

SORBONNE UNIVERSITE - FACULTE DES SCIENCES



ANALYSE DE REVIEWS ET RECOMMANDATION
UE Projet Logiciel

Autheurs :

Sofia BORCHANI

21212080

Souleymane MBAYE

28717871

Nolwenn PIGEON

21211546

Encadrants :

Nicolas BASKIOTIS

Vincent GUIGUE

Master 1 Données Apprentissage et Connaissances
Semestre 2

Table des matières

1	Introduction	3
2	Données d'analyse et prétraitements	4
2.1	Données utilisées	4
2.2	Format des données	4
3	Traitement automatique des langues	6
3.1	Exploration des données	6
3.1.1	Statistiques	6
3.1.2	Vocabulaire	8
3.1.3	Nettoyage des données	14
3.2	Classification des notes	14
3.2.1	Classes binaires	15
3.2.2	Expérimentations	15
3.2.3	Analyse des résultats	18
3.2.4	Classes ternaires	19
3.2.5	Expérimentations	19
3.2.6	Analyse des résultats	21
4	Recommandation	22
4.1	Outils et bibliothèques	22
4.2	Préliminaires	23
4.3	SVD	25
4.3.1	Expériences et résultats	26
4.4	Popular items	27
4.4.1	Distribution des jeux populaires	27
4.4.2	Expériences et résultats	28
4.5	LightFM	29
5	Conclusion	41

1 Introduction

Avec la montée en puissance du e-commerce, les entreprises sont de plus en plus conscientes de l'importance d'offrir une expérience d'achat satisfaisante pour leurs clients. Cela inclut non seulement la qualité du produit, mais également l'ensemble du processus d'achat, de la recherche du produit à l'après-vente. Dans ce contexte, les avis des clients sont devenus une source d'information cruciale pour les consommateurs, qui cherchent à obtenir des conseils et des avis avant de passer commande. En effet, les avis en ligne leur permettent de se faire une opinion sur la qualité et les performances du produit, ainsi que sur le service client de l'entreprise.

De plus, les plateformes d'avis en ligne ont vu leur popularité augmenter ces dernières années, et les consommateurs se tournent de plus en plus vers ces plateformes pour obtenir des conseils avant d'acheter un produit. Les réseaux sociaux ont également contribué à l'essor de la recommandation, où les clients partagent leurs opinions et leurs expériences avec leurs amis et leur réseau en ligne. Cette tendance a conduit les entreprises à investir davantage dans le marketing des réseaux sociaux pour atteindre leur public cible et générer des ventes grâce à la recommandation.

Dans ce projet, notre objectif est d'analyser les sentiments exprimés par les joueurs de jeux de société à travers les critiques et les avis sur le site TricTrac. En particulier, il s'agit de différencier les critiques positives des critiques négatives et d'évaluer l'impact de mots spécifiques pour comprendre comment l'analyse de sentiments peut aider les machines à recommander des jeux.

Nous commencerons par nettoyer et formater les données avant d'utiliser des méthodes de traitement automatique du langage naturel pour classifier les sentiments. Enfin, en utilisant le filtrage collaboratif, nous étudierons un système de recommandation basé sur les critiques et les notes des utilisateurs.

2 Données d'analyse et prétraitements

2.1 Données utilisées

Pour ce projet, une base de données de revues a été récoltée sur un site spécialisé dans les jeux de société, appelé TricTrac. La base contient 150 000 avis, 200 000 notes sur plus de 15 000 jeux.

2.2 Format des données

Les fichiers '*details*' et '*avis*' sont au format JSON. Pour visualiser plus facilement ces fichiers, nous avons décidé de travailler avec le système de gestion de base de données MongoDB. MongoDB est une base de données NoSQL (non relationnelle) open source qui permet de stocker des données sous forme de documents JSON.

La collection '*avis*' stocke des informations sur les avis et critiques d'un jeu de société donné. Chaque avis est représenté sous la forme d'un dictionnaire, contenant différents champs. Parmi ces champs, on peut trouver l'identifiant unique de l'avis, le nom de l'auteur de l'avis, la date et l'heure de publication de l'avis, le titre de l'avis, la note attribuée par l'auteur (sur une échelle de 0 à 10), l'URL vers la page de l'avis sur le site Tric Trac, le titre du jeu de société pour lequel l'avis a été écrit, l'URL vers la page du jeu sur le site Tric Trac, un booléen indiquant si l'avis a été traité ou non, et enfin un commentaire ou une critique de l'auteur sur le jeu de société. Ces informations permettent de se faire une idée de l'opinion générale des joueurs et des critiques sur le jeu de société en question.

```
{  
    "author": "Joleb",  
    "date_published": "2019-02-04 01:59:47",  
    "title_review": "Armez-vous paysans 🌱",  
    "note": 9.8,  
    "review_href": "https://www.trictrac.net/avis/armez-vous-paysans",  
    "title": "Goblivion",  
    "url": "https://www.trictrac.net/jeu-de-societe/goblivion/avis?limit=10000",  
    "treated": true,  
    "comment": "Excellent jeu de cartes de type deckbuilder. On devient très vite addicte à ce jeu 😊"  
}
```

FIGURE 1 – Exemple format avis

La collection '*details*' stocke des informations spécifiques sur un jeu de société, comme son titre complet, son URL sur le site Tric Trac, sa description, son casting (créateurs et illustrateurs), ses catégories, ainsi que des informations sur

son gameplay (nombre de joueurs, âge minimum, durée). La collection '*details*' stocke également des informations sur la note moyenne du jeu, ainsi que le nombre d'avis. Il y a trois plus précisément trois champs qui stockent des informations sur la note moyenne d'un jeu de société : 'Note', 'Note rectifiée' et 'Note Finkel'. La 'Note' correspond à la note moyenne du jeu, telle que calculée à partir de l'ensemble des avis des joueurs. La 'Note rectifiée' correspond à une note qui a été corrigée pour tenir compte de certains biais éventuels, tels que des notes extrêmes ou des notes données par des utilisateurs peu actifs. Enfin, la 'Note Finkel' est une note qui est attribuée par un expert du jeu de société (en l'occurrence, Monsieur Finkel). Cette note est donnée à titre indicatif et est destinée à aider les joueurs dans leur choix. En résumé, ces champs permettent d'avoir une vue d'ensemble sur les caractéristiques du jeu de société, ainsi que sur son accueil auprès des joueurs et des critiques.

```
{
  "categories": "",
  "casting": "Par Eric Lang et Andrea ChiarvesioIllustré par Édouard Guitonédité par CMON Limited",
  "gameplay": "1 à 4|14 ans et +|45",
  "description": "\n      Marvel United : unis contre les Super-Vilains !\n      Red Skull, Ultron et Taskmaster attaquent la ville, chacun avec son propre plan diabolique de destruction et de domination. Pour les arrêter, des super-héros comme Captain America, Iron Man et Captain Marvel unissent leurs forces. Au final, les efforts combinés de ces héros suffiront-ils à sauver le monde ?\n      Dans Marvel United, les joueurs doivent travailler ensemble pour arrêter les plans d'un Supervillain malfaisant. Ils utiliseront le paquet personnalisé de pouvoirs et de capacités de chaque héros, en jouant sur les forces des uns et des autres pour plus d'effet, tout en essayant de terminer les missions et d'éliminer le méchant avant qu'il ne soit trop tard. Ils se battront dans des endroits tout autour de la ville. Au final, soit la ville sera sauvée, soit le monde sera condamné.\n    ",
  "full_title": "Marvel United (2020)",
  "titre": "Marvel United",
  "url": "https://www.trictrac.net/jeu-de-societe/marvel-united",
  "Note": 7.58,
  "Note rectifiée": 8.4,
  "Note Finkel": 8.74,
  "Nombre d'avis": 19,
  "Nombre d'avis rectifiés": 10
}
```

