Exploration des systèmes de recommandation

Pierrick Dossin et **Guillaume Riu**
Département de génie informatique, Polytechnique Montréal, Canada
pierrick.dossin@polymtl.ca, guillaume.riu@polymtl.ca

Résumé

Les systèmes de recommandation, qui fournissent aux utilisateurs des recommandations de contenu adapté à leurs besoins, ont fait l'objet d'une grande attention dans le monde des affaires en ligne. Dans cet article, nous allons explorer les différentes approches de systèmes de recommandation en passant par les méthodes traditionnelles comme les approches collaboratives, puis par les approches contenu et agglomératives, enfin par des techniques de l'état de l'art d'apprentissage profond. Nous comparerons les performances de ces différentes méthodes grâce à la base de données MovieLens en calculant certaines métriques (MSE, MAE) et facteurs de succès (diversité, couverture, sérendipité...) afin de comprendre comment se comportent chacune des méthodes.

1 Introduction

Avec l'émergence du web, les systèmes de recommandation sont devenus indispensables pour filtrer les informations susceptibles d'intéresser les utilisateurs parmi des quantités énormes d'éléments. L'un des défis de ce domaine est d'arriver à prédire le plus précisément possible les préférences des utilisateurs afin de recommander les éléments les plus pertinents pour chaque utilisateur.

Parmi les approches classiques de systèmes de recommandation, nous pouvons citer les approches dites collaboratives qui se basent sur des données explicites, comme des votes, ou des données implicites, comme des achats ou des consultations de pages, afin de chercher des similarités entre les utilisateurs ou les items. On y retrouve notamment les approches utilisateur-utilisateur et item-item que l'on étudiera dans cet article. On peut également considérer les approches dites agglomératives qui sont aussi des méthodes de filtrage collaborative consistant à regrouper des éléments similaires en clusters ou en groupes afin de pouvoir recommander des éléments similaires à ceux qui ont été appréciés par les utilisateurs. De nombreux articles ont étudié ces approches collaboratives, on peut notamment citer l'article "Collaborative Filtering Recommender Systems" [Schafer *et al.*, 2007] qui introduit les concepts clés du filtrage collaboratif en passant en revue différents algorithmes et en discutant leurs utilisations et comment les évaluer.

Ensuite, nous pouvons citer, parmi les autres méthodes classiques de systèmes de recommandation, les approches dites contenu. Contrairement aux approches collaboratives, les approches contenu utilisent les attributs des utilisateurs, comme leur âge, leur sexe, leur profession, ainsi que les attributs des items, comme leur auteur, leur date de production, afin de trouver des similarités avec d'autres utilisateurs ou items ou pour prédire directement les préférences d'un utilisateur. Parmi toutes les approches contenu, nous pouvons notamment citer les méthodes bayésiennes qui calculent des probabilités conditionnelles pour des utilisateur d'aimer certains items en fonction des attributs propres à ces utilisateurs et ces items. De nombreux travaux ont étudié ces approches contenu, notamment "Content-based Recommender Systems : State of the Art and Trends" [Lops *et al.*, 2011] qui fournit une vue d'ensemble des systèmes de pointe, en mettant en évidence les techniques qui se sont révélées les plus efficaces, et les domaines d'application dans lesquels elles ont été adoptées.

Enfin, nous pouvons aussi prendre en compte les approches dites hybrides qui combinent les méthodes de filtrage collaboratif et contenu, par exemple en imputant des données manquantes grâce à une méthode de filtrage collaboratif puis en utilisant ces données imputées pour prédire les préférences des utilisateurs. Nombreuses de ces méthodes hybrides se basent sur des algorithmes neuronaux qui combinent naturellement les aspects collaboratif et contenu du filtrage. Ces méthodes par apprentissage profond sont assez populaires et consistent actuellement la majorité des techniques à l'état de l'art. On peut notamment citer la méthode de Two Towers qui repose sur l'utilisation de deux structures de couches denses, une pour les utilisateurs et l'autre pour les items puis on calcule la similarité comme le produit scalaire entre la sortie des deux tours. On peut également considérer la méthode AutoREC qui reprend l'idée des auto-encodeurs adaptée à des votes pour des utilisateurs ou des items. Parmi tous les travaux qui ont été menés sur le sujet de ces approches hybrides, on peut citer l'article "Hybrid Recommender Systems : Survey and Experiments" [Burke, 2002] qui passe en revue différents systèmes de recommandation hybrides.

