nos travaux sont nombreuses.

- La première perspective serait d'intégrer l'apprentissage par renforcement aux modèles déjà réalisés en utilisant le Q-Learning.
- La deuxième perspective serait d'ajouter d'autres entrées en plus des descriptions à nos modèles afin d'améliorer au maximum la recommandation.
- Une autre perspective consiste à améliorer l'interface de notre système de recommandation.

Bibliographie

- [1] **Rich, E. (1979).**User modeling via stereotypes. Cognitive science, 3(4):329–354.
- [2] **Malone, T. W., Grant, K. R., Turbak, F. A., Brobst, S. A. et Cohen, M. D. (1987).** Intelligent information-sharing systems. In Communications of the ACM, 30. Pages 390–402.
- [3] **Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992).** Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12):61–70.
- [4] **Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994).** Grouplens: an open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM conference on Computer supported cooperative work, pages 175–186. ACM.
- [5] **Shardanand, U. and Maes, P. (1995a).** Social information filtering: algorithms for automating Word of mouth In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 210– 217. ACM Press/Addison-Wesley Publishing Co.
- [6] **Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995).** Recommending and evaluating choices in a virtual community of use. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 194–201. ACM Press/Addison-Wesley Publishing Co.
- [8] **Burke, R. (2002).** Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction.
- [9] **Pazzani, M., J. (1999).** A Framework for Collaborative, Content-Based and Demographic Filtering. Artif. Intell. Rev., 13(5-6), 393-408.
- [10] **Stolze, M. and Rjaibi, W. (2001).** Towards scalable scoring for preference-based item recommendation. IEEE Data Eng. Bull., 24(3):42–49.
- [12] **Rocchio, J. J.1971.** Relevance feedback in information retrieval. In Salton (1971b), pp. 313-323.
- [13] **Paul Covington, Jay Adams, and Emre Sargin.** (**2016)** Deep neural networks for YouTube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, RecSys ’16, pages 191–198, New York, NY, USA. ACM.

- [14] **Mokhtar TAFFAR**. Initiation a l'apprentissage automatique. Université de Jijel, Support de Cours pour étudiants en Master en Intelligence Artificielle
- [15] **François Meunier, Christophe Marsala, Laurent Castanié.** Apprentissage par transfert pour la réutilisation de connaissances en classification d'objets 3D. Conférence Nationale sur les Applications Pratiques de l'Intelligence Artificielle, Jul 2017, Caen, France.
- [16] **Davalo et P. Naïm. (1989)**. « Des réseaux de neurones », Edition Eyrolle.
- [17] **Maureen Caudill. (Feb. 1989).** "Neural Network Primer: Part I".
- [18] **John Duchi, Elad Hazan, Yoram Singer, 2011**, Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.
- [19] **Diederik P. Kingma, Jimmy Ba, 2015**, Adam: a method for stochastic optimization.
- [20] **Abdessalem Chamekh, 2008**. Optimisation des procédés de mise en forme par les réseaux de neurones artificiels. Université d'Angers.
- [21] **F.A. Gers, J. Schmidhuber, F. Cummins, 1999**. Learning to forget: continual prediction with LSTM, 9th International Conference on Artificial Neural Networks: ICANN '99, 1999 p. 850 – 855.
- [22] **Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bougares, Fethi, Schwenk, Holger, and Ben-gio, Yoshua, 2014**. Learning phrase representations using rnn encoder-decoder for statistical machine translation. ArXiv preprint arXiv: 1406.1078.
- [23] **Yann LeCun, 1998.** LeNet-5 article de LeCun et al.
- [24] **Paul Covington, Jay Adams, and Emre Sargin, 2016.** Deep neural networks for Youtube recommendations. In Recsys. 191–198.
- [25] **Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others, 2016.** Wide & deep learning for recommender systems. In Recsys. 7–10.
- [26] **Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima, 2017.** Embedding-based News Recommendation for Millions of Users. In SIGKDD.

- [27] **N. Taghipour and A. A. Kardan, 2008**. “A hybrid web recommender system based on q-learning,” in Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, March 16-20, 2008, pp. 1164–1168.

Références hypertextes

- [7] **Samaidhu (January 24, 2015).** Recommendation engine. Récupéré le 02 fevrier 2019 sur https://dataaspirant.com/2015/01/24/recommendation-engine-part-1/
- [11] **Mathieu, 25.04.2012.** Dans Dossiers de Mathieu, Les algorithmes de recommandation. Récupéré le 10 mars 2019 sur http://www.podcastscience.fm/dossiers/2012/04/25/les-algorithmes-de-recommandation/