FIGURE 2 – Exemple format details

3 Traitement automatique des langues

Le Traitement Automatique du Langage naturel (Natural Language Processing en anglais) est un outil de l'apprentissage automatique spécialisé dans la linguistique. Il existe plusieurs sujets de TAL parmi lesquels on retrouve la classification de texte, le filtrage de contenu et l'analyse de sentiment.

Dans le cadre d'une classification de textes utilisant la méthode Bag of Words (BoW), la première étape consiste généralement en un prétraitement des données (preprocessing), puis en la construction du modèle BoW, l'apprentissage et enfin l'optimisation des paramètres.

Le but de cette première partie est de parvenir à distinguer les revues positives et négatives et d'avoir un tagging au niveau des mots dans le but d'apprendre ce que les joueurs aiment ou n'aiment pas dans un jeu. Notre jeu de données ne contenant pas de labels, nous avons choisi d'effectuer la classification de sentiments à l'aide des notes.

3.1 Exploration des données

3.1.1 Statistiques

Pour mieux comprendre les caractéristiques sémantiques des avis de jeu de société stockés dans la collection "avis", nous avons effectué des analyses statistiques approfondies.

mediane	moyenne	variance	ecart-type
8.0	7.97	4.519	2.125

TABLE 1 – Statistiques des notes

Nous avons d'abord calculé la répartition des notes attribuées par les critiques, ainsi que les statistiques descriptives des notes. En particulier, nous avons observé que la médiane des notes données par les joueurs est de 8 sur 10, ce qui est très élevé. Cela signifie que les joueurs ont tendance à donner des notes très élevées aux jeux de société qu'ils ont évalués. En effet, une note de 7 ou moins pourrait être interprétée comme une mauvaise note par les joueurs, étant donné la distribution des notes. Il est important de prendre en compte ce biais dans l'interprétation des notes moyennes des jeux. La variance de 4.52 et l'écart type de 2.12 indiquent que les notes sont relativement dispersées autour de la moyenne de

7,97. En d'autres termes, les avis des joueurs sont assez divergents quant à la qualité des jeux de société. Nous avons également identifié les jeux avec le plus grand nombre d'avis, afin de mieux comprendre les jeux les plus populaires auprès des joueurs.

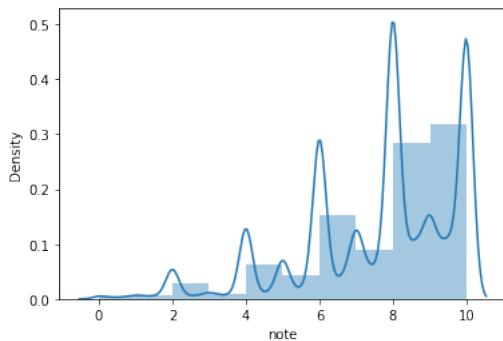


FIGURE 3 – Distribution globale des notes

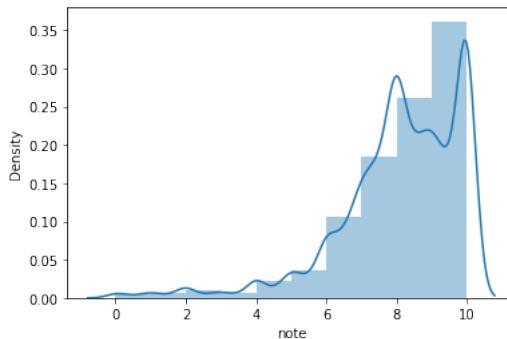


FIGURE 4 – Distribution des notes par auteurs

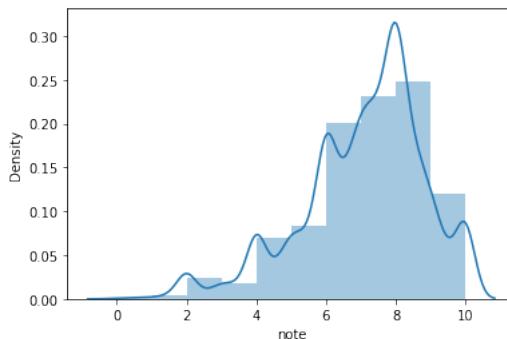


FIGURE 5 – Distribution des notes par jeux

3.1.2 Vocabulaire

A l'aide de techniques de preprocessing du langage, nous avons étudié les mots les plus fréquemment utilisés dans les commentaires des critiques, ainsi que les bigrammes et trigrammes les plus courants. Cette analyse nous a permis d'identifier les mots clés les plus associés à chaque jeu de société. Nous avons également identifié les "stop words", c'est-à-dire les mots les plus couramment utilisés en français, afin de mieux comprendre leur impact sur la sémantique des commentaires. Enfin, nous avons utilisé le "odds ratio" pour identifier les mots qui étaient le plus fortement associés à chaque jeu de société. Cette analyse nous a permis de mieux comprendre les mots clés qui ont le plus d'impact sur l'appréciation d'un jeu par les critiques.

Le vocabulaire a une taille initiale de 140837.