Nous évaluerons toutes les méthodes énoncées précédemment sur des métriques classiques comme l'erreur moyenne quadratique (MSE) et l'erreur moyenne absolue (MAE) mais aussi sur des facteurs de succès comme l'utilité, la diversité, la nouveauté, la sérendipité et la couverture. Nous utiliserons la base de données MovieLens qui regroupe 100 000 votes pour 1682 films et 943 utilisateurs différents. Les votes correspondent à des notes à valeurs entières entre 1 et 5. Les utilisateurs ont différents attributs comme leur âge, leur sexe et leur travail tandis que les films ont comme attributs leur titre, leur genre et leur date de sortie. De nombreux articles de recherche ont aussi travaillé à explorer et comparer différents systèmes de recommandation comme l'article "A Comparative Study of Recommendation Systems" qui passe en revue et évalue différentes approches collaboratives, contenu et hybrides. [Lokesh, 2019]. Cependant, nombreux de ces travaux ne prennent souvent en compte que les métriques classiques, telles que MSE ou MAE, mais ne considère pas les facteurs de succès, bien que ce soit des critères très importants qui impactent considérablement l'expérience utilisateur.

2 Présentation des méthodes

2.1 Approches collaboratives

Les méthodes collaboratives se basent sur les données explicites ou implicites, en l'occurrence ici on se basera sur les vecteurs de votes, afin de chercher des similarités entre utilisateurs ou entre items pour prédire les votes des utilisateurs.

Version utilisateur-utilisateur

L'approche utilisateur-utilisateur consiste à prédire les votes d'un utilisateur spécifique à partir des votes des autres utilisateurs en faisant une somme de leurs votes pondérés selon leur similarité.

La valeur estimée du vote de l'utilisateur a pour un l'item j, $\hat{v}_{a,j}$, est la somme pondérée des votes des autres utilisateurs, v_i, qui ont des votes communs avec l'utilisateur a :

$$\hat{v}_{a,j} = \lambda * \sum_{i=1}^{n} w_{a,i} * v_{i,j}$$

Où n est le nombre d'utilisateurs ayant des votes communs avec l'utilisateur a. $w_{a,i}$ correspond à la similarité entre l'utilisateur a et l'utilisateur i, mesuré comme la similarité cosinus entre les vecteurs de votes des deux utilisateurs. Enfin, λ représente le facteur de normalisation de poids, c'est-à-dire $\lambda = 1/(\sum_{i=1}^{n} |w_{a,i}|)$.

Cependant, cette formule ne prend pas en compte le biais des utilisateurs, c'est-à-dire leur propension à donner une note plus élevée ou plus basse que la moyenne. On peut ainsi évaluer de manière plus précise le vote de l'utilisateur a pour un l'item j, $\hat{v}_{a,j}$, en intégrant le biais utilisateur :

$$\hat{v}_{a,j} = \bar{v}_a + \lambda * \sum_{i=1}^{n} w_{a,i} * (v_{i,j} - \bar{v}_i)$$

Où $\bar{v}_a$ représente le vote moyen d'un utilisateur a et $\bar{v}_i$ le vote moyen de l'utilisateur i.

