Sorbonne Université

M1 Informatique - DAC
Compte rendu PLDAC

Analyse de Reviews et Recommandation

Etudiants :
Ghiles Ouhenia
Amayas Sadi
Mathilde Rodrigues Da Rocha

Encadrants :
Nicolas Baskiotis
Vincent Guigue

24 mai 2023

Table des matières

1 Abstract

Notre projet, intitulé « Analyse de reviews et recommandations », vise à exploiter les données provenant d'un site internet recensant un grand nombre de jeux et d'avis associés, rédigés par des utilisateurs. En utilisant des techniques de Systèmes de Recommantion, notre objectif est de développer un modèle de recommandation de jeux, basé sur les préférences des utilisateurs et des jeux similaires appréciés par d'autres joueurs partageant les mêmes intérêts. Les plateformes d'opinions en ligne sont extrêmement précieuses car elles permettent aux utilisateurs de partager leurs expériences et leurs opinions sur des produits ou des services. L'analyse des notes et avis des utilisateurs est devenue essentielle pour extraire des informations pertinentes et mesurer le ressenti des clients envers une marque ou un produit. Elle permet également de répondre à de nombreux défis, tels que l'amélioration de l'expérience utilisateur, la gestion de la réputation en ligne, la détection des tendances du marché et l'identification des insatisfactions et des problèmes de produits. De même, la recommandation de produits basée sur les préférences des utilisateurs est un enjeu majeur pour de nombreuses entreprises de commerce en ligne. Les modèles de recommandation doivent être capables de prédire les préférences des utilisateurs en fonction de leur historique de navigation et de leurs comportements d'achat, en utilisant des techniques de traitement du langage naturel et de recommandation. Dans notre projet, nous utiliserons des techniques avancées de systèmes recommandation pour proposer des jeux pertinents aux utilisateurs.

2 Contexte

Les systèmes de recommandation (SR) sont des outils qui fournissent des suggestions d'articles susceptibles d'être utiles à un utilisateur. Ces suggestions visent à soutenir les utilisateurs dans divers processus de prise de décision tels que les articles à acheter, la musique à écouter ou les services à utiliser. Ils font généralement appel à des algorithmes basés sur les préférences et les comportements passés de l'utilisateur, ainsi que sur des modèles de similarité entre les différents éléments proposés. Les algorithmes de SR proposent une liste d'items donnée en prédisant quel produit ou service est, basé sur les préférences de l'utilisateur, le plus adapté. Les préférences sont collectées soit de manière explicite, c'est-à-dire que l'utilisateur a noté certains produits, soit de manière implicite, l'utilisateur a navigué sur une page contenant un certain produit pour générer des recommandations personnalisées et pertinentes. L'objectif principal est d'améliorer l'expérience de l'utilisateur en lui offrant des suggestions adaptées à ses goûts et à ses besoins spécifiques. Les SR se sont avérés être un moyen précieux pour les utilisateurs en ligne de faire face à la surcharge d'information et sont devenus l'un des outils les plus puissants et populaires dans le commerce électronique. Les motivations des entreprises qui proposent ce service sont d'augmenter le nombre d'items vendus, vendre des items plus divers (ne pas vendre uniquement ceux qui sont les plus populaires car ils ne sont pas forcément adaptés pour tout public), améliorer la satisfaction de l'utilisateur et sa fidélité et mieux comprendre ce que l'utilisateur veut.

Diverses techniques de génération de recommandations ont été proposées et, au cours de la dernière décennie, beaucoup d'entre elles ont également été déployées avec succès dans des environnements commerciaux.

Les systèmes de recommandation jouent un rôle important dans des sites Internet tels qu'Amazon, YouTube, Netflix, Yahoo, etc. De plus, de nombreuses entreprises de médias développent et déploient maintenant des systèmes de recommandation dans le cadre des services qu'elles fournissent à leurs abonnés. Par exemple, Netflix a attribué un prix, "Netflix Prize" [15] d'un million de dollars à l'équipe qui a réussi à améliorer les performances de son système de recommandation.

Cependant, malgré leur popularité et leur utilité, les systèmes de recommandation ne sont pas sans défis. L'un des principaux défis est de recommander des articles qui sont non seulement pertinents pour l'utilisateur, mais aussi nouveaux et intéressants. C'est là que le filtrage collaboratif et l'approche basée sur le contenu entrent en jeu.

Le filtrage collaboratif et l'approche basée sur le contenu sont deux des paradigmes les plus couramment utilisés dans les systèmes de recommandation. Le filtrage collaboratif se base sur le comportement passé de l'utilisateur et celui d'autres utilisateurs pour prédire ce que l'utilisateur pourrait aimer. D'autre part, l'approche basée sur le contenu recommande des articles en se basant sur la description de l'article et un profil des intérêts de l'utilisateur. Ces deux approches ont leurs propres avantages et inconvénients, et sont souvent utilisées en combinaison pour améliorer la qualité des recommandations.

Dans les sections suivantes, nous examinerons plus en détail le filtrage collaboratif et l'approche basée sur le contenu, en expliquant leur fonctionnement, leurs avantages et inconvénients, leurs évaluations et leurs applications.

2.1 Filtrage collaboratif

Le filtrage collaboratif est l'une des techniques les plus couramment utilisées dans les systèmes de recommandation qui se base sur le comportement passé de l'utilisateur et celui d'autres utilisateurs pour prédire ce que l'utilisateur pourrait aimer. Il repose sur l'idée que si deux utilisateurs sont d'accord sur un problème, ils sont susceptibles d'être d'accord sur d'autres problèmes. Les systèmes de filtrage collaboratif ont été largement utilisés dans de nombreux systèmes de recommandation en raison de leur efficacité et de leur simplicité. Cependant, ils souffrent de certains problèmes tels que le démarrage à froid, la rareté (sparsity, matrice utilisateur-article très clairsemée, où la majorité des entrées sont inconnues) et l'évolutivité (en raison du nombre d'utilisateurs et d'items qui augmente continuellement, nécessite de sans cesse s'adaptater). Il existe deux approches principales pour le filtrage collaboratif : basé sur la factorisation matricielle et basé sur le voisinnage [9].

Avant cela, nous allons vous présenter l'algorithme de base souvent utilisé comme point de départ pour comparer les performances d'autres algorithmes de recommandation plus sophistiqués

Baseline algorithm L'algorithme baseline est utilisé pour estimer les notes manquantes dans un ensemble de données de notation collaborative. Il s'agit d'un algorithme simple qui utilise des prédictions basées sur la moyenne globale des notes μ, les biais d'utilisateur b_u et les biais d'item b_i.

Cet algorithme repose sur le fait qu'il existe une tendance pour certains utilisateurs à attribuer des notes plus élevées que d'autres, ainsi que pour certains éléments à recevoir des notes plus élevées que d'autres, ce que nous avons constaté précédemment durant notre analyse des données. Il est donc important de prendre en compte ces effets lors de l'estimation des notes manquantes. La prédiction basique r_{ui} d'une note inconnue est donc désignée par b_{ui} obtenu en ajoutant à la moyenne globale la contribution des biais d'utilisateur et des biais d'élément associés à cette note spécifique [3] :

$$r_{ui} = b_{ui} = \mu + b_u + b_i$$

où

- $b_i = \frac{\sum_{u:(u,i)\in\mathcal{K}}(r_{ui}-\mu)}{\lambda_2+|\{u|(u,i)\in\mathcal{K}\}|}$ pour chaque item
- $b_u = \frac{\sum_{i:(u,i)\in\mathcal{K}}(r_{ui}-\mu-b_i)}{\lambda_3+|\{i|(u,i)\in\mathcal{K}\}|}$ pour chaque utilisateur
- Avec $\mathcal{K}$ l'ensemble où sont stockées les paires (u,i). $\mathcal{K} = \{(u,i)|r_{ui} \text{ est connu}\}$

Les moyennes sont ramenées à zéro en utilisant les paramètres de régularisation λ_2, λ_3 qui sont déterminés par validation croisée. Par exemple, les valeur typiques dans l'ensemble de données Netflix sont $\lambda_2 = 25$, $\lambda_3 = 10$. Nous allons essayer de déterminer les valeurs optimales de régularisation sur notre ensemble de données.

2.1.1 Modèles basés sur la factorisation matricielle

SVD L'algorithme SVD (Singular Value Decomposition) est un algorithme de factorisation matricielle qui est utilisé dans la bibliothèque Surprise pour la recommandation de films. Il est basé sur la même technique utilisée par Simon Funk lors du Netflix Prize.

L'idée de base est de représenter à la fois les utilisateurs et les items dans un espace de caractéristiques de faible dimension. Chaque utilisateur et chaque item est représenté par un vecteur dans cet espace, et la note qu'un utilisateur donnerait à un item est prédite en prenant le produit scalaire de leurs vecteurs.

Plus précisément, l'algorithme représente chaque utilisateur u par un vecteur p_u et chaque item i par un vecteur q_i. Ces vecteurs sont appris à partir des données d'entraînement en minimisant la somme des carrés des erreurs de prédiction, avec une régularisation pour éviter le surapprentissage. La note prédite pour l'utilisateur u et l'item i est donnée par [3, 4, 9] :

$$r_{ui} = b_{ui} = \mu + b_u + b_i + q_i^T p_u$$

Les vecteurs p_u et q_i et les biais b_u et b_i sont appris en utilisant une technique d'optimisation telle que la descente de gradient stochastique (SGD). À chaque itération, les paramètres sont mis à jour en utilisant les formules suivantes :

$$b_u^{(new)} = b_u + \gamma(e_{ui} - \lambda b_u)$$

$$b_i^{(new)} = b_i + \gamma(e_{ui} - \lambda b_i)$$

$$p_u^{(new)} = p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u)$$

$$q_i^{(new)} = q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i)$$

où :

- $e_{ui} = r_{ui} - \hat{r}_{ui}$ est l'erreur de prédiction pour l'utilisateur u et l'item i
- γ est le taux d'apprentissage
- λ est le terme de régularisation qui prévient le surapprentissage

2.1.2 Modèles basés sur le voisinnage

KNN Basic L'algorithme KNN Basic fonctionnne en trouvant les "k" utilisateurs (ou items) les plus similaires à un utilisateur (ou item) donné, puis en utilisant leurs notes pour prédire la note que l'utilisateur u donnerait à un item i non noté.

La prédiction $\hat{r}_{ui}$ est donnée par :

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u, v) \cdot r_{vi}}{\sum_{v \in N_i^k(u)} sim(u, v)}$$

L'algo calcule une moyenne pondérée des notes que les "k" voisins de u ont données à i où $N_i^k(u)$ est l'ensemble des "k" voisins de u qui ont noté l'item i.

Surprise propose différentes mesures de similarité préféfinies telles que : `cosine` (similarité cosinus), `msd` (Mean Squared Difference), `pearson` (coefficient de corrélation de Pearson).
Il existe d'autre algorithmes que nous n'allons pas présenter plus que cela ici tels que :

- **SVD++** qui est une extention de SVD qui introduit des facteurs latents pour chaque item noté par un utilisateur en plus des facteurs latents associés à chaque utilisateur et à chaque item. L'idée derrière l'algorithme est que la notation d'un item par un utilisateur fournit non seulement une information explicite sur la préférence de l'utilisateur pour cet item (capturée par le vecteur de facteurs latents de l'utilisateur), mais aussi une information implicite sur les préférences de l'utilisateur en général (capturée par les vecteurs de facteurs latents des items notés par l'utilisateur) [9].
- **Co-clustering** : basé sur la collaboration qui effectue simultanément un clustering des utilisateurs et des items. L'idée est que certains groupes d'utilisateurs ont tendance à noter certains groupes d'items de manière similaire, et ces tendances peuvent être utilisées pour faire des prédictions [2].
- **SlopeOne** prédit les évaluations des utilisateurs en calculant la différence moyenne de notation entre deux éléments sur la base des évaluations des autres utilisateurs. Pour prédire la note qu'un utilisateur u donnerait à un item i qu'il n'a pas encore noté, l'algorithme calcule une moyenne pondérée des déviations de i par rapport à tous les autres items que u a notés [5].
- **Non-negative Matrix Factorization** (NMF) : est un algorithme de factorisation matricielle très similaire à l'algorithme SVD. Il est basé sur la décomposition d'une matrice en deux matrices non négatives, ce qui peut être interprété comme une agrégation de caractéristiques positive. Chaque utilisateur et chaque item est représenté par un vecteur dans cet espace, et la note qu'un utilisateur donnerait à un item est prédite en prenant le produit scalaire de leurs vecteurs [6].
- **Bayesian Personalized Ranking** (BPR) est une fonction de coût qui s'utilise avec l'algorithme LightFM qui est un modèle de recommandation hybride à représentation latente. Lorsqu'il est utilisé avec la fonction de coût BPR, LightFM cherche à apprendre des rangs relatifs des items pour chaque utilisateur. Plus précisément, l'algorithme cherche à maximiser la probabilité que l'utilisateur préfère l'item qu'il a aimé à l'item qu'il n'a pas aimé [8].