FIGURE 6 – Vocabulaire initial

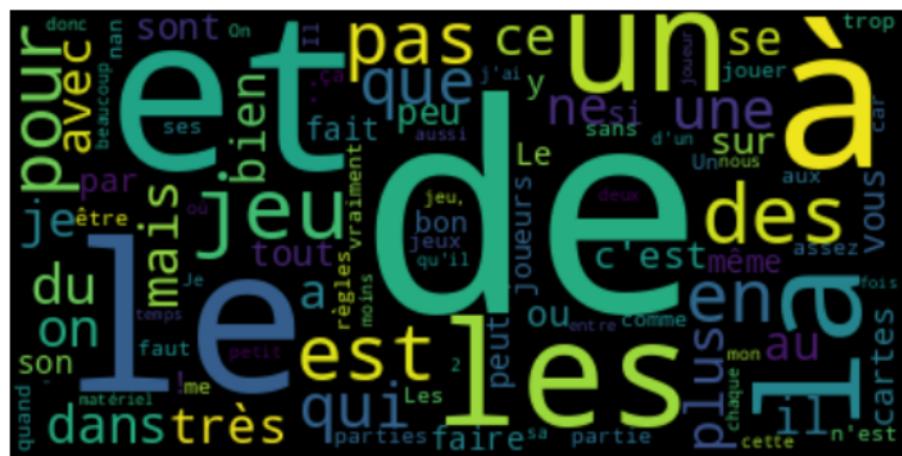


FIGURE 7 – Les 100 termes les plus fréquents - wordcloud

	1 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80	81 - 90	91 - 100
0	de	pas	mais	se	par	bon	trop	assez	moins	cette
1	et	pour	ne	c'est	même	joueurs	aux	n'est	sa	-
2	le	en	dans	sur	fait	partie	faut	Il	Je	d'un
3	à	que	avec	sont	si	ça	vraiment	car	On	chaque
4	les	qui	a	tout	!	jouer	comme	quand	nous	où
5	la	une	je	ou	peut	nan	Un	petit	deux	matériel
6	un	du	très	:	y	Les	règles	aussi	j'ai	temps
7	jeu	on	il	vous	son	parties	me	donc	mon	joueur
8	des	plus	au	peu	Le	jeux	être	qu'il	jeu,	entre
9	est	ce	bien	cartes	faire	sans	ses	2	beaucoup	fois

FIGURE 8 – Les 100 termes les plus fréquents - tableau

	1 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80	81 - 90	91 - 100
0	le jeu	de jeu	je ne	avec des	de ce	avec un	ne pas	jeux de	une fois	est une
1	un jeu	qu il	les règles	que le	jeu qui	pas de	le matériel	le plus	et est	quand même
2	de la	il faut	des cartes	que les	plus de	pour le	peu de	même si	ce que	des règles
3	ce jeu	il est	les cartes	est très	les parties	on est	je ai	de plus	pour un	je et
4	un peu	ce qui	on peut	de cartes	peut être	dans les	le thème	en plus	et des	je suis
5	jeu de	et de	dans la	un bon	les autres	sur la	tout le	très bon	le monde	qui ne
6	est un	dans le	et le	bon jeu	on se	si on	avec les	de jouer	pour moi	les jeux
7	jeu est	qu on	sur le	on ne	que on	si vous	un petit	de base	très bien	je trouve
8	du jeu	pour les	la partie	qui est	et on	une partie	sur les	est bien	est le	beaucoup de
9	est pas	et les	que je	les joueurs	et la	petit jeu	tous les	ce est	nombre de	le plateau

FIGURE 9 – Les bigrammes les plus fréquents

	1 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80	81 - 90	91 - 100
0	le jeu est	les règles sont	jeu de base	autour de la	je ne suis	ne suis pas	un peu trop	il est pas	en même temps	plus ou moins
1	un jeu de	un bon jeu	je ai pas	qu il faut	dans ce jeu	pas du tout	est pas un	il ne faut	pour ne pas	ce qu on
2	est un jeu	le jeu de	un petit jeu	le nombre de	fin de partie	en fin de	les autres joueurs	des jeux de	le fait de	des points de
3	ce jeu est	ce qui est	de la table	de ne pas	je me suis	bon jeu de	jeu est pas	la première partie	je pense que	la possibilité de
4	tout le monde	très bon jeu	un peu plus	le système de	du jeu de	la fin de	plateau de jeu	en tout cas	que les autres	temps en temps
5	ce est pas	de la partie	tout de même	jeu de cartes	pas mal de	mise en place	un jeu très	les cartes sont	tout de suite	je trouve que
6	un peu de	que le jeu	qu il est	les jeux de	on ne peut	points de victoire	pas grand chose	jeu de gestion	jeu est un	un bon moment
7	un jeu qui	ne sont pas	les parties sont	le thème est	jeu est très	un excellent jeu	et le jeu	entre les joueurs	bon petit jeu	prise de tête
8	le matériel est	un très bon	du jeu est	ce qu il	petit jeu de	le jeu en	pour ceux qui	est un bon	de temps en	durée de vie
9	de ce jeu	est un peu	pour un jeu	dans le jeu	que ce jeu	sur le plateau	de jeu est	si vous avez	le jeu se	qui permet de

FIGURE 10 – Les trigrammes les plus fréquents



FIGURE 11 – Vocabulaire initial sans les stopwords



FIGURE 12 – Les 100 termes les plus fréquents sans les stopwords - wordcloud

	1 - 10	11 - 20	21 - 30	31 - 40	41 - 50	51 - 60	61 - 70	71 - 80	81 - 90	91 - 100
0	jeu	peut	trop	petit	joueur	plateau	choix	fin	plutôt	enfants
1	plus	faire	sans	moins	5	bonne	non	extension	alors	1
2	bien	bon	faut	temps	points	autre	quelques	grand	contre	Bref
3	très	parties	2	3	thème	encore	reste	trouve	place	permet
4	cartes	jouer	autres	4	où	toujours	va	mécanique	part	avant
5	peu	ça	vraiment	simple	tour	plaisir	mal	aime	actions	mieux
6	joueurs	être	assez	beaucoup	entre	vite	ambiance	après	base	prendre
7	partie	jeux	donc	matériel	là	joue	rien	sympa	monde	plusieurs
8	fait	nan	deux	fois	chaque	hasard	coup	stratégie	rapide	souvent
9	si	règles	car	avoir	carte	dés	surtout	dire	moment	point

FIGURE 13 – Les 100 termes les plus fréquents sans les stopwords - tableau



FIGURE 14 – Les 100 termes les plus discriminants au sens du odds ratio



FIGURE 15 – Les 100 termes les plus fréquents par note

3.1.3 Nettoyage des données

Après avoir nettoyé les données et nous avons constaté la présence de 13623 utilisateurs et 10709 jeux dans une base de taille 246524.

Nous avons ensuite cherché à identifier les cas où un utilisateur a noté plusieurs fois le même jeu en regroupant les avis par auteur et titre. Nous avons remarqué que 111 notes avaient été renotées au moins une deuxième fois, représentant 0.063% de l'ensemble des notes. Par ailleurs, 81 utilisateurs ont renoté au moins une deuxième fois un jeu, soit 0.594% des utilisateurs et 65 jeux ont été renotés au moins une deuxième fois, soit 0.606% des jeux.

Nous avons enregistré ces nouvelles notations dans un autre fichier JSON pour observer comment les utilisateurs changent leurs avis. Nous avons ensuite supprimé les doublons dans notre base de données et décidé de garder la dernière note pour la suite de nos analyses.