Enfin, on peut aussi considérer que seuls les votes des voisins les similaires de l'utilisateur sont intéressants à prendre en compte pour prédire les votes. Ainsi, on peut évaluer plus précisément le vote de l'utilisateur a pour un l'item j, $\hat{v}_{a,j}$, uniquement en prenant en compte les 100 voisins les plus proches :

$$\hat{v}_{a,j} = \bar{v}_a + \lambda * \sum_{i \in N_{100}} w_{a,i} * (v_{i,j} - \bar{v}_i)$$

où N_{100} correspond aux 100 plus proches voisins de l'utilisateur a en terme de similarité cosinus.

Version item-item

L'approche item-item est exactement comme l'approche utilisateur-utilisateur sauf qu'on prédit le vote d'un utilisateur pour un certain item à partir des votes de ce même utilisateur pour les autres items en pondérant les votes selon la similarité entre les items.

Ainsi, la valeur estimée du vote de l'utilisateur a pour un l'item j, $\hat{v}_{a,j}$, sans prendre en compte le biais de l'item est :

$$\hat{v}_{a,j} = \lambda * \sum_{i=1}^{m} w_{j,i} * v_{a,i}$$

Où m est le nombre d'items ayant des votes communs avec l'item j. $w_{j,i}$ correspond à la similarité entre l'item j et l'item i, mesuré comme la similarité cosinus entre les vecteurs de votes des deux items. Enfin, λ représente le facteur de normalisation de poids, c'est-à-dire $\lambda = 1/(\sum_{i=1}^{m} |w_{j,i}|)$.

On peut évaluer de manière plus précise le vote de l'utilisateur a pour un l'item j, $\hat{v}_{a,j}$, en intégrant le biais item :

$$\hat{v}_{a,j} = \bar{v}_j + \lambda * \sum_{i=1}^{m} w_{j,i} * (v_{a,i} - \bar{v}_i)$$

Où $\bar{v}_j$ représente le vote moyen de l'item j et $\bar{v}_i$ le vote moyen de l'item i.

Enfin, on peut encore améliorer la prédiction du vote de l'utilisateur a pour un l'item j, $\hat{v}_{a,j}$, uniquement en prenant en compte les 100 voisins les plus proches :

$$\hat{v}_{a,j} = \bar{v}_j + \lambda * \sum_{i \in N_{100}} w_{j,i} * (v_{a,i} - \bar{v}_i)$$

Où N_{100} correspond aux 100 plus proches voisins de l'item j en terme de similarité cosinus.

Méthode SVD

Les approches par factorisation de matrices sont un autre type d'approche collaborative dans lequel on décompose notre matrice de votes R en un produit de trois matrices, U pour les utilisateurs de taille (nombre_utilisateurs, l), V pour les items de taille (nombre_items, l) et Σ une matrice diagonale qui représente les valeurs singulières de R où l est le nombre de facteurs latents. Nous avons implanté la décomposition SVD où notre matrice de votes est calculée à l'aide de la formule suivante :

$$R = U\Sigma V^T$$

Où :
— U matrice orthonormale de taille (nombre_utilisateurs, l) qui représente les vecteurs propres de RR^T
— Σ une matrice diagonale de taille (l, l) qui représente les valeurs singulières de R (racines carrées des valeurs propres positives de RR^T)
— V : matrice orthonormale de taille (nombre_items, l) qui représente les vecteurs propres de $R^T R$

Les valeurs singulières de R indiquent l'importance relative des facteurs latents et sont ordonnées par ordre décroissant dans Σ. Un problème de la factorisation SVD est la gestion des valeurs manquantes dans la matrice de votes de départ. Pour cela on réalise une imputation : pour chaque valeur manquante, on calcule la moyenne de la moyenne des autres valeurs de la ligne et de la moyenne des autres valeurs de la colonne. On applique la décomposition SVD à notre matrice de votes imputée et on regarde les erreurs obtenues lorsqu'on prédit nos votes on considérant différents nombres de facteurs latents. Nos meilleurs résultats sont obtenus pour l = 14 facteurs latents.