2.2 Content-based

Le filtrage basé sur le contenu est une autre approche populaire pour les systèmes de recommandation. Il se concentre sur les caractéristiques des items eux-mêmes plutôt que sur les évaluations des utilisateurs. L'idée sous-jacente est que si un utilisateur a aimé un article particulier dans le passé, il est probable qu'il apprécie également les articles similaires en termes de contenu.

L'approche basée sur le contenu identifie les caractéristiques communes des éléments qui ont reçu une évaluation favorable d'un utilisateur, puis recommande à l'utilisateur de nouveaux éléments qui partagent ces caractéristiques. Dans les systèmes de recommandation basés sur le contenu, une information riche décrivant la nature de chaque élément est supposée être disponible sous la forme d'un vecteur de caractéristiques.

Fonctionnement Une fois que le profil d'intérêt de l'utilisateur est construit, les recommandations sont générées en cherchant des articles dont les caractéristiques correspondent le mieux aux préférences de l'utilisateur. Cela peut être fait en utilisant des techniques de

similarité telles que la similarité cosinus ou des algorithmes d'apprentissage automatique tels que les arbres de décision ou les réseaux de neurones [9].

Les systèmes de recommandation basés purement sur le contenu souffrent généralement des problèmes d'analyse de contenu limitée et de sur-spécialisation. L'analyse de contenu limitée se produit lorsque le système ne peut pas comprendre le contenu aussi bien qu'un humain le ferait. Par exemple, le système peut ne pas comprendre le contexte ou l'ironie. La sur-spécialisation se produit lorsque le système recommande des éléments trop similaires entre eux. Cela peut conduire à un manque de diversité dans les recommandations, ce qui peut être ennuyeux pour l'utilisateur
Cependant, l'approche basée sur le contenu présente plusieurs avantages. Elle permet une indépendance de l'utilisateur, une transparence et la capacité de recommander des éléments non encore évalués par aucun utilisateur. De plus, elle peut être utilisée pour prédire l'évaluation d'un utilisateur pour un nouvel élément, en construisant pour chaque valeur d'évaluation un profil de contenu qui est la moyenne des vecteurs de caractéristiques des éléments qui ont reçu cette valeur d'évaluation de l'utilisateur. La valeur d'évaluation prédite pour un élément est la valeur pour laquelle le profil de contenu est le plus similaire au vecteur de caractéristiques de l'élément

2.3 Défis

Les systèmes de recommandation sont devenus un outil essentiel pour aider les utilisateurs à gérer la grande quantité d'informations disponibles en ligne. Cependant, malgré leur popularité et leur utilité, ces systèmes sont confrontés à plusieurs défis majeurs qui peuvent entraver leur efficacité. Ces défis comprennent le démarrage à froid, la rareté des données et l'évolutivité [9], chacun nécessitant des approches et des solutions uniques pour être surmonté. Examinons ces défis plus en détail.

Démarrage à froid Le démarrage à froid est l'un des défis majeurs auxquels sont confrontés les systèmes de recommandation. Il se réfère à la difficulté de fournir des recommandations précises et pertinentes lorsqu'un nouvel utilisateur ou un nouvel élément est introduit dans le système sans historique d'interactions préalables. L'absence de données sur les préférences ou les caractéristiques de l'utilisateur limite la capacité du système à générer des recommandations personnalisées. De plus, lorsqu'un nouvel élément est ajouté, il peut être difficile de comprendre ses attributs et ses relations avec d'autres éléments existants. Les approches traditionnelles comme la recommandation collaborative ou la recommandation basée sur le contenu ne fonctionnent pas efficacement dans ces situations. Les chercheurs explorent donc des techniques telles que l'apprentissage actif, l'utilisation de méta-informations, ou encore l'analyse des réseaux sociaux pour surmonter le défi du démarrage à froid et améliorer la précision des recommandations pour les nouveaux utilisateurs et éléments. Afin de surmonter ce défi, des chercheurs ont tenté de combiner des techniques de recommandation pour construire des systèmes de recommandation hybrides (combinaison de filtrage collaboratif et content-based) [1].

Sparsity La rareté, également connue sous le nom de sparsity en anglais, se réfère au phénomène selon lequel la plupart des utilisateurs n'ont qu'une interaction limitée avec un petit nombre d'éléments parmi un vaste ensemble d'éléments disponibles. Cela se traduit par une matrice utilisateur-élément très clairsemée, où la majorité des valeurs sont

manquantes. La rareté pose un problème car elle rend difficile la découverte de relations et de modèles significatifs dans les données d'interaction. Les approches traditionnelles de filtrage collaboratif peuvent être affectées négativement par la rareté, car elles s'appuient sur la similarité entre les utilisateurs ou les éléments. Les méthodes de traitement des données clairsemées incluent l'imputation de valeurs manquantes, la régularisation pour réduire le sur-apprentissage, et l'utilisation de techniques telles que la factorisation matricielle et les modèles de recommandation basés sur les connaissances pour améliorer la qualité des recommandations malgré la rareté des données.

Evolutivité L'évolutivité, également connue sous le nom de scalability en anglais, est un défi crucial dans les systèmes de recommandation en raison de l'augmentation constante du volume de données et du nombre d'utilisateurs et d'éléments. À mesure que les systèmes de recommandation traitent des ensembles de données massifs, il devient essentiel de concevoir des solutions capables de gérer efficacement cette évolutivité. Les approches traditionnelles peuvent être confrontées à des problèmes de performances lorsqu'elles sont appliquées à grande échelle. Les temps de calcul augmentent, les coûts de stockage deviennent prohibitifs et la capacité de mettre à jour rapidement les recommandations en temps réel peut être compromise. Pour relever ce défi, des techniques telles que la parallélisation, la distribution de calcul, le traitement en flux continu et l'utilisation de systèmes de recommandation basés sur le cloud sont utilisées pour améliorer les performances et garantir une évolutivité optimale. De plus, les avancées dans les infrastructures de calcul distribué et les techniques d'apprentissage en ligne permettent de répondre aux besoins d'évolutivité des systèmes de recommandation dans des environnements de données volumineux et dynamiques. [11], parmi d'autres, ont tenté de surmonter ce défi en utilisant un algorithme qui s'appuie sur un petit modèle SVD précalculé et fournit des modèles SVD plus larges à l'aide de techniques peu coûteuses.

2.4 Le présent travail

Nous envisageons d'exploiter les systèmes de recommandation en utilisant des données collectées à partir d'un site web qui compile une vaste collection d'avis d'utilisateurs sur divers items. Plus précisément, nous nous concentrerons sur l'élaboration d'un modèle de recommandation conçu pour suggérer des items en fonction des préférences individuelles des utilisateurs, tout en tenant compte des jeux similaires qui ont été appréciés par d'autres joueurs ayant des intérêts similaires. En d'autres termes, notre objectif est de créer un système de recommandation qui non seulement comprend et répond aux goûts uniques de chaque joueur, mais qui est également capable de découvrir et de proposer de nouveaux jeux qui correspondent à leurs intérêts, en se basant sur les préférences de joueurs aux goûts similaires.

3 Prise en main des données

Les données que nous utilisons proviennent d'une base de données qui répertorie plus de 16 000 jeux de société et fournit des informations détaillées à leur sujet. Les utilisateurs ont la possibilité de noter les jeux et de laisser des commentaires. Pour notre analyse, nous avons accès à deux jeux de données : l'un contenant des informations détaillées sur les jeux, et l'autre contenant les avis des utilisateurs.

3.1 Prétraitement

Afin de mener au mieux nos expérimentations, nous avons commencé par prétraiter nos deux bases de données qui sont à l'état brut, nous nous sommes débarassés de certaines informations négligeables.

3.1.1 Base de données jeux

La base de données des jeux comprend les colonnes suivantes :

Nombre d'avis rectifiés	Note rectifiée	url	Note Finkel	Nombre d'avis

titre	casting	Note	categories	description	gameplay

Après une analyse plus approfondie des différentes colonnes de données, nous avons constaté qu'il existait des similitudes entre certaines d'entre elles. En conséquence, nous avons pris la décision d'examiner attentivement ces colonnes et d'apporter des modifications à la base de données en conséquence.

Titre et url : L'objectif de cette étape est de pouvoir identifier un jeu spécifique dans la base de données. Pour cela, nous avons décidé de tester la colonne `titre` comme identifiant, afin de déterminer si elle est effectivement unique pour chaque jeu.

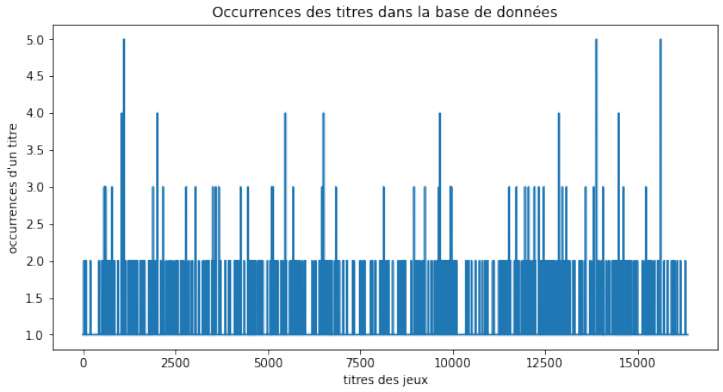


FIGURE 1 – Occurrences des titres dans la base de données

Après examen, nous avons constaté qu'il existe plusieurs jeux ayant le même titre. Cette duplication des titres s'explique par la présence de jeux avec des versions différentes. Ainsi, deux jeux ayant des versions distinctes peuvent partager un titre commun.

En regroupant les jeux par leur titre et en affichant l'`url` correspondante, on observe que la version du jeu est mentionnée dans ce dernier, tandis que le `titre` reste le même.

titre	url
Attila	https ://www.trictrac.net/jeu-de-societe/attila-2
Attila	https ://www.trictrac.net/jeu-de-societe/attila-1
Attila	https ://www.trictrac.net/jeu-de-societe/attila-0
Attila	https ://www.trictrac.net/jeu-de-societe/attila

TABLE 1 – Les différentes versions du jeu Attila

Suite à la constatation que l'`url` est unique pour chaque jeu, y compris pour différentes versions, nous avons pris la décision d'extraire le nom du jeu avec sa version directement à partir de l'`url` en utilisant des expressions régulières (regex). Le résultat de cette extraction sera stocké dans une nouvelle colonne nommée `_id`.

titre	_id
Attila	attila-2
Attila	attila-1
Attila	attila-0
Attila	attila

TABLE 2 – Les différentes versions du jeu Attila et leurs identifiants respectifs

Nombre d'avis et Notes : Étant donné que nous disposons de plusieurs colonnes contenant différents types de notes, nous avons pris la décision d'examiner de manière plus approfondie la signification.

— Note : correspond à la moyenne des notes attribuées par les utilisateurs à chaque jeu, représentant ainsi une mesure de la satisfaction globale des utilisateurs envers un jeu donné. Il convient de noter que les notes sont évaluées sur une échelle de 0 à 10, où 0 représente la plus basse satisfaction et 10 représente la plus haute satisfaction possible.

— Note rectifiée : est le résultat d'une mise à jour de la note initiale en prenant en compte les avis des utilisateurs qui ont été révisés ou corrigés

— Note Finkel : est calculée comme une combinaison linéaire de la note moyenne et du nombre d'avis d'un jeu, exprimée de la manière suivante :

Pour un jeu j

$$NoteFinkel_j = \mu_{Notes_j} + \gamma_j \times |Avis_j|$$

avec γ_j étant l'ajustement de Finkel pour un jeu donné, décrit ainsi $\gamma_j = \frac{\mu_{Notes_j} - 3}{100}$

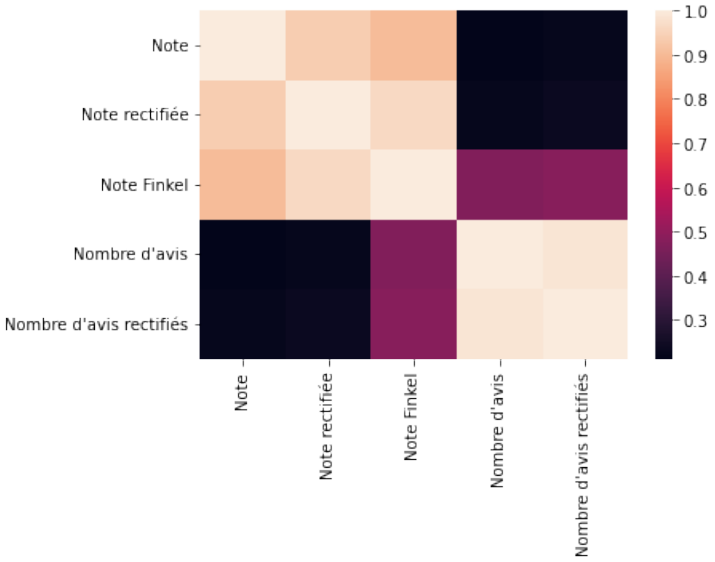


FIGURE 2 – Matrice de corrélation pour les colonnes Nombre d'avis et Notes

Après une analyse des corrélation des colonnes, il est apparu une forte corrélation entre les attributs tels que `Note`, `Note Finkel` et `Note rectifiée`. Par conséquent, dans le cadre de notre objectif de recommandation, nous avons décidé de conserver et d'utiliser uniquement la note originale, afin de simplifier le processus et d'éviter la redondance d'informations.