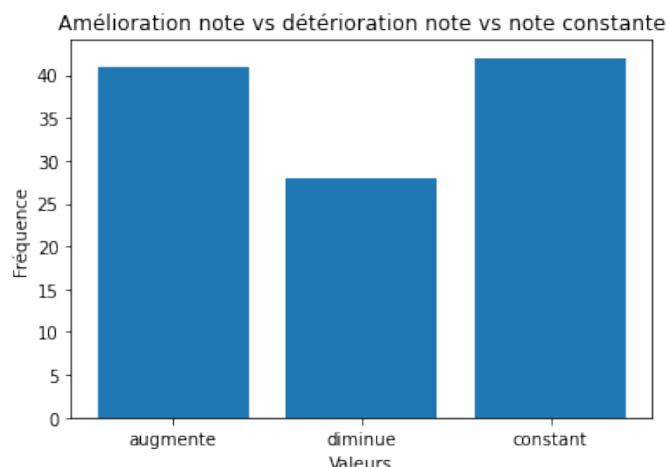


FIGURE 16 – Amélioration, Détérioration des notes

3.2 Classification des notes

Après avoir effectué des analyses statistiques sur les données, nous avons décidé de nous intéresser à la classification des notes des jeux de société en fonction des avis des joueurs. Pour cela, nous avons développé des classificateurs binaires et ternaires pour prédire la note attribuée par les joueurs en se basant sur la description et l'avis donné pour chaque jeu. L'objectif de cette analyse était de voir si la classification binaire était suffisante ou si une classification plus fine était nécessaire pour capturer les nuances des avis des utilisateurs. Ces classificateurs ont été créés dans le but de pouvoir mieux comprendre les facteurs qui influencent la note attribuée par les joueurs et de pouvoir proposer des recommandations de jeux en

fonction des goûts des joueurs.

Nous avons utilisé deux méthodes de vectorisation, CountVectorizer et TfifdVectorizer, ainsi que quatre algorithmes de classification : naives bayes, régression logistique, SVM et random forest pour classer les commentaires des jeux. Nous avons également examiné l'impact de l'utilisation de stop words et des termes les plus fréquents comme stop words sur les performances de classification. Pour évaluer la performance de nos modèles nous avons utilisé la précision et le F1-score.

3.2.1 Classes binaires

Après avoir effectué l'opération de binarisation avec un seuil fixé à 5, nous avons procédé à une séparation des données en deux ensembles distincts, à savoir un ensemble d'entraînement et un ensemble de test. Par ailleurs, dans le but de rétablir l'équilibre des classes lors de l'apprentissage, nous avons également employé la technique de RandomUnderSampler.

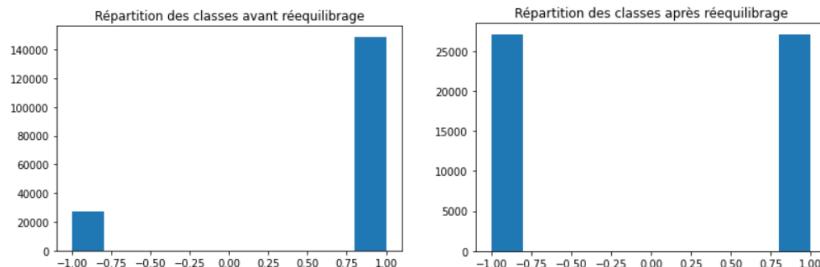


FIGURE 17 – Rééquilibrage des classes binaires.

3.2.2 Expérimentations

Dans cette partie nous avons mené différentes expérimentations en variant un certain nombre de paramètres. Pour CountVectorizer et TfifdVectorizer, nous avons regardé les paramètres suivants :

- **lowercase** : conversion en minuscule ou non du corpus.
- **binary** : BoW binaire ou non.
- **stopwords** : liste des stopwords à supprimer.
- **max_df** : seuil de fréquence maximum pour les mots qui apparaissent dans le texte.
- **min_df** : seuil de fréquence minimum pour les mots qui apparaissent dans le texte.

- **ngram_range** : intervalle des ngrammes.

Nous avons établi diverses fonctions de prétraitement :

- **delete_punctuation** qui supprime la ponctuation dans un texte.
- **delete_digit** qui supprime tous les chiffres d'un texte.
- **stem** qui effectue la racinisation (ou "stemming" en anglais) des mots dans un texte.
- **lemmatize** qui effectue la lemmatisation des mots dans un texte.

Différents jeux de test :

1. CountVectorizer
2. TfidfVectorizer
3. CountVectorizer, stop_words
4. TfidfVectorizer, stop_words
5. CountVectorizer, stop_words et les 200 termes les plus fréquents
6. TfidfVectorizer, stop_words et les 200 termes les plus fréquents
7. CountVectorizer, stop_words et les 2000 termes les plus fréquents
8. TfidfVectorizer, stop_words et les 2000 termes les plus fréquents
9. CountVectorizer, stop_words et les 20000 termes les plus fréquents
10. TfidfVectorizer, stop_words et les 20000 termes les plus fréquents
11. CountVectorizer, stop_words et les 20000 termes les plus fréquents, stem
12. TfidfVectorizer, ngram_range=(1, 2)
13. CountVectorizer, ngram_range=(1, 2), max_df=0.5, stop_words
14. TfidfVectorizer, ngram_range=(1, 2), max_df=0.5, stop_words
15. CountVectorizer, ngram_range=(1, 3)
16. CountVectorizer, ngram_range=(1, 3), max_df=0.5, stop_words
17. TfidfVectorizer, ngram_range=(1, 3), max_df=0.5, stop_words
18. CountVectorizer, stop_words, stem
19. TfidfVectorizer, stop_words, stem

V	Accuracy	F1 score	Best model
1	0.809	0.825	Logistic Regression
2	0.824	0.837	Logistic Regression
3	0.777	0.799	SVM
4	0.819	0.833	Logistic Regression
5	0.788	0.808	Logistic Regression
6	0.806	0.822	Logistic Regression
7	0.823	0.797	Random Forest
8	0.783	0.803	Logistic Regression
9	0.843	0.776	Random Forest
10	0.843	0.776	Random Forest
11	0.614	0.667	Logistic Regression
12	0.842	0.852	SVM
13	0.825	0.838	Logistic Regression
14	0.833	0.845	SVM
15	0.835	0.846	Logistic Regression
16	0.825	0.838	Logistic Regression
17	0.833	0.845	SVM
18	0.775	0.797	SVM
19	0.813	0.828	Logistic regression

TABLE 2 – Classification binaire résultats

Nous avons également extrait les termes les plus significatifs pour un modèle de classification donné, en utilisant un objet vectorizer spécifique. Les termes ont été classés en termes positifs et négatifs en fonction de leur poids respectif dans le modèle. Les résultats obtenus pour certains modèles sont les suivants :

V	Mots positifs	Mots négatifs
2	excellent, bon, bémol, bien, adore	intérêt, déception, aucune, pas, ennui
5	bémol, pépite, régal, priver, assurée	décevant, ennuyé, ennuyeux, poussif, fade
12	excellent, bon jeu, très bon, pas trop, pas mal	déception, intérêt, pas, trop, ennuyeux
15	excellent, pas mal, adore, pas trop, excellente	bof, ennuyeux, ennui, déception, intérêt
19	excellent, bémol, bon, agréabl, efficace	ennui, décept, intérêt, aucun, déçu

TABLE 3 – Termes les plus significatifs

3.2.3 Analyse des résultats

Les modèles basés sur le TfIdfVectorizer ont donné des résultats globalement solides, avec des précisions allant de 0.783 à 0.842 et des scores F1 de 0.776 à 0.852. Ces résultats indiquent une capacité prometteuse à prédire avec précision les classes positives et négatives. Parmi les configurations testées, la régression logistique et SVM se sont avérés être les meilleurs modèles, offrant une bonne discrimination entre les classes.