Méthode K-means

Les méthodes agglomératives permettent de définir des classes d'utilisateurs ou d'items. On crée des clusters à partir des valeurs de votes et on assigne à un utilisateur (ou un item) la classe dont le centroid est le plus similaire. La méthode agglomérative que nous avons implanté est K-means.

Algorithm 1 K-means

Input : ensemble de données X
Parameter : Nombre de clusters k

1: Initialiser C avec k centres de cluster aléatoires
2: **while** les centres de cluster changent **do**
3: Assigner chaque point $x_i \in X$ au centre de cluster c_j le plus proche : $j = \arg\min_{c \in C} d(x_i, c)$
4: **for** chaque cluster $c_j \in C$ **do**
5: Calculer le nouveau centre c'_j en prenant la moyenne des points assignés au cluster : $c'j = \frac{1}{|S_j|} \sum x_i \in S_j x_i$
6: **end for**
7: Mettre à jour les centres des clusters : $C \leftarrow c'_1, c'_2, \ldots, c'_k$
8: **end while**
9: **return** C et S

Dans notre implantation nous utilisons la librairie pyclustering et la corrélation de Pearson comme mesure de similarité. Nous avons essayé différentes valeurs pour le nombre de clusters (2, 5, 10, 20, 40) et avons obtenu les meilleurs résultats pour 20 clusters.

2.2 Approches contenu

Les approches contenu utilisent des attributs propres aux utilisateurs ou aux items pour établir la similarité entre ces derniers et effectuer des prédictions.

Méthode bayésienne

La méthode bayésienne, que nous avons implanté utilise cette approche : on calcule la probabilité d'une hypothèse H (avoir une valeur de vote égale à v) étant donné des évidences $E_1, .., E_p$.

Pour cela, on utilise la méthode du ratio de chances :

$$O(H|E_1, ..., E_p) = \frac{P(H|E_1, ..., E_p)}{P(\bar{H}|E_1, ..., E_p)}$$

En appliquant la formule de Bayes, on obtient :

$$O(H|E_1, ..., E_p) = \frac{P(H)}{P(\bar{H})} \frac{P(E_1, ..., E_p|H)}{P(E_1, ..., E_p|\bar{H})}$$

En supposant l'indépendance des évidences, on obtient :

$$O(H|E_1, ..., E_p) = \frac{P(H)}{P(\bar{H})} \prod_{i=1}^{p} \frac{P(E_i|H)}{P(E_i|\bar{H})}$$

On peut ensuite revenir à la valeur de la probabilité :

$$P(H|E_1, ..., E_p) = \frac{O(H|E_1, ..., E_p)}{1 + O(H|E_1, ..., E_p)}$$

Les différentes probabilités sont calculées à l'aide des fréquences. Par exemple, pour calculer la probabilité d'aimer un film i sachant que l'utilisateur est une femme, on compte dans notre ensemble de données le nombre de femmes qui ont aimé le film i et on divise par le nombre total d'utilisateurs qui sont des femmes.

La valeur attendue pour le vote est ensuite calculée comme l'espérance du vote sachant les différentes évidences propres à l'utilisateur :

$$E(v_{u,i}) = \sum_v P(v_{u,i} = v|E_1, ..., E_p) \times v$$

Où $v_{u,i}$ est le vote de l'utilisateur u pour l'item i et $E_1, ..., E_p$ correspondent aux différentes évidences propres à l'utilisateur u (âge, sexe, métier...).

Dans notre cas, nous considérons trois évidences / attributs pour estimer les votes : l'âge, le sexe et le métier des utilisateurs.

2.3 Approches hybrides

Les méthodes par approche hybride utilisent à la fois les informations des votes et les caractéristiques des utilisateurs et des items pour prédire les votes. Nous avons ici implanté les modèles Two Towers et AutoREC. Pour cette partie nous utilisons les librairies TensorFlow et Keras.