De manière similaire, nous avons également constaté une corrélation entre le `Nombre d'avis` et le `Nombre d'avis rectifiés`. Dans ce cas, nous avons décidé de ne prendre en compte que le nombre d'avis dans notre évaluation.

Description : La colonne `Description` contient les descriptions de chaque jeu. Voici un exemple pour illustrer son contenu :

	description
0	Marvel United : Unis contre les Super-Vilains ...
7	1962 - La guerre froide continue alors qu'un n...
7570	Goodies
10912	Aucune description

TABLE 3 – Exemples de descriptions de jeux

Dans une première étape, nous avons supprimé les descriptions qui étaient indiquées comme "Aucune description". Cela a impacté un total de 1857 jeux dans la base de données.

Par la suite, nous avons décidé d'analyser la langue dans laquelle les descriptions sont rédigées. Cette information nous sera utile pour faciliter la lemmatisation et la sélection des stop words dans le traitement du texte.

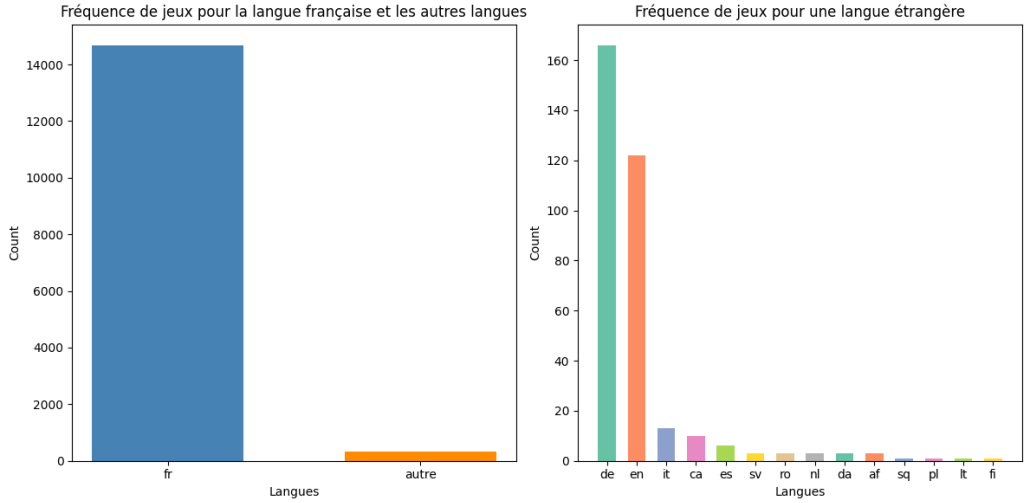


FIGURE 3 – Répartition des jeux selon la langue des descriptions : français (fr) et autres langues (autre)

Nous avons utilisé un détecteur de langue qui a identifié la présence de moins de 500 jeux dont la description n'était pas rédigée en français. Parmi ceux-ci, nous avons détecté 14 langues différentes. Pour confirmer qu'il s'agit effectivement de langues étrangères, nous avons entrepris des vérifié manuellement.

	Description
15	Une carte Goddie Promotionnelle proposant 1 Leader "Greg & Emmy"
29	Devenez un marchand riche.
246	Projet Kickstarter 2015
666	Planche bonus pour Heroes Of Normandie : les soldats Siegfried et Bill Furlong

TABLE 4 – Exemples de descriptions avec les langues étrangères

Nous avons finalement constaté que ces descriptions étaient en français. Notre détecteur de langue a mal prédit la langue en raison de la présence de descriptions courtes contenant des noms propres.

Enfin, nous avons effectué un processus de nettoyage sur les descriptions, qui comprenait les étapes suivantes :

— Lemmatisation : réduction des mots à leur forme de base pour faciliter l'analyse.

— Suppression des stopwords : élimination des mots courants tels que "le", "la", "et", etc., qui ne contribuent pas significativement à l'analyse.

— Suppression de la ponctuation, des chiffres et des caractères spéciaux : nous avons éliminé la ponctuation telle que les points, les virgules ... ainsi que les chiffres et les caractères spéciaux présents dans les descriptions.

Les étapes de nettoyage ont été essentielles pour préparer les descriptions en vue d'une analyse précise et pertinente. Elles ont permis de supprimer les éléments indésirables et de se concentrer sur le contenu textuel essentiel.

Casting : L'attribut `Casting` dans sa forme actuelle regroupe les différentes personnes et entités impliquées dans la création d'un jeu, y compris les créateurs, les illustrateurs, les éditeurs et les distributeurs. Cependant, il est important de noter que ces informations sont actuellement mal formatées, étant toutes regroupées dans une seule chaîne de caractères.

casting
Par Eric Lang et Andrea ChiarvesioIllustré par Édouard GuitonÉdité par CMON Limited
Par Elizabeth HargraveIllustré par Matt Paquette et Indi MaverickÉdité par Gigamic

Table 5 – Exemples de castings de jeux avant nettoyage

Nous avons pris la décision de subdiviser l'attribut en quatre colonnes distinctes, afin de mieux organiser les informations. Chaque colonne représente une catégorie d'informations, notamment les créateurs, les illustrateurs, les éditeurs et les distributeurs.

creators	illustrators	editors	distributors
Eric Lang et Andrea Chiarvesio	Édouard Guiton	CMON Limited	NaN
Elizabeth Hargrave	Matt Paquette et Indi Maverick	Gigamic	NaN

Table 6 – Le castings de jeux après nettoyage

Gameplay L'attribut `Gameplay` regroupe des informations concernant le nombre de joueurs, la durée moyenne d'une partie et l'âge minimal requis pour jouer au jeu. Cependant, tout comme l'attribut `Casting`, il présente le même problème de formatage. Par conséquent, il nécessitera un traitement similaire pour le restructurer de manière adéquate.

gameplay
1 à 4|14 ans et +|45
2 à 5|14 ans et +|60

Table 7 – Exemples de gameplay des jeux avant nettoyage

Nous avons aussi discrétisé l'âge en 3 catégories `enfant`, `ado` et `adulte` en fonction des intervalles d'âge d'un jeu, et également le nombre de joueurs en `solo`, `duo` et `multi`.

duration	enfant	ado	adulte	solo	duo	multi
45.0	0	1	1	1	0	1
60.0	0	1	1	0	1	1

Table 8 – Les gameplays des jeux après traitement

Nous avons pris la décision de permettre aux adultes de jouer à des jeux destinés aux adolescents ou aux enfants, ainsi qu'aux adolescents de jouer à des jeux destinés aux enfants, mais pas l'inverse !

Categories La colonne `Categories` contient les catégories auxquelles chaque jeu appartient. Voici un exemple illustrant le contenu de cette colonne :

	categories
1	NaN
2	Enchères|Aventures dans la jungle|Collection / famille|Exploration|Placement d'ouvrier

TABLE 9 – Exemples de catégories des jeux avant nettoyage

Nous constatons que certaines entrées de la colonne `Categories` sont nulles, et que chaque jeu peut appartenir à plusieurs catégories. Par conséquent, nous allons procéder à une séparation appropriée des différentes catégories pour chaque jeu.

Pour cela, nous allons représenter les catégories d'un jeu sous forme de listes. Ensuite, nous appliquerons le même processus de traitement de texte que celui utilisé pour les descriptions, pour chaque catégorie individuellement. Ainsi, nous obtiendrons des résultats spécifiques à chaque catégorie. Voici le résultat obtenu :

	categories
1	NaN
2	[aventure jungle, collection famille, enchere, exploration, placement ouvrier]

TABLE 10 – Exemples de catégories des jeux après nettoyage

3.1.2 Base de données avis

La base de données des avis comprend les colonnes suivantes :

title	note	author	url	comment	review_href

date_published	title_review

Après avoir examiné les différentes colonnes de la base de données, nous avons constaté que la colonne `review_href` ne nous est pas utile, car elle contient uniquement les liens vers les commentaires, ce qui ne sera pas nécessaire pour notre objectif. De plus, nous avons identifié les colonnes nécessitant un prétraitement spécifique, à savoir :

Url : La colonne `url` contient les liens vers les jeux commentés, et nous allons l'utiliser pour obtenir un identifiant unique pour chaque jeu. Cela nous permettra de référencer directement le jeu commenté, ce qui sera utile pour la partie recommandation. Nous appliquerons donc le même prétraitement que celui fait a la base données jeux.

Comment La colonne `comment` contient les commentaires rédigés par les utilisateurs pour chaque jeu. Ces commentaires sont en français et fournissent des informations sur leur expérience avec le jeu. Voici un exemple pour illustrer le contenu de cette colonne :

	comment
0	Lorsque le jeu est jeu, bon, réflexif, joli pour qui est sensible à ce style d'illustration...
1	Comment continuer après un mega hit ? Simplement. Après les oiseaux, les papillons...
2	Vin d'jeu : Avec Mariposas, Elizabeth Hargrave parvient à « simuler » cette épopée...

Table 11 – Exemples de commentaire avant nettoyage

Nous remarquons qu'il est nécessaire d'effectuer un traitement NLP sur chaque commentaire. Pour cela, nous utiliserons le même processus de prétraitement que celui appliqué aux descriptions des jeux. Ce processus comprendra la lemmatisation, la suppression des (stop words), des chiffres et de la ponctuation.

Date_published La colonne représente la date de publication d'un commentaire, au format (YYYY-MM-DD HH-MM-SS), comme illustré dans le tableau ci-dessous :

	date_published
0	2021-01-27 11 :06 :44
1	2020-10-18 10 :04 :21
2	2021-02-01 08 :35 :08

Table 12 – Exemples de Date_published des commentaire

Pour cette colonne, nous avons décidé de conserver uniquement l'année de publication des commentaires. Les autres détails tels que le mois, le jour, l'heure et les secondes ne sont pas considérés comme pertinents pour notre analyse.

4 Statistiques

Cette section présente les statistiques relatives aux bases de données sur les jeux et les avis des utilisateurs. Ces données sont essentielles pour comprendre les tendances des utilisateurs. Nous examinerons en détail la distribution des avis des utilisateurs et les schémas de notation afin d'obtenir des informations pertinentes.

4.1 Jeux

	Nombre d'avis	Note	duration
count	16873.00	16873.00	14134.00
mean	13.31	4.59	51.92
std	52.06	3.67	36.33
min	0.00	0.00	1.00
25%	0.00	0.00	20.00
50%	1.00	6.00	45.00
75%	7.00	7.92	60.00
max	1452.00	10.00	120.00

TABLE 13 – Statistiques sur les données quantitatives des jeux

Nombre d'avis La figure illustrant la répartition des avis révèle clairement un pic marqué à la valeur zéro, indiquant un nombre significatif de jeux n'ayant reçu aucun avis. Cela suggère qu'une majorité de jeux a obtenu un nombre limité d'avis, tandis que relativement peu de jeux ont réussi à accumuler un grand nombre d'avis. De plus, le tableau et la figure ci-dessous démontrent qu'un grand nombre de jeux n'ont reçu aucun avis.

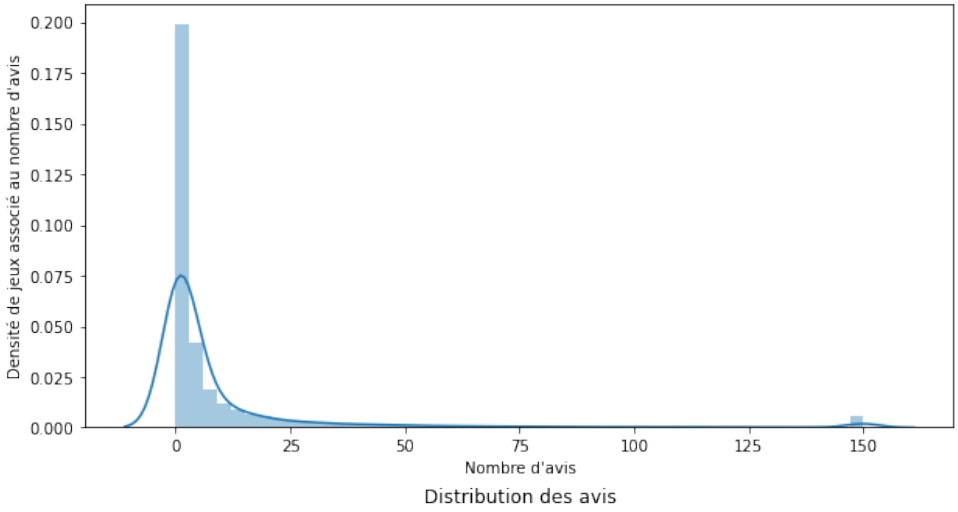


FIGURE 4 – Répartition du nombre d'avis

Notes par jeu Lorsque nous examinons la répartition des notes par jeu, nous constatons que 35% des jeux ont une note nulle, ce qui nous semble être un pourcentage considérable. Par conséquent, nous formulons l'hypothèse selon laquelle lorsqu'un jeu ne reçoit aucune note, il se voit automatiquement attribuer la note zéro.