En ce qui concerne le CountVectorizer, les résultats ont montré des précisions allant de 0.614 à 0.825 et des scores F1 de 0.667 à 0.846. Bien que inférieurs à ceux du CountVectorizer, ces scores indiquent toujours une performance solide dans la classification binaire. Les meilleurs modèles pour ces configurations étaient également la régression logistique et SVM.

Lors de l'exploration de différentes variantes, telles que l'inclusion ou l'exclusion des mots vides (stop words) et l'ajout de termes les plus fréquents, il n'y avait pas de tendance claire d'amélioration significative des performances. Les résultats ont varié dans une fourchette relativement proche pour ces configurations, avec des précisions allant de 0.777 à 0.843 et des scores F1 de 0.799 à 0.852.

L'utilisation de la racinisation (stemming) n'a pas conduit à une amélioration significative des résultats. Les configurations qui incluaient le stemming n'ont pas atteint les mêmes niveaux de précision et de score F1 que les autres configurations.

L'utilisation de n-grammes (unigrammes, bigrammes, trigrammes) a amélioré les performances dans certains cas. Par exemple, la configuration utilisant des n-grammes de 1 à 2 a obtenu des résultats prometteurs, avec une précision de 0,842 et un score F1 de 0,852 en utilisant SVM.

Dans l'ensemble, il est intéressant de noter que la régression logistique et SVM étaient des choix cohérents pour les meilleurs modèles dans différentes configu-

rations. Cela suggère que ces algorithmes sont adaptés à la tâche de classification binaire avec les jeux de données et les caractéristiques utilisés. Random Forest a obtenu un score F1 maximale de 0.797.

En conclusion, cette analyse des résultats met en évidence les performances variées des différentes configurations de vecteurs de caractéristiques dans la classification binaire. Ces résultats servent de base pour affiner et choisir la meilleure configuration pour des tâches de classification similaires à l'avenir.

3.2.4 Classes ternaires

Nous avons opéré une classification de l'ensemble de données en trois catégories distinctes, à savoir les notes inférieures ou égales à 5, les notes comprises entre 6 et 7, et enfin les notes supérieures ou égales à 8. Ensuite, nous avons procédé à une séparation des données en un ensemble d'entraînement et un ensemble de test. On utilise également RandomUnderSampler pour rééquilibrer les classes en apprentissage.

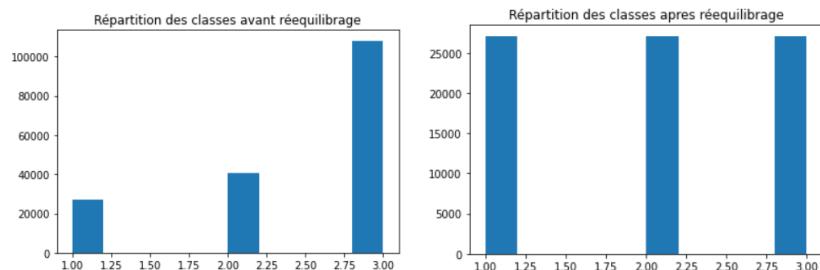


FIGURE 18 – Rééquilibrage des classes ternaires.

3.2.5 Expérimentations

Dans cette partie nous avons mené différentes expérimentations en variant un certain nombre de paramètres.

Différents jeux de test :

- 1. CountVectorizer
- 2. TfidfVectorizer
- 3. CountVectorizer, stop_words
- 4. TfidfVectorizer, stop_words
- 5. CountVectorizer, stop_words et les 200 termes les plus fréquents
- 6. TfidfVectorizer, stop_words et les 200 termes les plus fréquents
- 7. CountVectorizer, stop_words et les 2000 termes les plus fréquents
- 8. TfidfVectorizer, stop_words et les 2000 termes les plus fréquents

- les 2000 termes les plus fréquents
9. CountVectorizer, stop_words et les 20000 termes les plus fréquents
 10. TfidfVectorizer, stop_words et les 20000 termes les plus fréquents
 11. CountVectorizer, stop_words et les 20000 termes les plus fréquents, stem
 12. TfidfVectorizer, ngram_range=(1, 2)
 13. CountVectorizer, ngram_range=(1, 2), max_df=0.5, stop_words
 14. TfidfVectorizer, ngram_range=(1, 2), max_df=0.5, stop_words
 15. CountVectorizer, ngram_range=(1, 3)
 16. CountVectorizer, ngram_range=(1, 3), max_df=0.5, stop_words
 17. TfidfVectorizer, ngram_range=(1, 3), max_df=0.5, stop_words
 18. CountVectorizer, stop_words, stem
 19. TfidfVectorizer, stop_words, stem

V	Accuracy	F1 score	Best model
1	0.556	0.582	Logistic Regression
2	0.587	0.612	Logistic Regression
3	0.553	0.578	Naïve Bayes
4	0.582	0.607	Logistic Regression
5	0.529	0.556	Logistic Regression
6	0.561	0.586	Logistic Regression
7	0.445	0.473	Naïve Bayes
8	0.561	0.586	Logistic Regression
9	0.614	0.471	Random Forest
10	0.615	0.471	Random Forest
11	0.490	0.518	Naïve Bayes
12	0.607	0.631	Logistic Regression
13	0.608	0.627	Naïve Bayes
14	0.610	0.629	Naïve Bayes
15	0.621	0.638	Naïve Bayes
16	0.612	0.630	Naïve Bayes
17	0.610	0.629	Naïve Bayes
18	0.550	0.576	Logistic Regression
19	0.577	0.602	Logistic regression

TABLE 4 – Classification ternaire résultats

3.2.6 Analyse des résultats

Parmi les différents jeux de test, les modèles basés sur le CountVectorizer ont donné des résultats variables. Les précisions allaient de 0.445 à 0.621 et les scores F1 de 0.471 à 0.638. Ces résultats indiquent une certaine variabilité dans la capacité des modèles à prédire avec précision les trois classes. La régression logistique, la forêt aléatoire et le modèle de Bayes naïf se sont avérés être les meilleurs modèles pour différentes configurations.

En ce qui concerne le TfidfVectorizer, les résultats ont montré des précisions allant de 0.587 à 0.621 et des scores F1 de 0.612 à 0.638. Ces scores indiquent une performance similaire à celle du CountVectorizer, bien que légèrement supérieure dans certains cas. Les meilleurs modèles pour ces configurations étaient la régression logistique et le modèle de Bayes naïf.