Modèle Two Towers

La méthode Two Towers consiste à utiliser deux réseaux de neurones feed-forward : un pour les utilisateurs et un pour les items. Les informations des utilisateurs et items (votes et features) sont transformées par un embedding puis passées en entrée de leur réseau de neurones respectif. La similarité est calculée par un produit scalaire entre les sorties des deux réseaux de neurones. La prédiction est calculée à l'aide de la formule suivante :

$$pred_{i,j} = \sigma(b + b_{u_i} + b_{f_j} + E_{u_i}^T E_{f_j}) * (M_{vote} - m_{vote}) + m_{vote}$$

Où :

— $(E_{u_i}, E_{f_j}) \in \mathbb{R}^n \times \mathbb{R}^n$ sont respectivement les embeddings de l'utilisateur u_i et du film f_j.
— $b \in \mathbb{R}$ est la constante qui représente la moyenne.
— $b_{u_i} \in \mathbb{R}$ est le biais associé à l'utilisateur u_i.
— $b_{f_j} \in \mathbb{R}$ est le biais associé au film f_j.
— σ est la fonction sigmoïde.
— M_{vote} et m_{vote} sont respectivement les valeurs maximale et minimale possibles pour les votes.

On effectue le produit scalaire entre les vecteurs utilisateur et item, on ajoute les biais (utilisateur, item et moyenne des votes) et on applique une fonction sigmoïde pour obtenir une valeur comprise entre 0 et 1. Ensuite, on multiplie cette valeur par la différence entre la valeur maximale et minimale possible d'un vote, puis on ajoute la valeur minimale pour obtenir une valeur prédite comprise entre m_{vote} et M_{vote}, en l'occurence ici entre 1 et 5.

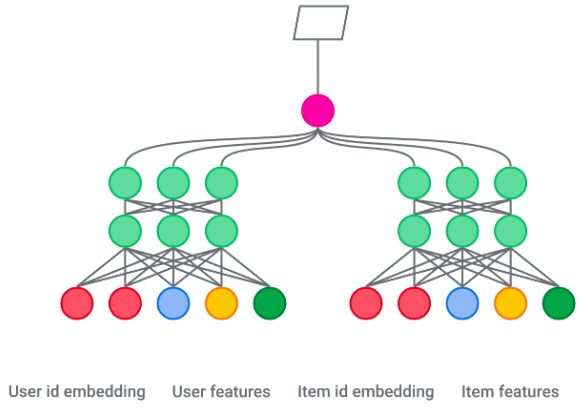

FIGURE 1 – Architecture du modèle Two Towers

Modèle AutoREC

Le modèle AutoREC est basé sur une architecture auto-encodeur : il va prendre en entrée un vecteur de valeurs partiellement observées, le projeter dans un espace latent de plus faible dimension puis le reconstruire en sortie afin de minimiser une fonction de perte de reconstruction.
La fonction de perte à minimiser est une version modifiée de l'erreur quadratique où on ne considère que les valeurs des votes observés (présents dans la matrice de votes initiale).

Concernant l'architecture de notre auto-encodeur, nous utilisons une version "profonde" comme conseillé dans l'article "AutoRec : Autoencoders Meet Collaborative Filtering" [Sedhain *et al.*, 2015]. Notre réseau de neurones est composée de 3 couches cachées (respectivement constituées de 500, 250 et 500 neurones) chacune avec une fonction d'activation sigmoïde, et une couche de sortie pour que notre vecteur de sortie soit de la même taille que celui en entrée. Nous avons essayé d'autres configurations de profondeur (1 et 6 couches cachées) et largeur (500, 1000, 500 neurones) mais les résultats obtenus en terme de RMSE sur l'ensemble de validation étaient inférieurs.