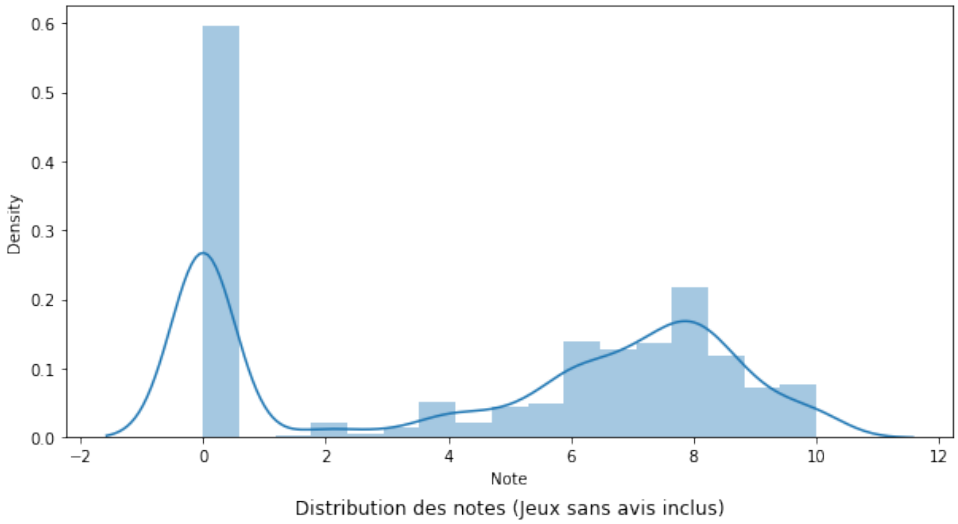


FIGURE 5 – Répartition des notes par jeu

De plus, comme précédemment observé dans la Figure 4, la plupart des jeux ont très peu d'avis, voire aucun. Afin d'évaluer la proportion des jeux sans avis, nous avons tracé la Figure 6 ci-dessous :

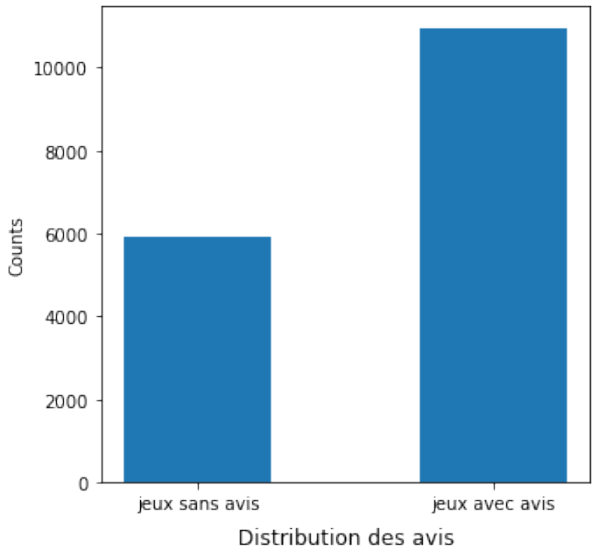
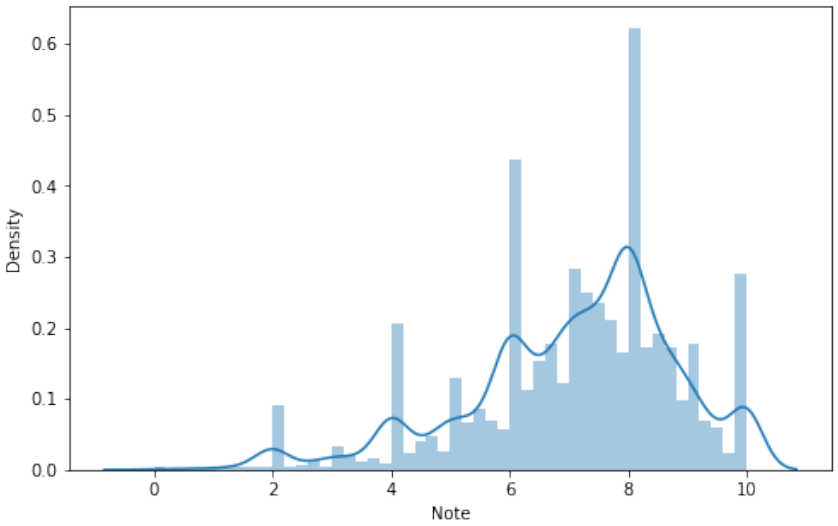


FIGURE 6 – Nombre de jeux sans et avec avis

Nous trouvons le même pourcentage de jeux sans avis que de jeux avec une note nulle : 35%, voici donc la vraie distribution des notes des jeux avec avis cette fois, ainsi que les statistiques correspondantes :

	Note
count	10936.00
mean	7.07
std	1.79
min	0.00
25%	6.00
50%	7.36
75%	8.20
max	10.00



Distribution des notes des jeux (avec avis)

4.2 Avis

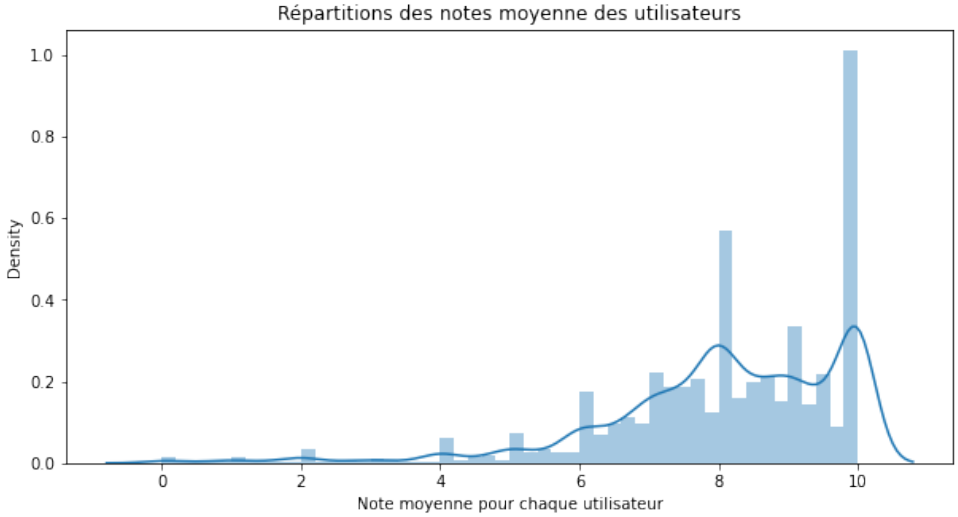


FIGURE 9 – Répartition des notes moyennes données par utilisateur

Notes par utilisateur L'analyse de la figure de répartition des notes moyennes attribuées par les utilisateurs met en évidence deux modes distincts. Le premier mode se situe autour de la valeur 8, tandis que le deuxième mode est observé à la note maximale de 10. On observe une tendance vers des évaluations positives dans l'ensemble. Il n'existe pas vraiment de variété d'opinions et d'appréciations, les notes sont relativement élevées.

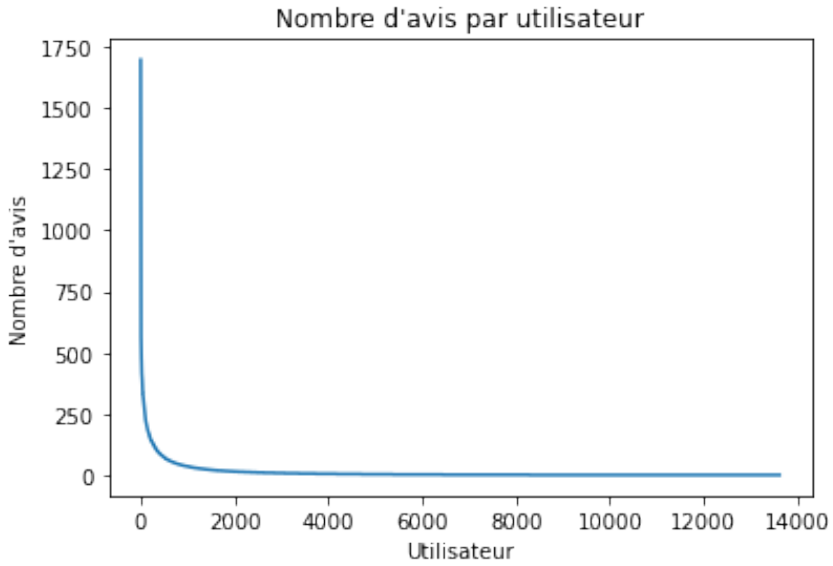


FIGURE 10 – Courbe du nombre d'avis par utilisateur

Avis par utilisateur La distribution des avis par utilisateur peut être décrite par une loi de Zipf, comme illustré dans la Figure 10 ci-dessus. En d'autres termes, un petit nombre d'utilisateurs a donné un grand nombre d'avis, tandis que la majorité des utilisateurs en a donné très peu.

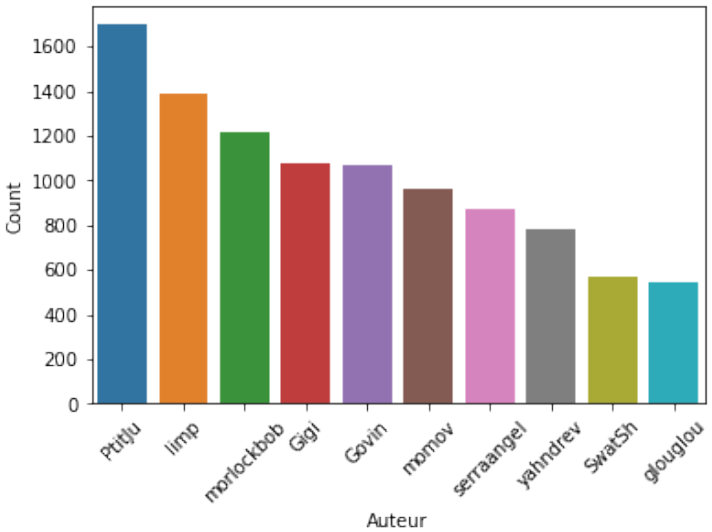


FIGURE 11 – Nombre d'avis des 10 utilisateurs les plus actifs

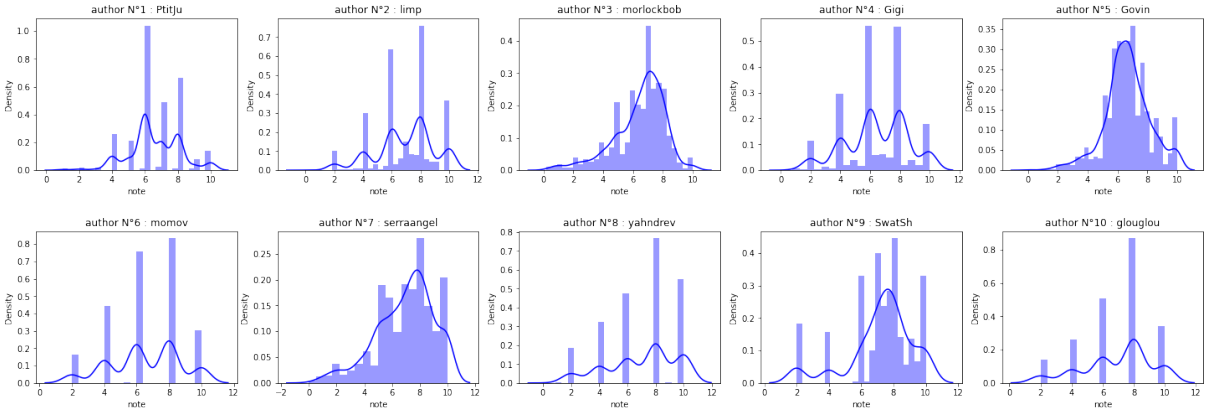


FIGURE 12 – Répartition des notes données par les 10 utilisateurs les plus actifs

Nous avons identifié les 10 joueurs les plus actifs, et nous avons observé que leurs comportements étaient assez distincts (voir Figures 11 et 12). Certains joueurs ont tendance à donner des notes principalement dans une plage spécifique, ce qui se traduit par un seul mode dans la fonction de densité. En revanche, pour d'autres joueurs, nous avons constaté une plus grande variabilité dans leurs évaluations, ce qui se reflète par la présence de plusieurs modes dans la fonction de densité. Ces observations mettent en évidence la diversité des préférences et des critères d'évaluation des joueurs les plus actifs.