L'analyse des différentes configurations de vecteurs de caractéristiques révèle que l'inclusion ou l'exclusion des mots vides (stop words), l'utilisation des termes les plus fréquents et la modification des hyperparamètres tels que les n-grammes n'ont pas conduit à des améliorations significatives des performances. Les résultats ont varié dans une fourchette relativement proche pour ces configurations, avec des précisions allant de 0.445 à 0.621 et des scores F1 de 0.473 à 0.638. C'est la configuration utilisant des n-grammes de 1 à 3 qui a obtenu des résultats prometteurs, avec un score F1 de 0.638 en utilisant le modèle de Bayes naïf.

En conclusion, cette analyse corrigée des résultats met en évidence la diversité des performances des différentes configurations de vecteurs de caractéristiques dans la classification ternaire. Les meilleurs modèles étaient souvent la régression logistique et le modèle de Bayes naïf, indiquant leur pertinence pour cette tâche de classification. Il convient de noter que les performances variaient considérablement en fonction des jeux de données et des caractéristiques utilisés.

Ces résultats fournissent des indications importantes pour sélectionner la configuration optimale lors de la construction de modèles de classification pour des tâches similaires à l'avenir.

4 Recommandation

Maintenant que nous avons abordé l'analyse de sentiments, nous pouvons passer à la problématique de la recommandation. En effet, l'analyse de sentiments peut être utilisée pour améliorer la qualité des recommandations en prenant en compte les opinions et les émotions des utilisateurs.

La recommandation est un domaine clé de l'apprentissage automatique qui consiste à suggérer des produits, des services ou des contenus pertinents pour les utilisateurs en se basant sur leurs préférences et leurs comportements passés. Dans le contexte des données textuelles telles que des commentaires et des notes, les techniques de filtrage collaboratif peuvent être utilisées pour recommander des articles ou des produits similaires à ceux que l'utilisateur a déjà appréciés.

Sur le site TricTrac, les utilisateurs peuvent évaluer les jeux de société ou simplement les consulter, révélant ainsi leurs préférences. Toutes ces évaluations peuvent être considérées comme une matrice de taille ($n_{\text{utilisateurs}}, n_{\text{jeux}}$) qui enregistre toutes les évaluations. Cependant, cette matrice est souvent très clairsemée car les utilisateurs n'évaluent qu'un sous-ensemble de jeux. Pour prédire les évaluations manquantes et recommander des jeux pertinents, le filtrage collaboratif est utilisé en exploitant la similarité des modèles d'évaluation des utilisateurs. Le but du filtrage collaboratif est de prédire l'évaluation qu'un utilisateur donnerait à un jeu donné, en remplissant les évaluations manquantes de la matrice. Les algorithmes de factorisation matricielle sont souvent utilisés pour cette tâche. Ces algorithmes permettent de décomposer la matrice d'évaluations en deux sous-matrices : une pour les utilisateurs et une pour les jeux de société.

4.1 Outils et bibliothèques

Pour la mise en place de notre système de recommandation de jeux de société basé sur les commentaires et les notes des utilisateurs, nous allons utiliser différentes bibliothèques en Python. Tout d'abord, nous allons utiliser la bibliothèque Pandas, qui est un outil de traitement des données très utile pour la manipulation et l'analyse de données en tableaux. En utilisant Pandas, nous pourrons facilement charger et nettoyer les données provenant de TricTrac et préparer la matrice d'évaluations nécessaire à la mise en place du filtrage collaboratif.

Ensuite, nous allons utiliser la bibliothèque Seaborn, qui est un outil très pratique pour visualiser les données en créant de beaux graphiques. Avec Seaborn, nous pourrons visualiser la distribution des évaluations de chaque jeu, la distribution des évaluations par utilisateur, ainsi que la corrélation entre les différentes

évaluations.

Enfin, nous allons utiliser la bibliothèque Scikit-surprise, qui est une bibliothèque dédiée à la mise en place de systèmes de recommandation basés sur le filtrage collaboratif. Scikit-surprise fournit différents algorithmes de filtrage collaboratif, ainsi que des outils pour évaluer et comparer la performance de ces algorithmes. Nous testerons notamment trois algorithmes de recommandation différents : SVD, Popular items et BPR, pour comparer leur performance en termes de précision et de diversité des recommandations. Par conséquent, en utilisant Scikit-surprise, nous pourrons facilement entraîner et évaluer notre système de recommandation de jeux de société et choisir l'algorithme le plus performant.

4.2 Préliminaires

Nous avons ensuite calculé les statistiques sur le nombre d'utilisateurs en fonction du nombre minimum de notes qu'ils ont données. Nous avons retenu les cas où les utilisateurs ont noté au moins 10, 50, 100 ou 200 jeux. Nous avons ainsi pu observer que le nombre d'utilisateurs restant dans la base de données diminue à mesure que le nombre minimal de notes augmente, avec 2849 utilisateurs ayant noté au moins 10 jeux, 734 pour au moins 50 jeux, 359 pour au moins 100 jeux, et 128 pour au moins 200 jeux. La même tendance se reflète également pour le nombre de jeux restant dans la base et la taille du DataFrame restant.

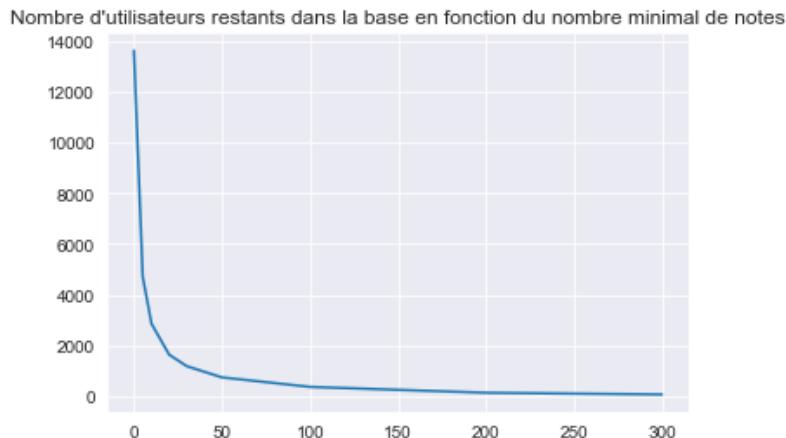


FIGURE 19 – Nombre d'utilisateurs restant dans la base en fonction du nombre minimal de notes

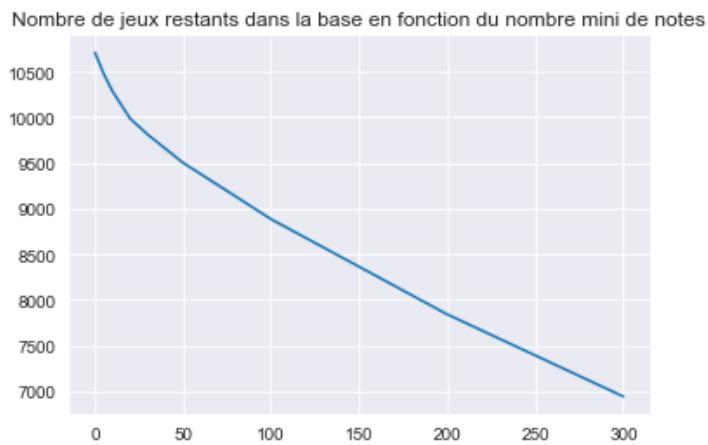


FIGURE 20 – Nombre de jeux restant dans la base en fonction du nombre minimal de notes