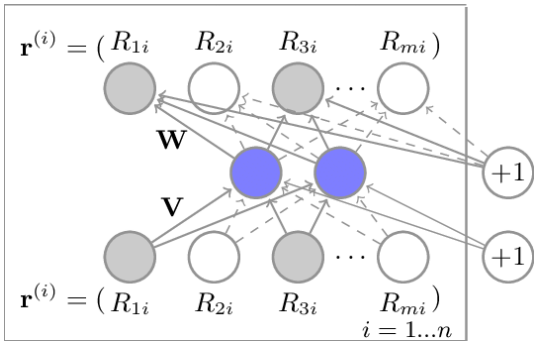

FIGURE 2 – Architecture du modèle AutoRec

3 Évaluations expérimentales

3.1 Métriques et facteur de succès

Afin de comparer les différentes méthodes, nous allons les évaluer sur base des métriques de MSE et MAE. Cependant, plusieurs autres facteurs doivent être pris en compte pour évaluer la performance d'un système de recommandation.

Dans la prochaine partie, nous allons utiliser les notations suivantes :

— R est une liste de recommandations pour un utilisateur. Dans notre cas, cela correspond aux 10 films avec la note prédite la plus élevée pour l'utilisateur
— U et I sont respectivement les ensembles des utilisateurs et items.
— |.| est l'opérateur de cardinalité.

Utilité

L'utilité des recommandations mesure à quel point les recommandations sont susceptibles de plaire à l'utilisateur. L'utilité peut être calculée simplement comme la moyenne arithmétique des valeurs de votes des items recommandés pour tous les utilisateurs.

Diversité

La diversité est la capacité d'un système de recommandation à recommander des items appartenant à des catégories ou à des dimensions latentes différentes. Pour chaque utilisateur, on calcule la distance entre les items recommandés, ici nous utilisons la distance cosinus, et on divise par le nombre de

terme pour avoir une moyenne. On peut exprimer la diversité à l'aide de la formule suivante :

$$Diversity = \frac{\sum_{i \in R} \sum_{j \in R\{i\}} cos(i,j)}{|R| \times (|R| - 1)}$$

Nouveauté

La nouveauté désigne la capacité de recommander à un utilisateur un item pas déjà recommandé ou consulté. Pour mesurer la nouveauté, on utilise $p(i)$ la proportion d'utilisateurs ayant voté pour un item i. On calcule $p(i)$ pour tous les items i recommandés à l'utilisateur, on applique $-log_2()$ (un item beaucoup voté entraîne une faible nouveauté), et on divise par le nombre d'items recommandés pour avoir une moyenne. On peut écrire la nouveauté à l'aide de la formule suivante :

$$Novelty = \frac{\sum_{i \in R} -log_2(p(i))}{|R|}$$

Sérendipité

La sérendipité est la capacité d'un système de recommandation à produire une recommandation inattendue mais quand même pertinente pour l'utilisateur. Le caractère inattendu d'une recommandation peut être mesuré à l'aide du PMI (Pointwise Mutual Information). Ainsi, pour évaluer la sérendipité nous utilisons la formule suivante :

$$Serendipity = PMI \times utility$$

Pour chaque utilisateur, on calcule le PMI entre chaque paire d'items recommandés à l'aide des fréquences des votes selon la formule suivante :

$$PMI(i,j) = log_2(p(i,j)/(p(i) \times p(j)))$$

— $p(i)$ est le nombre d'utilisateurs ayant votés pour l'item i divisé par le nombre total d'utilisateurs (de même pour $p(j)$)
— $p(i,j)$ est le nombre d'utilisateurs ayant votés à la fois pour les items i et j divisé par le nombre total d'utilisateurs.

On calcule ensuite la moyenne des PMI de chaque paire d'items et on multiplie par l'utilité des recommandations de l'utilisateur pour obtenir une mesure de la sérendipité des recommandations d'un utilisateur. On répète ce calcul pour tous les utilisateurs et on calcule la sérendipité globale par la moyenne des sérendipités.