FIGURE 13 – Wordcloud des avis en fonction de la note

Avis La Figure 13 présente les bigrammes les plus fréquents dans les avis des utilisateurs en fonction de la note attribuée. On observe que les expressions "très bien" et "très bon"

reviennent fréquemment, ce qui indique une satisfaction générale des utilisateurs. Cependant, il est surprenant de constater que l'expression "très bien" est également présente dans les avis accompagnés d'une note de 4. Cette observation souligne la complexité de l'analyse des avis, où certains utilisateurs peuvent utiliser des expressions positives même s'ils donnent une note relativement basse. Un autre point d'intérêt est la différence de vocabulaire en fonction des notes attribuées. Les mots utilisés dans les avis varient considérablement pour les notes allant de 0 à 2 ou de 6 à 10, ce qui suggère une divergence dans les expériences et les opinions des utilisateurs.

4.3 Bilan

Après avoir examiné en détail les statistiques relatives à la distribution des notes par jeu et par utilisateur, ainsi que la distribution du nombre d'avis par jeu, nous avons constaté certaines tendances intéressantes. Tout d'abord, il est apparu que les utilisateurs ont généralement tendance à donner des notes élevées. Cependant, nous avons également observé des différences dans les comportements d'évaluation des utilisateurs. Certains joueurs ont tendance à attribuer des notes dans une plage spécifique, tandis que d'autres présentent une plus grande variabilité dans leurs évaluations. Ces résultats statistiques nous fournissent une base solide pour passer à la partie suivante qui abordera les recommandations.

5 Recommandation

Dans le cadre de notre système de recommandation, nous avons choisi d'utiliser deux approches principales : le *filtrage collaboratif* et l'approche *content based*. Ces deux méthodes nous permettent d'offrir des recommandations personnalisées aux utilisateurs en exploitant différentes sources d'information.

Nous évaluerons l'efficacité avec des métriques adaptées à ce genre de problème, par exemple la MRR et la nDCG. Comme c'est un probleme non supervisée,

5.1 Filtrage collaboratif

Le filtrage collaboratif est une méthode utilisée dans les systèmes de recommandation pour prédire les préférences d'un utilisateur en se basant sur les données collectives de plusieurs utilisateurs. Le principe sous-jacent est que si deux utilisateurs ont des préférences similaires sur certains éléments, ils sont susceptibles d'avoir des préférences similaires sur d'autres éléments également.

Le filtrage collaboratif fonctionne en utilisant les notations et les interactions des utilisateurs avec les éléments pour calculer des similarités entre les utilisateurs ou entre les éléments. Ces similarités sont ensuite utilisées pour prédire les préférences manquantes d'un utilisateur.

Pour mettre en œuvre cette approche, nous aurons besoin des informations sur les préférences des utilisateurs on utilisera les notes des utilisateurs pour chaque avis.

5.1.1 Prétraitement & statistiques

Afin de procéder à la recommandation en utilisant l'approche filtrage collaboratif nous avons filtré les données pour ne conserver que les jeux qui ont reçu plus de 8 avis et les utilisateurs qui ont mis plus de 8 avis. Ensuite, il est nécessaire d'itérer jusqu'à la stabilisation du nombre d'avis par utilisateur et par item. En effet, dans le filtrage collaboratif, les seuils sont souvent utilisés pour filtrer les données afin de supprimer les avis ou les utilisateurs qui ont un nombre insuffisant d'avis. En supprimant les avis ou les utilisateurs qui ont moins d'avis que le seuil spécifié, on peut réduire les risques d'incohérence ou de recommandations peu fiables basées sur un petit nombre d'avis. Par exemple, si un utilisateur n'a donné qu'un seul avis positif, cela ne signifie pas nécessairement que tous les éléments similaires doivent être fortement recommandés. On réduit ainsi le bruit et la variabilité des avis en se concentrant sur des utilisateurs ou des avis plus établis et plus représentatifs.

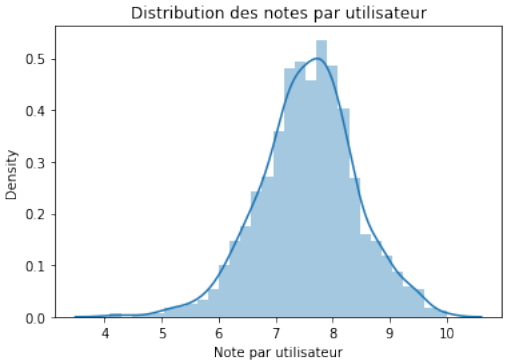


FIGURE 14 – Répartition des notes par utilisateur

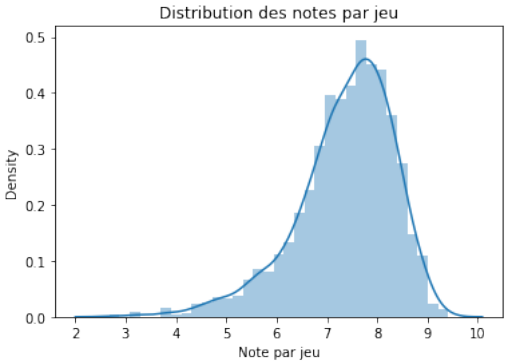


FIGURE 15 – Répartition des notes par jeu

Les figures présentées ci-dessus illustrent la distribution des notes par utilisateur et par jeu après tri. Il est clair que la répartition est beaucoup plus uniforme, avec un seul mode dominant. Cependant, cette faible variabilité des notes peut potentiellement représenter un inconvénient pour le filtrage collaboratif. En effet, dans le filtrage collaboratif, il est essentiel de trouver des utilisateurs qui partagent des préférences similaires afin de recommander des jeux pertinents. Si les notes sont très similaires et peu variées, il peut être plus difficile de détecter des correspondances significatives entre les utilisateurs. Cela pourrait limiter la précision des recommandations basées sur le filtrage collaboratif et nécessiter des ajustements supplémentaires pour compenser cette faible variabilité des notes.

	Avant tri		**Après tri**	
	avis/user	avis/jeu	avis/user	avis/jeu
Mean	12.92	16.44	43.17	37.27
1er quantile	1.0	2.0	11.0	12.0
2e quantile	2.0	4.0	19.0	20.0
3e quantile	7.0	15.0	42.0	45.0
Min	1	1	8	8
Max	1688	172	1289	158
Std	46.05	31.03	73.00	38.50
Nb avis	176071		134268	
Nb users	13623		3110	
Nb jeux	10709		3603	

TABLE 14 – Description des données avant et après tri

Le tableau fournit une comparaison entre les données avant et après tri pour le filtrage collaboratif. Ces données décrivent les changements dans la répartition des avis par utilisateur et par jeu après l'application du tri dans le cadre du filtrage collaboratif. Les résultats montrent une augmentation significative du nombre moyen d'avis par utilisateur et par jeu après le tri, ainsi qu'une augmentation des quantiles et une diminution de l'écart type. Cela suggère une concentration plus importante des avis sur un nombre réduit d'utilisateurs et de jeux, ce qui peut potentiellement améliorer l'efficacité des recommandations basées sur le filtrage collaboratif.

5.1.2 Méthode

Nous avons testé différents algorithmes de recommandation issus de la bibliothèque Surprise. Surprise est une bibliothèque en Python conçue pour faciliter le développement et l'évaluation d'algorithmes de filtrage collaboratif. Elle fournit des outils et des modèles préimplémentés pour effectuer des recommandations basées sur des données de notion collaborative telles que les systèmes de recommandation utilisés par des plateformes comme Netflix, Amazon et Spotify. Voici les différents algorithmes que nous avons utilisés ont été présentés dans la section 2.1 et sont les suivants : Baseline algorithm, SVD et KNN Basic.

Pour entraîner et évaluer nos modèles, nous avons divisé la base de données en un ensemble d'apprentissage et un ensemble de validation. Étant donné qu'il s'agit d'un problème non supervisé, nous avons dû adopter une approche différente en cachant une partie des données aux modèles et en les entraînant sur le reste. Cette méthode est couramment utilisée dans les problèmes de recommandation.

En l'occurrence, nous avons caché les avis des utilisateurs dont la note était supérieure à 8, en les considérant comme leurs jeux préférés. Notre objectif était de vérifier si nos modèles étaient capables de les recommander avec succès.

5.1.3 Expérimentations

Dans cette section, notre objectif est de rechercher les paramètres optimaux pour chaque modèle afin de pouvoir les comparer ultérieurement. Nous utiliserons la métrique de l'erreur quadratique moyenne (MSE) pour évaluer les performances des modèles, nous comparons nos algorithmes avec un modèle de référence appelé "Modèle Naïf", qui prédit la moyenne des notes dans l'ensemble d'entraînement.

Nous appliquons une validation croisée à chaque algorithme et calculons la moyenne des performances sur les ensembles d'entraînement et de test.

SVD Pour l'algorithme SVD, nous avons choisi de modifier les paramètres pour optimiser les performances du modèle. Nous avons notamment ajusté le terme de régularisation, noté λ (`reg_all`), ainsi que `n_factors` qui représente le nombre de facteurs latents utilisés pour décomposer la matrice de notation. La valeur de `n_factors` influence la taille des matrices de facteurs latents et peut jouer un rôle crucial dans la capacité du modèle à capturer les relations complexes entre les utilisateurs et les jeux. En ajustant ces paramètres, nous avons cherché à trouver le bon équilibre entre la capacité de généralisation du modèle et sa complexité.

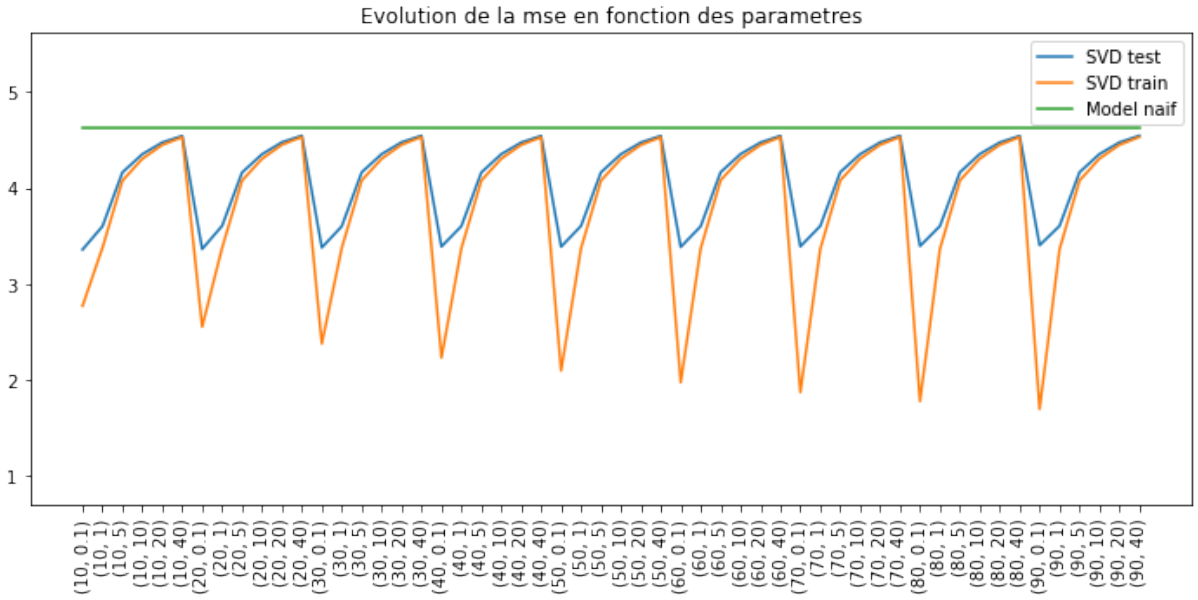


FIGURE 16 – courbe de variation des paramètre

L'analyse de la figure révèle des observations intéressantes concernant l'effet de la dimension latente et de la régularisation sur la MSE.

D'abord, en augmentant la dimension latente, la MSE pour l'ensemble de test reste relativement stable tandis que celle de l'ensemble d'entraînement diminue. Cela indique que l'ajout de dimensions latentes permet au modèle de mieux ajuster les données d'entraînement, mais entraîne un surapprentissage où le modèle devient trop spécifique aux données d'entraînement et perd en capacité de généralisation. Par la suite, nous avons préféré opter pour une valeur pas trop grande pour `n_factors`, avec 30 dimensions.