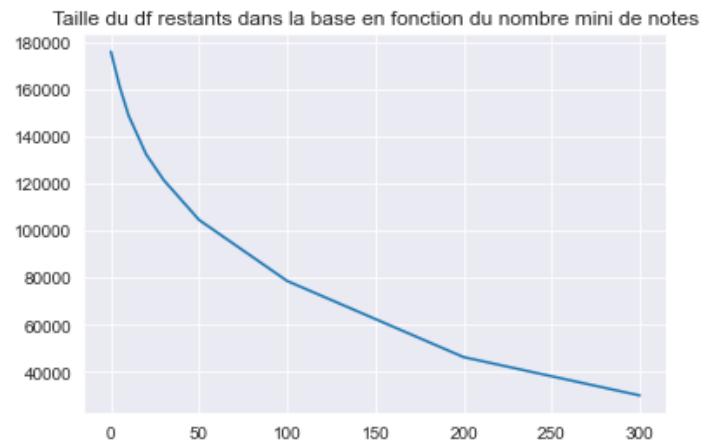


FIGURE 21 – Taille du df restant dans la base en fonction du nombre minimum de notes

Nous avons ensuite divisé notre base principale en quatre sous-bases distinctes, selon le nombre minimal de notes que les utilisateurs ont données. Ainsi, nous avons obtenu les sous-bases df_avis_10, df_avis_50, df_avis_100 et df_avis_200, respectivement constituées de 148961 lignes pour 2849 utilisateurs et 10284 jeux, 104510 lignes pour 734 utilisateurs et 9502 jeux, 78541 lignes pour 359 utilisateurs et 8884 jeux et 46182 lignes pour 128 utilisateurs et 7841 jeux. Ces sous-bases nous permettront de mener des analyses plus ciblées et plus spécifiques selon le nombre minimal de notes que les utilisateurs ont données. Voici ce qu'on

obtient :

Cas où les utilisateurs ont noté plus de 10 jeux.

- Nombre d'utilisateurs dans l'ensemble de la base : **2849**
- Nombre d'utilisateurs dans l'ensemble d'apprentissage en moyenne : **2842.8**
- Nombre d'utilisateurs dans l'ensemble de test en moyenne : **2385.2**

Cas où les utilisateurs ont noté plus de 50 jeux.

- Nombre d'utilisateurs dans l'ensemble de la base : **734**
- Nombre d'utilisateurs dans l'ensemble d'apprentissage en moyenne : **734**
- Nombre d'utilisateurs dans l'ensemble de test en moyenne : **734**

Cas où les utilisateurs ont noté plus de 100 jeux.

- Nombre d'utilisateurs dans l'ensemble de la base : **359**
- Nombre d'utilisateurs dans l'ensemble d'apprentissage en moyenne : **359**
- Nombre d'utilisateurs dans l'ensemble de test en moyenne : **358**

Cas où les utilisateurs ont noté plus de 200 jeux.

- Nombre d'utilisateurs dans l'ensemble de la base : **128**
- Nombre d'utilisateurs dans l'ensemble d'apprentissage en moyenne : **128**
- Nombre d'utilisateurs dans l'ensemble de test en moyenne : **128**

Avant toute chose, pour créer un système de recommandation nous avons également besoin de diviser les jeux déjà vus et les jeux susceptibles d'être appréciés par les utilisateurs en fonction de différents seuils de notes attribuées. On créé donc deux dictionnaires :

- `already_seen` : il s'agit des jeux que l'utilisateur a déjà notés et que nous ne voulons pas recommander à nouveau
- `ground_truth` : il s'agit des jeux que l'utilisateur est susceptible d'aimer avec une note supérieure ou égale à 10. Ce dictionnaire représente notre vérité pour évaluer nos prédictions.

4.3 SVD

L'algorithme SVD (Singular Value Decomposition) est une technique établie pour l'identification des facteurs sémantiques latents dans la recherche d'informations. Dans le domaine du filtrage collaboratif, l'application de SVD nécessite de factoriser la matrice de notation utilisateur-jeu.

Le calcul d'une prédiction est le suivant :

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

Dans cette équation, \hat{r}_{ui} représente la note attribuée par un utilisateur u à un élément i . μ est la note moyenne globale, b_u est l'effet de l'utilisateur u , b_i est l'effet de l'élément i , et q_i^T est le produit scalaire entre les vecteurs de facteurs latents de l'utilisateur u et de l'élément i .

4.3.1 Expériences et résultats

Après avoir chargé et initialisé les listes correspondant aux jeux notés par au moins 10, 50, 100, 200 utilisateurs, nous avons entraîné nos modèles basé sur la décomposition en valeurs singulières puis nous avons prédit les notes pour chaque jeu de test. En utilisant la liste de pertinence et la fonction MRR (Mean Reciprocal Rank) nous avons évalué les performances de recommandation. Plus précisément nous avons obtenus les résultats suivants :

- Avec les utilisateurs ayant noté au moins 10 jeux, nous avons obtenu une MRR moyenne de **34.8** sur une moyenne de 9679.6 jeux existants.
- Avec les utilisateurs ayant noté au moins 50 jeux, nous avons obtenu une MRR moyenne de **16.6** sur une moyenne de 8887.4 jeux existants.
- Avec les utilisateurs ayant noté au moins 100 jeux, nous avons obtenu une MRR moyenne de **10.6** sur une moyenne de 8261 jeux existants.
- Avec les utilisateurs ayant noté au moins 200 jeux, nous avons obtenu une MRR moyenne de **7.2** sur une moyenne de 7230.8 jeux existants.

En modifiant la valeur de référence `ground_truth` pour considérer les notes d'au moins 7 comme pertinentes pour les jeux, et non plus seulement les notes de 10, nous obtenons de meilleurs résultats. Les résultats améliorés sont les suivants :

- Avec les utilisateurs ayant noté au moins 10 jeux, nous avons obtenu une MRR moyenne de **30.0** sur une moyenne de 9679.6 jeux existants.
- Avec les utilisateurs ayant noté au moins 50 jeux, nous avons obtenu une MRR moyenne de **10.8** sur une moyenne de 8887.4 jeux existants.
- Avec les utilisateurs ayant noté au moins 100 jeux, nous avons obtenu une MRR moyenne de **6.6** sur une moyenne de 8261 jeux existants.
- Avec les utilisateurs ayant noté au moins 200 jeux, nous avons obtenu une MRR moyenne de **4.4** sur une moyenne de 7230.8 jeux existants.