Couverture

La couverture est la capacité du système de recommandation à recommander la plus grande partie du catalogue des items. Ce facteur peut être particulièrement important, par exemple dans le cas des sites de rencontres, où les recommandations sont d'autres utilisateurs, afin qu'un profil ne soit pas "exclu" des recommandations. Pour calculer la couverture, on compte le nombre d'items qui apparaissent au moins dans une des recommandations et on divise par le nombre total d'items. On peut l'écrire à l'aide de la formule suivante :

$$Coverage = \frac{|\bigcup_{u \in U} R_u|}{|I|}$$

Où R_u représente les items recommandés à l'utilisateur u.

3.2 Résultats

Les résultats obtenus pour les approches collaboratives (utilisateur-utilisateur, item-item, SVD, K-means), contenu bayésienne et hybrides (Two Towers et AutoREC) sont présentés dans les tableaux ci-dessous. Les approches utilisateur-utilisateur et item-item correspondent aux versions avec biais et avec voisins rapprochés.

Métrique	MSE	MAE
Utilisateur-Utilisateur	0.900	0.742
Item-Item	**0.843**	**0.721**
SVD	0.898	0.756
K-means	0.951	0.785
Bayes	1.146	0.855
Two Towers	0.855	0.737
AutoREC	1.055	0.734

TABLE 1 – Comparaison MSE et MAE obtenus pour les différentes méthodes

Facteur	Utilité	Diversité	Nouveauté
Utilisateur-Utilisateur	5.254	0.880	5.774
Item-Item	**5.343**	0.940	7.737
SVD	4.301	**0.988**	**8.696**
K-means	0.186	0.569	1.894
Bayes	4.264	0.766	3.339
Two Towers	2.938	0.893	6.218
AutoREC	3.892	0.940	6.315

TABLE 2 – Comparaison des facteurs de succès utilité, diversité et nouveauté pour les différentes méthodes

Facteur	Sérendipité	Couverture
Utilisateur-Utilisateur	6.072	**0.386**
Item-Item	7.863	0.349
SVD	**8.773**	0.070
K-means	3.085	0.189
Bayes	4.093	0.173
Two Towers	6.484	0.006
AutoREC	6.600	0.011

TABLE 3 – Comparaison des facteurs de succès sérendipité et couverture pour les différentes méthodes

Pour évaluer la performance des différents systèmes de recommandation, nous avons effectué une validation croisée avec 5 plis sur la base de données MovieLens pour les approches collaboratives et contenu. Pour les approches

hybrides, nous avons divisé aléatoirement l'ensemble des votes de MovieLens en une base d'entraînement et une base de validation. La base d'entraînement contient 80% des votes et la base de validation contient les 20% restants.

L'implémentation des différentes méthodes ainsi que leurs évaluations sont disponibles sur notre **GitHub** : *https://github.com/Pierrosin/Exploration-systemes-de-recommandation*.

4 Discussion et conclusion

L'approche collaborative item-item obtient les meilleurs résultats en terme de MSE et MAE, suivi de près par l'approche hybride Two Towers. L'approche utilisant le modèle AutoREC obtient les résultats les moins bons en terme de MSE. Cependant, comme AutoREC utilise directement les vecteurs de votes, la méthode se prête bien à l'ajout de nouveaux utilisateurs / items à notre ensemble de données, contrairement au modèle Two Towers qui utilise des embeddings utilisateurs et items et nécessite un nouvel entraînement à chaque ajout d'utilisateur ou d'item. On remarque que des approches finalement assez simples comme celle de l'utilisateur-utilisateur prédisent mieux les votes que des modèles d'apprentissage profond comme Two Towers ou AutoREC. Cela peut être dû au fait que les modèles d'apprentissage profond ont de nombreux hyperparamètres qu'il faut choisir de manière optimale sans quoi les résultats peuvent être bien moins bons. Pour améliorer les résultats sur ces méthodes, nous aurions pu explorer plus de combinaisons d'hyperparamètres avec une recherche en gridsearch par exemple, mais cela nécessite beaucoup de temps et de ressource. De plus, ces modèles d'apprentissage profond nécessitent aussi beaucoup de données d'entraînement afin de pouvoir généraliser au mieux ses capacités de prédiction de votes. Pour améliorer les résultats de ces méthodes, nous aurions pu utiliser une base d'entraînement plus grande comme la version de MovieLens avec 25 million de votes sur 62 000 films par 162 000 utilisateurs. Cependant, le temps d'entraînement de ces méthodes auraient été trop long et les ressources nécessaires trop importantes.