Ensuite, nous avons examiné l'influence de la régularisation sur la MSE. À mesure que la valeur de régularisation augmente, l'écart entre les valeurs de MSE pour l'ensemble d'entraînement et l'ensemble de test diminue. À un certain seuil, généralement autour d'une valeur de régularisation de 1, les courbes de MSE pour les deux ensembles commencent à se rapprocher.

Pour mieux illustrer ces résultats, nous allons comparer l'intensité de la MSE sur l'ensemble de test à la différence entre la MSE sur l'ensemble d'entraînement et celle sur l'ensemble de test en fonction des paramètres.

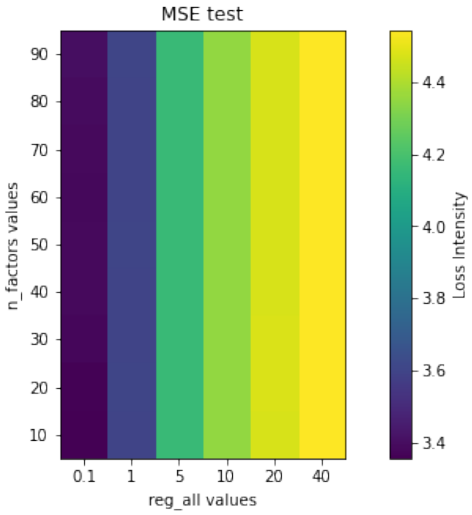


FIGURE 17 – Intensité de la MSE en test en fonction des paramètres

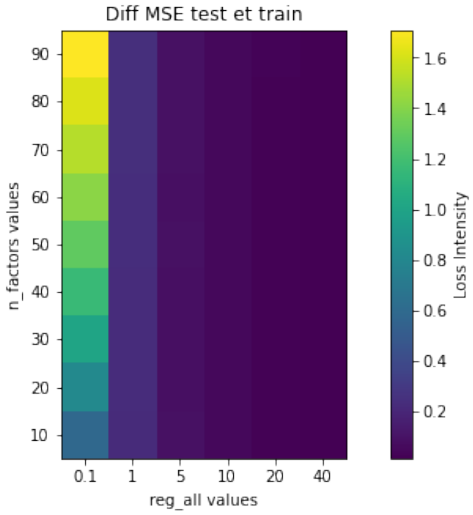


FIGURE 18 – Différence de la MSE en test et en train en fonction des paramètres

On observe que pour des valeurs faibles de régularisation, c'est-à-dire inférieures à 1, les valeurs de MSE sont petites. Cependant, la différence entre la MSE sur l'ensemble de test et celle sur l'ensemble d'entraînement est grande, ce qui indique une possibilité de surapprentissage du modèle. En revanche, pour des valeurs élevées de régularisation, la MSE augmente et la différence entre la MSE sur l'ensemble de test et celle sur l'ensemble d'entraînement diminue.

Nous proposons donc de combiner les deux résultats afin de déterminer la plage optimale des paramètres. Pour ce faire, nous utilisons la formule suivante :

$$min\ (\alpha \cdot mse_{test} + \beta \cdot |mse_{train} - mse_{test}|)$$

Le paramètre α représente le taux de prise en compte de l'erreur du modèle. Plus sa valeur est élevée, plus nous cherchons à minimiser l'erreur sur l'ensemble de test. Ainsi, en augmentant α, nous accordons une plus grande importance à la performance du modèle sur les données de test.

Il convient de noter que le paramètre β représente la pondération de la différence entre les erreurs sur l'ensemble d'entraînement et l'ensemble de test. Une valeur élevée de β indique que nous attachons une plus grande importance à la réduction de cette différence, ce qui peut être un indicateur de la capacité du modèle à généraliser aux nouvelles données et éviter le surapprentissage.

On obtient avec $\alpha = 0.5$ et $\beta = 1$ la heatmap suivante :

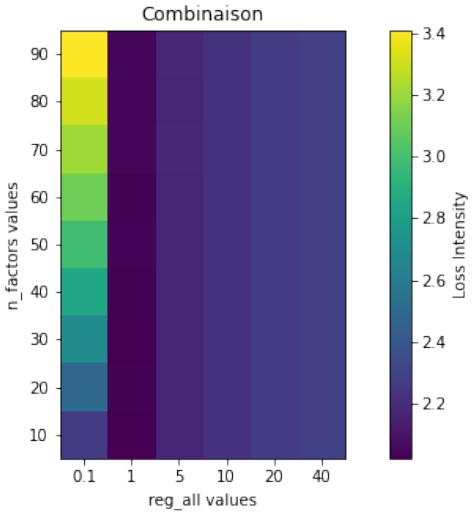


FIGURE 19 – Combinaison des deux intensités

La heatmap confirme que les valeurs optimales de reg_all se situent dans le voisinage de 1, ce qui confirme les résultats précédents basés sur les courbes de la MSE. Par conséquent, nous choisissons de fixer la valeur optimale de reg_all à 1.

Knn Pour l'algorithme KNN, nous avons un seul paramètre à prendre en compte, qui est le nombre de voisins (ou clusters) à considérer. Pour étudier l'effet de ce paramètre sur les performances de l'algorithme, nous allons le faire varier en testant plusieurs valeurs différentes.

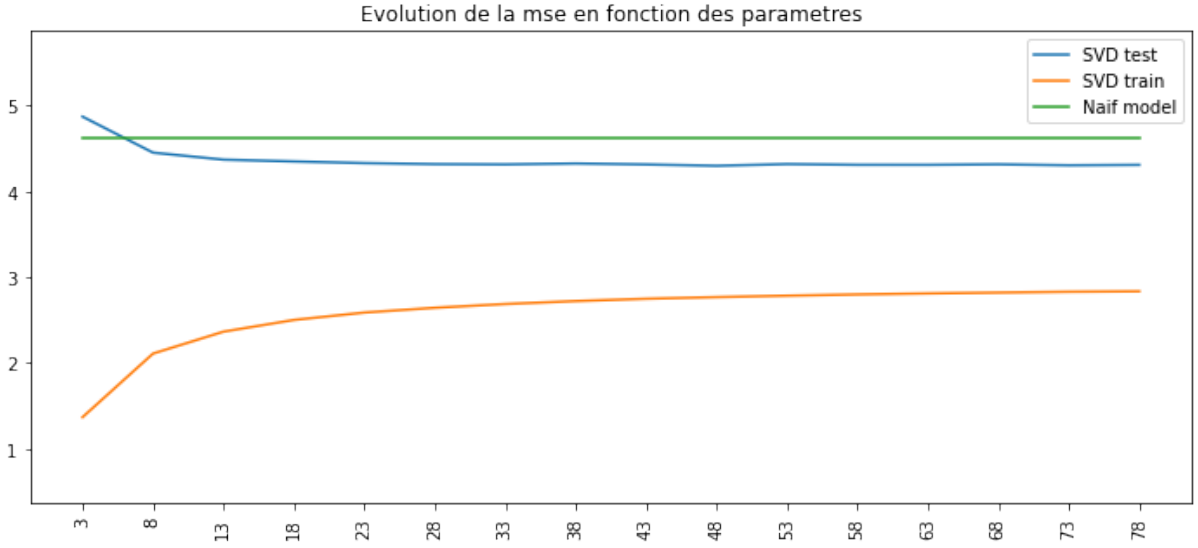


FIGURE 20 – courbe de la mse en variant du nombre de clusters

Après avoir tracé la courbe de la MSE en variant le nombre de clusters pour l'algorithme KNN, nous avons observé que les performances du modèle ne s'amélioraient pas significativement au-delà de 18 clusters. Par conséquent, nous avons déterminé que le nombre optimal de clusters pour notre modèle KNN était de 18.

Baseline Nous avons décidé de faire varier les deux paramètres de régularisation pour le modèle Baseline, qui sont reg_u et reg_i, correspondant respectivement à λ_2 et λ_3 des biais du modèle.

$$b_i = \frac{\sum_{u:(u,i)\in\mathcal{K}}(r_{ui} - \mu)}{\lambda_2 + |\{u|(u,i) \in \mathcal{K}\}|}$$
$$b_u = \frac{\sum_{i:(u,i)\in\mathcal{K}}(r_{ui} - \mu - b_i)}{\lambda_3 + |\{i|(u,i) \in \mathcal{K}\}|}$$

et avons obtenu les résultats suivants :

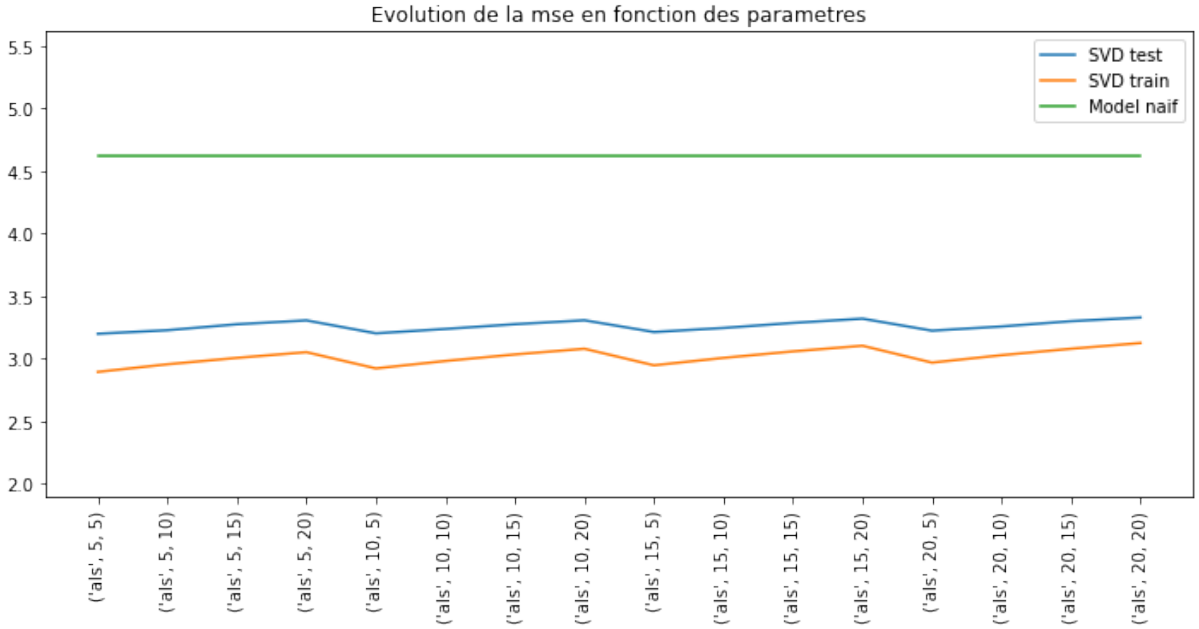


FIGURE 21 – Courbes de la mse en variant les deux paramètres de la Baseline

Après avoir fait varier les deux paramètres de l'algorithme Baseline, nous avons identifié les valeurs optimales qui donnent les meilleurs résultats. Nous avons trouvé que la valeur optimale pour le paramètre reg_u était de 25, tandis que la valeur optimale pour le paramètre reg_i était de 5.

5.1.4 Résultats

Dans cette section, notre objectif est de comparer les différents modèles en utilisant deux métriques différentes : la MSE et la MRR qui est plus adaptée pour les systèmes de recommandation.

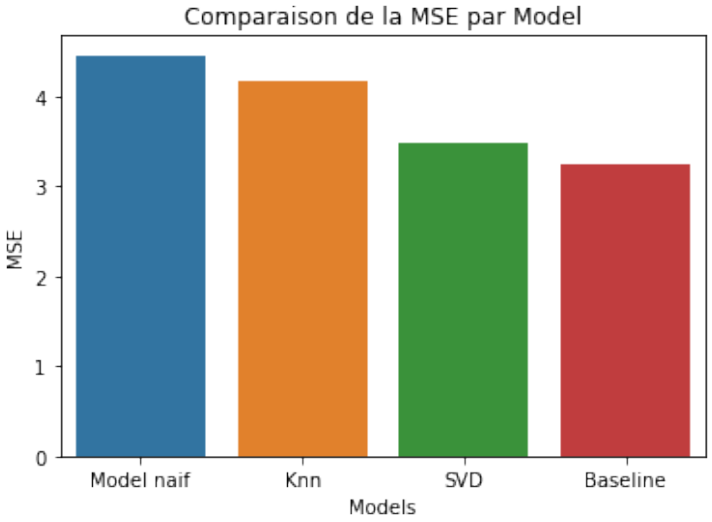


FIGURE 22 – MSE des modèles avec leurs paramètres optimaux

Nous observons que, en termes de MSE, les erreurs de nos modèles sont inférieures à celles du modèle de référence, qui est le modèle naïf. Cette constatation indique une amélioration significative par rapport au modèle de référence en ce qui concerne la précision des prédictions.