Concernant les facteurs de succès, on observe des disparités entre les différentes méthodes. En terme d'utilité et de couverture, les approches collaboratives utilisateurs-utilisateurs et item-item obtiennent les meilleurs résultats, mais elles ne fournissent pas les recommandations les plus diversifiées. La méthode par factorisation de matrice SVD recommande des films dont la diversité, la nouveauté et la sérendipité sont les élevées parmi toutes les méthodes. Cependant, la méthode SVD n'offre pas une très bonne couverture. La méthode K-means n'obtient pas de très bons résultats en particulier en terme de diversité et de nouveauté, ce qui n'est pas surprenant étant donné qu'elle se contente de prédire les votes des utilisateurs comme le centroïde du clusters le plus proche, il y a donc très peu de diversité de recommandations. La méthode bayésienne n'offre également pas de très bons résultats sur les facteurs de succès. Nous aurions pu essayer d'intégrer davantage d'attributs d'utilisateur ou d'item pour améliorer sa performance. Enfin, les approches par réseaux de neurones n'obtiennent pas de bons résultats sur la plupart des facteurs. A nouveau, ces méthodes possèdent davantage de paramètres et on pourrait probablement avoir de meilleurs résultats en calibrant mieux ces derniers.

Ainsi, on peut voir que les métriques classiques comme la MSE et la MAE ne sont pas les seuls critères à prendre en compte lors de l'évaluation d'un système de recommandations. En effet, les méthodes obtenant les meilleurs résultats sur ces métriques de MSE et MAE ne sont pas nécessairement les meilleurs selon les critères de succès. Chaque approche peut avoir son intérêt en fonction du cas d'usage. En particulier, concernant l'expérience utilisateur, il est tout aussi important voire même plus important d'avoir des recommandations qui soient diversifiées, inattendues ou nouvelles que simplement des recommandations qui s'avèrent utiles mais prévisibles, peu diversifiés ou déjà consultées.

L'approche que nous avons utilisée pour apprendre le sujet des systèmes de recommandation consistait à lire différents articles de recherche ou des cours sur les systèmes de recommandation et à réimplémenter différentes approches. Avec du recul, nous aurions pu passer moins de temps à implémenter les méthodes qui étaient de toutes les façons facilement disponibles sur internet et consacrer plus de temps à évaluer ces méthodes dans différents contextes ou avec d'autres bases de données. Il n'était pas toujours évident de bien saisir les algorithmes de certaines approches décrites dans les articles et nous avions parfois des difficultés à correctement les implémenter. Néanmoins, l'expérience est très enrichissante car elle nous aura permis de saisir en profondeur les concepts clés autour des systèmes de recommandation que ce soit en théorie et en pratique et d'avoir une large vision des différentes méthodes disponibles actuellement.

Références

[Burke, 2002] Robin Burke. Hybrid recommender systems : Survey and experiments. *User Modeling and User-Adapted Interaction*, 12, November 2002.

[Lokesh, 2019] Ashwini Lokesh. A comparative study of recommendation systems. *Masters Theses & Specialist Projects*, October 2019.

[Lops *et al.*, 2011] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems : State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*. January 2011.

[Schafer *et al.*, 2007] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web : Methods and Strategies of Web Personalization*. January 2007.

[Sedhain *et al.*, 2015] Suvash Sedhain, Aditya Menon, Scott Sanner, and Lexing Xie. Autorec : Autoencoders meet collaborative filtering. May 2015.