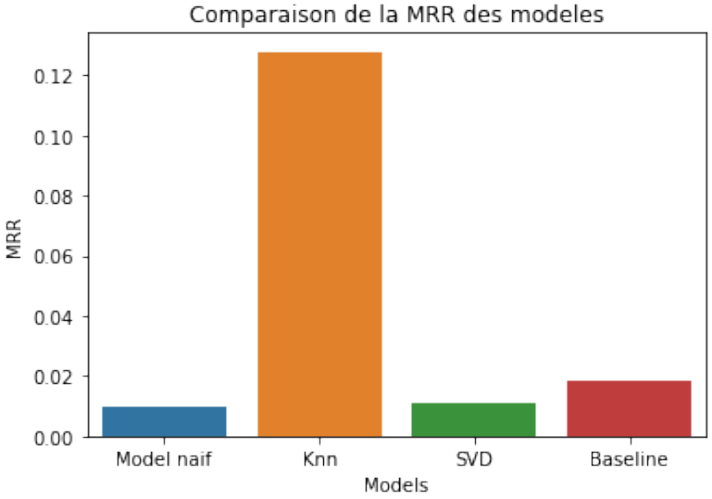


FIGURE 23 – MRR des modèles avec leur paramètre optimaux

De manière similaire, en termes de MRR, nos modèles ont surpassé l'algorithme de référence, en particulier le KNN qui a obtenu un MRR de 0.12. Cela signifie qu'en moyenne, il nécessite seulement 8 recommandations pour que l'utilisateur trouve un jeu qu'il apprécie, tandis que l'algorithme naïf nécessite 100 recommandations avant de tomber sur un jeu apprécié par l'utilisateur.

5.1.5 Discussion

L'algorithme baseline est celui qui minimise le plus la MSE. Il est généralement basé sur des calculs simples, tels que des moyennes, des médianes ou des approches similaires

pour estimer les évaluations manquantes ou prédire les évaluations futures. En raison de sa simplicité, il peut être moins sensible aux bruits ou aux erreurs dans les données, ce qui peut conduire à de meilleurs résultats en termes de MSE.

Le KNN et le SVD sont des algorithmes plus complexes qui peuvent nécessiter une plus grande quantité de données pour fournir des recommandations précises. Ils peuvent être sensibles à la densité des données, à la sparsité et à d'autres caractéristiques spécifiques du jeu de données. Si les données sont limitées ou si elles ne sont pas bien adaptées à ces algorithmes, cela peut entraîner des résultats moins performants en termes de MSE.

Le MRR est une mesure qui se concentre sur le classement des éléments recommandés. L'algorithme KNN peut être plus performant en termes de MRR car il peut mieux classer les éléments recommandés par rapport à l'algorithme baseline et à l'algorithme naïf. Cela peut être dû à sa capacité à capturer les relations de similarité entre les utilisateurs et à recommander des éléments plus pertinents en termes de goûts et de préférences de l'utilisateur cible.

5.2 Content Based

Le filtrage basé sur le contenu est une approche utilisée dans les systèmes de recommandation. Il se concentre sur les caractéristiques intrinsèques des éléments eux-mêmes, telles que la description, les catégories, les genres, les acteurs, etc. L'objectif est de trouver des éléments similaires à ceux que l'utilisateur a déjà appréciés.

Le fonctionnement du filtrage basé sur le contenu consiste à analyser les caractéristiques des éléments existants pour créer des profils ou des représentations. Ces profils sont ensuite comparés avec les préférences de l'utilisateur pour identifier les éléments qui correspondent à ses goûts. Par exemple, si un utilisateur a aimé des jeux de stratégie, le système de recommandation peut proposer d'autres jeux de stratégie similaires en se basant sur les caractéristiques telles que le genre, la description, etc.

Pour mettre en place le filtrage basé sur le contenu, les colonnes requises dépendent des caractéristiques spécifiques des éléments. Dans notre cas, les colonnes telles que la description, les catégories, etc. sont nécessaires pour analyser les caractéristiques des jeux et créer des profils.

5.2.1 Prétraitement & statistiques

Catégories Après analyse, nous avons remarqué que nous disposons d'un total de 172 catégories différentes, ce qui représente un nombre considérable. Cette observation nous pousse à examiner plus en détail cette situation.

	categories
1	[enchere, enchere poing ferme, enchere cachee, enchere anglais, enchere japonais]
2	[carte, dé, hasard dé carte]

Table 15 – Exemples de catégories qui sont similaires

Après avoir analysé nos catégories, nous avons observé qu'il était possible de les regrouper en fonction de leur similarité syntaxique. Par exemple, la première catégorie du tableau pourrait être regroupée sous le terme "enchère", tandis que la deuxième ligne en : "carte" et "dé".

Afin d'effectuer le regroupement syntaxique des catégories, nous avons utilisé l'algorithme des KNN pour effectuer une clusterisation. Nous avons choisi de fixer manuellement le nombre de clusters à 155.

Voici les clusters obtenus :

	Cluster
cluster 1	[enchere, enchere poing ferme, enchere cachee, enchere anglais, enchere japonais]
cluster 2	[carte, dé, enchere, hasard dé carte]
cluster 3	[placement, placement ouvrier]
cluster 4	[cambriolage]

TABLE 16 – Exemples de cluster sur les catégories

Ensuite, nous avons nommé chaque cluster en se basant sur le mot le plus fréquent dans chaque cluster. Dans le cas où plusieurs mots avaient la même fréquence, nous les avons inclus tous les deux dans le nom du cluster. Par exemple, le premier cluster a été nommé "enchère", le deuxième cluster "dé carte", etc.

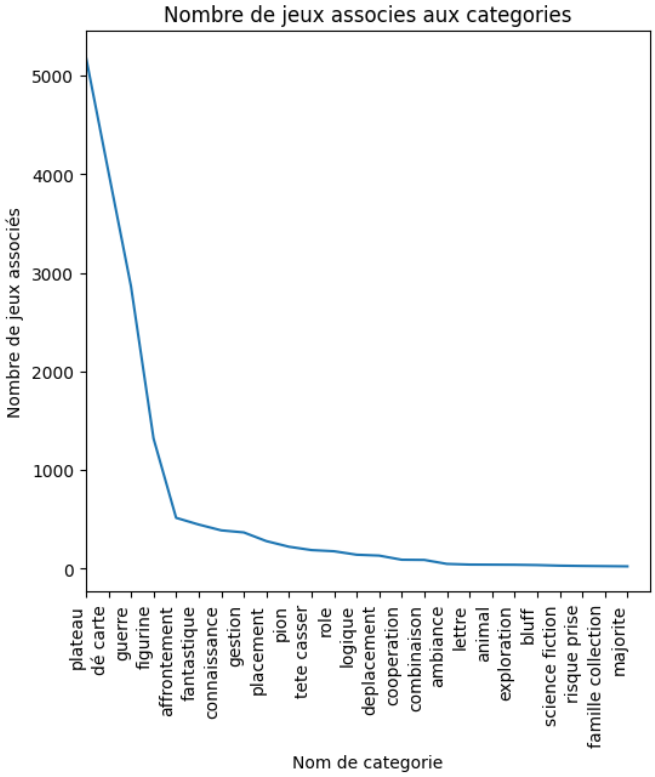


FIGURE 24 – Nombre de jeux associés aux catégories

Après avoir observé que les fréquences des catégories tendent rapidement vers de petits nombres, nous avons pris la décision de ne conserver que la catégorie la plus pertinente pour chaque jeu. Dans notre contexte, la pertinence est définie par la fréquence de la catégorie.

En réduisant le nombre de catégories de 59, nous avons réussi à en conserver seulement 96 catégories distinctes.

Nous avons appliqué une réduction basée sur la fréquence des catégories. Nous avons décidé de fixer un seuil minimum de 6 occurrences pour chaque catégorie. Ainsi, nous avons finalement obtenu un total de 35 catégories distinctes.

En appliquant cette réduction basée sur la fréquence des catégories, nous avons retiré la catégorie associée à environ 1% des jeux de la base de données. Cela signifie que ces jeux n'ont plus de catégorie assignée après la réduction. Ils ont été désormais considérés comme des jeux sans catégorie spécifique.

description : Dans le contexte de cette colonne, nous avons choisi d'appliquer une approche simple de TF-IDF pour l'analyse. Cette méthode nous permet de quantifier l'importance relative des termes présents dans la colonne en se basant sur leur fréquence et leur distribution dans l'ensemble des données.

5.2.2 Expérimentation

Pour l'approche basée sur le contenu, nous nous concentrons sur l'utilisation de la description et de la catégorie des jeux pour établir des similarités entre eux. Avant de procéder, nous éliminons les jeux qui ne disposent pas d'une description, ce qui correspond à une proportion de 11,58 % de l'ensemble des jeux.

Nous créerons une matrice jeux * jeux dans laquelle chaque valeur représente la similarité cosinus entre deux jeux. Cette matrice permettra de quantifier les similarités entre chaque paire de jeux en utilisant la mesure du cosine similarity.
Étant donné que nous utilisons plusieurs colonnes, nous avons décidé de calculer la similarité séparément pour chaque colonne, puis de les combiner en utilisant une pondération

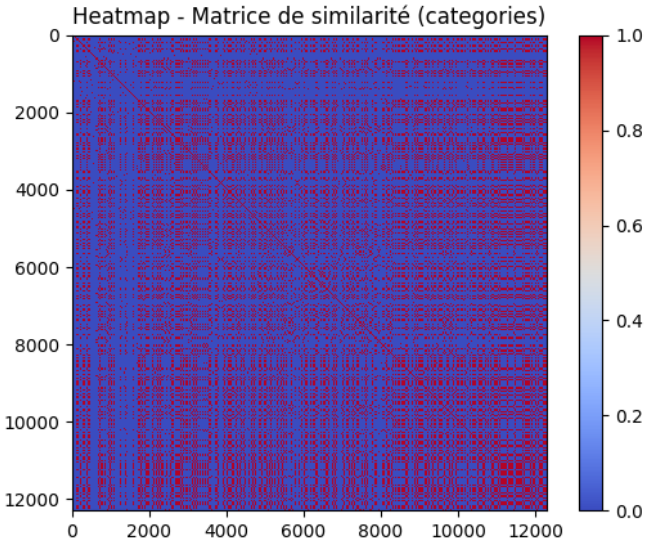


FIGURE 25 – Heatmap - Matrice de similarité (categories)

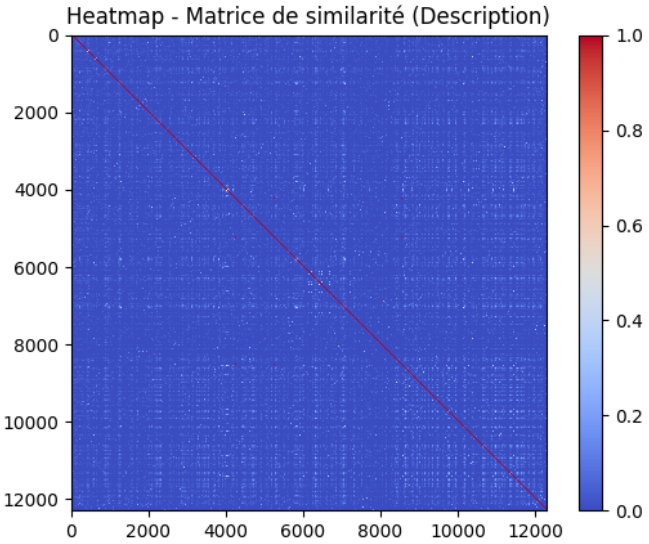


FIGURE 26 – Heatmap - Matrice de similarité (description)

Les figures ci-dessus représentent les matrices de similarité pour les colonnes "catégorie" et "description". Nous observons que pour la colonne "catégorie", la matrice de similarité identifie de nombreuses similarités, ce qui est attendu car il s'agit simplement de jeux appartenant à la même catégorie. En revanche, pour la colonne "description", la matrice de similarité trouve également des similarités, bien que leur valeur soit faible. Cela peut s'expliquer par le grand nombre de mots distincts présents dans les descriptions, ce qui conduit à des valeurs de similarité plus faibles en utilisant la mesure du cosine similarity.

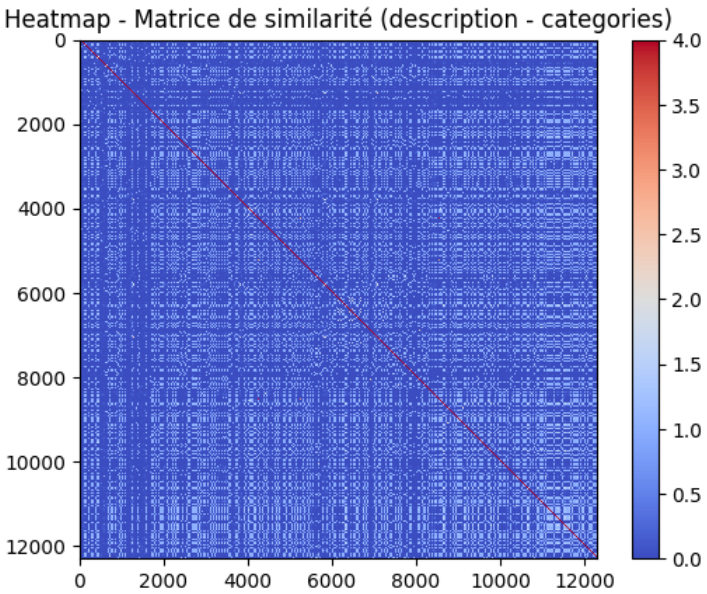


FIGURE 27 – Heatmap - Matrice de similarité (description - categories)

La figure représente une pondération des deux matrices précédentes (3 × description et 1 × catégorie). Nous avons choisi un poids de 3 pour la matrice de similarité basée

sur la description, car les valeurs de similarité obtenues étaient très petites. Cette matrice pondérée est considérée comme plus complète, car elle intègre davantage de colonnes dans le calcul de la similarité, en accordant une plus grande importance à la description des jeux.

5.2.3 Résultats

Afin d'évaluer nos matrices de similarité, nous avons sélectionné quelques jeux et affiché les jeux les plus similaires trouvés pour chacune des trois matrices.

titre	description	categories
Troyes Dice	troyes dice invite decouvrir histoire ville troyes ...	dé carte
Golden Horde	golden horde simulation simple rapide bataille ...	guerre

Nous avons sélectionné les deux jeux mentionnés précédemment afin de présenter les jeux similaires renvoyés par chaque matrice. Pour chaque matrice, nous afficherons à la fois le score de similarité et les jeux les plus similaires.

Catégories Nous commencerons par la matrice des catégories et afficherons les trois jeux les plus similaires.

titre	description	categories	score_similarite
Nineteen	nineteen placement reflexion base dé ici ...	dé carte	1.00
Talisman - 4e Édition	reedition corrigee grand classique ...	dé carte	1.00
The Artemis Project	europa lune jupiter croute terrestre ...	dé carte	1.00

TABLE 17 – Jeux similaire à Troyes Dice

titre	description	categories	score_similarite
Virus War	virus war prevu offrir party rythmee rapide ...	guerre	1.00
Ars Bellum	revivre grand bataille antiquit tete armee ...	guerre	1.00
Les trois portails	portail complexite moyenne wargame ...	guerre	1.00

TABLE 18 – Jeux similaire à Golden Horde

Après avoir examiné les résultats, nous constatons qu'elle ne renvoie que des jeux de la catégorie "guerre" avec un score de similarité de 1 pour le jeu de guerre "Golden Horde" et de même pour les jeux de cartes et de dés similaires à "Troyes Dice". Cela indique que la matrice se base uniquement sur cette colonne comme référence. Bien que les jeux de la même catégorie soient assez similaires entre eux, il est clair que se fier uniquement à cette dernière n'est pas suffisant pour capturer la diversité des jeux.

Description Nous passons à la matrice des descriptions et afficherons les trois jeux les plus similaires.

titre	description	categories	score_similarite
Saint Malo	faire prosperer citer victoire tete ...	plateau	0.188
Ginkgopolis	ginkgo bilober arbre vieux resistant ...	placement	0.181
Voyage autour du monde	contenu plateau pion drapeau ...	plateau	0.169

Table 19 – Jeux similaire à Troyes Dice

titre	description	categories	score_similarite
Zürich 1799	bataille systeme jour gloire initier vae victis ...	guerre	0.228
La Bérézina 1812	bataille systeme jour gloire initier vae victis ...	guerre	0.214
Tide of Iron	Fury of the Bear extension tid of iron ...	guerre	0.203

Table 20 – Jeux similaire à Golden Horde

Bien que les score de similarité soient plus faibles pour les descriptions que pour les catégories, en raison d'une plus grande dimensionnalité des vecteurs embedding des descriptions, on retrouve tout de même des jeux de catégories similaires. Par exemple, dans le cas du jeu Golden Horde, tous les jeux prédits sont de la même catégorie, "guerre" car les descriptions des jeux de guerre sont assez similaires.

Description & catégories Nous allons maintenant examiner les résultats de la matrice obtenue en combinant la description et la catégorie des jeux.

titre	description	categories	score_similarite
Troyes	editeur devenir veritabl tendance marche ...	dé carte	1.449
Blackrock City	blackrock city mettre puiser poker pervers ...	dé carte	1.405
Barbarossa	barbarossa carte type construction deck ...	dé carte	1.356

Table 21 – Jeux similaire à Troyes Dice

titre	description	categories	score_similarite
Zürich 1799	bataille systeme jour gloire initier vae ...	guerre	1.684
La Bérézina 1812	bataille systeme jour gloire initier vae ...	guerre	1.642
Tide of Iron	Fury of the Bearextension tid of iron ...	guerre	1.609

Table 22 – Jeux similaire à Golden Horde

Les résultats obtenus sont satisfaisants, car notre système de recommandation utilise la catégorie des jeux pour maximiser la similarité entre eux. Cependant, il utilise également la description pour éviter les recommandations aléatoires parmi les jeux de la même catégorie, comme le faisait la matrice de similarité basée uniquement sur la catégorie. Cette approche permet d'obtenir des recommandations plus pertinentes et diversifiées.

5.2.4 Discussion

Les résultats obtenus confirment l'efficacité de la combinaison des colonnes dans la matrice de similarité pour capturer des similarités significatives. Il est crucial de prendre en compte la qualité et les dimensions de chaque colonne pour déterminer les valeurs de pondération appropriées. Certains facteurs à considérer lors de l'attribution des pondérations sont la pertinence de la colonne par rapport à l'objectif de recommandation, la variabilité des valeurs dans la colonne, ainsi que la diversité des jeux représentés. Une approche itérative en ajustant les pondérations et en évaluant les performances peut aider à trouver un équilibre optimal pour chaque colonne. Il peut également être utile de solliciter les retours des utilisateurs et d'expérimenter différentes combinaisons de pondérations pour obtenir les meilleurs résultats.

Il est souvent difficile d'évaluer de manière précise un système de recommandation basé sur l'approche content-based utilisant la similarité cosinus. Cela est dû au fait que les résultats de ce type de système sont généralement des listes d'items considérés comme les plus similaires à un item donné, mais il n'existe pas de métrique standard pour évaluer directement cette similarité. Contrairement aux systèmes de recommandation collaboratifs qui peuvent être évalués à l'aide de mesures telles que celles que nous avons utilisées, les systèmes basés sur le contenu ne bénéficient pas d'une métrique équivalente pour évaluer la qualité des recommandations. Par conséquent, il est souvent nécessaire de recourir à des approches indirectes, telles que des évaluations par des experts ou des études utilisateurs, pour évaluer l'efficacité et la pertinence des recommandations basées sur le contenu, chose que nous n'avons pas pu faire. Ces approches qualitatives peuvent fournir des informations précieuses, mais elles ne permettent pas toujours une évaluation quantifiable et objective des performances du système, ce qui constitue une limite à ce type d'approche.

6 Conclusion

En conclusion, nous avons commencé par présenter une analyse approfondie du jeu de données afin de les prendre en main. Cette analyse nous a bien servi par la suite et les biais de nos données se sont fait ressentir dans les résultats de nos algorithmes de recommandation. Bien que nos résultats soient meilleurs que l'algorithme naïf qui prédit toujours la note moyenne de notre ensemble de données, ils n'ont pas su passer outre certains biais dont nous fait état plus haut. En effet, le nombre significatif de jeux n'ayant reçu peu ou pas avis, la tendance des utilisateurs à mettre des évaluations très positives, entraînent un biais dans les recommandations. Les systèmes survalorisent les items populaires et négligent les items moins fréquemment notés, même s'ils pourraient être pertinents pour certains utilisateurs.

En somme, ce rapport a mis en évidence le fait que, dans le cadre de l'élaboration de systèmes de recommandation, il est important d'utiliser des données qualitatives (fiables, complètes et cohérentes), des données diversifiées (tant du point de vue des utilisateurs que des items) et de l'utilisation de métriques d'évaluation appropriées.

Améliorations possibles Afin de construire un modèle de recommandation plus robuste, nous aurions pu utiliser des techniques avancées de NLP pour extraire des informations sémantiques à partir des commentaires des utilisateurs, en utilisant notamment l'analyse de sentiments afin de déterminer l'attitude ou l'opinion exprimée dans un avis ou l'exploration d'opinion qui consiste à extraire des informations spécifiques à partir des avis des utilisateurs, telles que les aspects mentionnés et les sentiments associés à ces aspects. Cela permet de comprendre les aspects appréciés ou critiqués par les utilisateurs dans les jeux. Des techniques telles que l'extraction de termes-clés, l'analyse de phrases ou l'exploration de thèmes peuvent être utilisées pour réaliser cette tâche.

En combinant les techniques de traitement du langage naturel avec les approches de recommandation, il est possible de développer un modèle robuste de recommandation de jeux basé sur les préférences des utilisateurs et les caractéristiques sémantiques des commentaires. Ces systèmes de recommandation peuvent améliorer l'expérience des utilisateurs en leur fournissant des suggestions personnalisées et pertinentes.

Il aurait également été intéressant de tester des techniques plus sophistiquées telles que les Graph Neural Networks (GNN). Les GNNs exploitent ces informations de graphe pour apprendre des représentations denses et significatives pour chaque item. Les nœuds du graphe, c'est-à-dire les items, sont mis à jour itérativement en agrégeant les informations de leurs voisins. Les GNNs peuvent également prendre en compte les attributs associés aux nœuds, tels que les descriptions d'items, les préférences des utilisateurs, etc., pour enrichir les représentations apprises [13, 14].

7 Bibliographie

Références

[1] Robin Burke. 2007. Hybrid Web Recommender Systems. DOI : 10.1007/978-3-540-72079-9_12

[2] T. George and S. Merugu. 2005. A Scalable Collaborative Filtering Framework Based on Co-Clustering. In Fifth IEEE International Conference on Data Mining (ICDM'05), IEEE, Houston, TX, USA, 625–628. DOI : 10.1109/ICDM.2005.14

[3] Yehuda Koren. 2010. Factor in the neighbors : Scalable and accurate collaborative filtering. ACM Trans. Knowl. Discov. Data 4, 1 (January 2010), 1–24. DOI : 10.1145/1644873.1644874

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. Computer 42, 8 (August 2009), 30–37. DOI : 10.1109/MC.2009.263

[5] Daniel Lemire and Anna Maclachlan. 2008. Slope One Predictors for Online Rating-Based Collaborative Filtering. DOI : 10.48550/arXiv.cs/0702144

[6] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. 2014. An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems. IEEE Transactions on Industrial Informatics 10, 2 (May 2014), 1273–1284. DOI : 10.1109/TII.2014.2308433

[7] Krupa Patel and Hiren B. Patel. 2020. A state-of-the-art survey on recommendation system and prospective extensions. Computers and Electronics in Agriculture 178, (November 2020), 105779. DOI : 10.1016/j.compag.2020.105779

[8] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR : Bayesian Personalized Ranking from Implicit Feedback. Retrieved May 20, 2023 from http://arxiv.org/abs/1205.2618

[9] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). 2011. Recommender Systems Handbook. Springer US, Boston, MA. DOI : 10.1007/978-0-387-85820-3

[10] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, ACM, Hong Kong Hong Kong, 285–295. DOI : 10.1145/371920.372071

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. Fifth International Conference on Computer and Information Science (January 2002).

[12] Sunny Sharma, Vijay Rana, and Vivek Kumar. 2021. Deep learning based semantic personalized recommendation system. International Journal of Information Management Data Insights 1, 2 (November 2021), 100028. DOI : 10.1016/j.jjimei.2021.100028

[13] Jingjing Wang, Haoran Xie, Fu Lee Wang, Lap-Kei Lee, and Oliver Tat Sheung Au. 2021. Top-N personalized recommendation with graph neural networks in MOOCs. Computers and Education : Artificial Intelligence 2, (January 2021), 100010. DOI : 10.1016/j.caeai.2021.100010

[14] Chenyan Zhang, Shan Xue, Jing Li, Jia Wu, Bo Du, Donghua Liu, and Jun Chang. 2023. Multi-Aspect enhanced Graph Neural Networks for recommendation. Neural Networks 157, (January 2023), 90–102. DOI : 10.1016/j.neunet.2022.10.001

[15] NetflixPrize-description.pdf. Retrieved May 22, 2023 from https://www.cs.uic.edu/ liub/KDD-cup-2007/NetflixPrize-description.pdf