En examinant les résultats, nous avons constaté que plus le seuil du nombre de jeux notés par les utilisateurs était élevé, plus les performances de recommandation s'améliorent en termes de MRR. Cela indique que les utilisateurs ayant noté un grand nombre de jeux ont tendance à recevoir des recommandations plus pertinentes.

On peut observer que dans les deux cas, en éliminant les utilisateurs ayant noté moins de jeux, les performances du modèle SVD sont améliorées en termes de

MRR. Cependant, les améliorations sont plus significatives lorsque nous utilisons uniquement les données avec des ratings supérieurs ou égaux à 7. En éliminant les utilisateurs ayant noté un faible nombre de jeux et en se concentrant sur ceux qui ont noté davantage de jeux, les performances de recommandation s'améliorent, ce qui suggère une meilleure adéquation entre les préférences des utilisateurs et les recommandations proposées. Cela suggère que le filtrage basé sur les ratings permet de mieux cibler les utilisateurs et les jeux les plus pertinents, ce qui conduit à de meilleures recommandations.

4.4 Popular items

Pour comparer nos résultats obtenus avec SVD, il nous faut implémenter d'autres algorithme. Nous avons donc choisi d'implémenter celui basé sur les jeux populaires, également connu sous le nom de "popular items recommendation algorithm". C'est une approche qui vise à recommander les jeux les plus populaires aux utilisateurs. Cette méthode repose sur l'idée que les jeux populaires ont tendance à être appréciés par un grand nombre d'utilisateurs et sont donc plus susceptibles de générer des recommandations pertinentes.

4.4.1 Distribution des jeux populaires

Nous avons commencé par calculer la popularité de chaque jeu en comptant le nombre de notes qu'il a reçu pour ensuite dresser une liste des jeux les plus populaires. Nous avons obtenus les jeux tels que le Uno, Cluedo, Shogun, Carcassonne, Civilisation, Les mystères de Pékin... Nous avons également créer des diagrammes des jeux populaires pour chaque base des utilisateurs ayant noté au moins 10, 50, 100, 200 jeux. Cependant nous avons remarqué que les distributions étaient relativement plate. Il y aurait donc une répartition relativement uniforme des notes et du nombre de botes parmi les jeux. Bien que cela soit étonnant, ce phénomène peut s'expliquer notamment par le biais des utilisateurs et la grande variété des jeux. En conséquence, les systèmes de recommandation peuvent être sensibles aux fluctuations de la popularité des jeux au fil du temps.

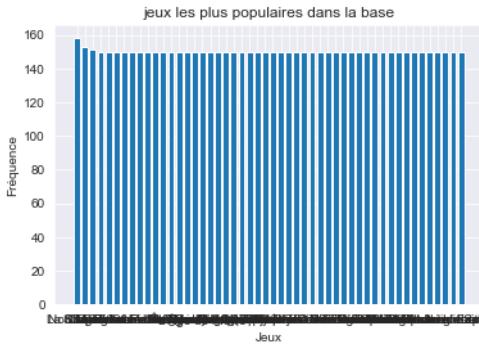


FIGURE 22 – Jeux les plus populaires dans la base des utilisateurs ayant noté au moins 10 jeux

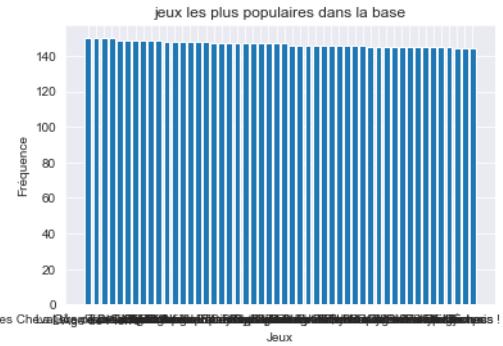


FIGURE 23 – Jeux les plus populaires dans la base des utilisateurs ayant noté au moins 50 jeux

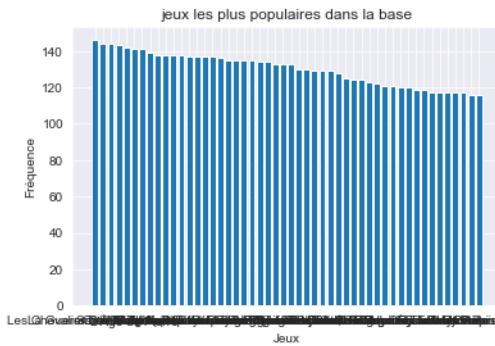


FIGURE 24 – Jeux les plus populaires dans la base des utilisateurs ayant noté au moins 100 jeux

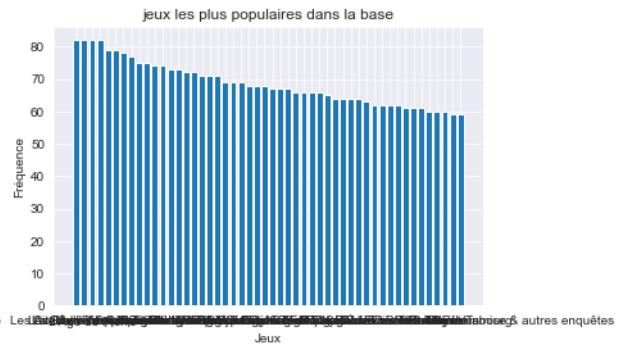


FIGURE 25 – Jeux les plus populaires dans la base des utilisateurs ayant noté au moins 200 jeux

4.4.2 Expériences et résultats

De la même manière que pour SVD, pour évaluer les performances de cet algorithme, nous avons divisé les utilisateurs en fonction du nombre de jeux qu'ils ont notés. En utilisant la mesure de pertinence MRR, nous avons évalué la qualité des recommandations fournies par l'algorithme.

Les résultats obtenus ont montré que plus le seuil du nombre de jeux notés par les utilisateurs était élevé, meilleures étaient les performances du système de recommandation en termes de MRR. Cela suggère que les utilisateurs ayant noté un plus grand nombre de jeux fournissent des informations plus fiables pour la prédiction et sont donc plus enclins à recevoir des recommandations pertinentes.

- En éliminant les utilisateurs ayant noter moins de 10 jeux nous avons une mrr de **51.6** en moyenne sur 9679.6.
- En éliminant les utilisateurs ayant noter moins de 50 jeux nous avons une mrr de **18.2** en moyenne sur 8887.4.
- En éliminant les utilisateurs ayant noter moins de 100 jeux nous avons une mrr de **8.0** en moyenne sur 8261.0
- En éliminant les utilisateurs ayant noter moins de 200 jeux nous avons une mrr de **5.0** en moyenne sur 7230.8.

En modifiant la valeur de référence `ground_truth` pour considérer les notes d'au moins 7 comme pertinentes pour les jeux, et non plus seulement les notes de 10, nous avons constaté des améliorations significatives des performances de recommandation.

- En éliminant les utilisateurs ayant noter moins de 10 jeux nous avons une mrr de **35.4** en moyenne sur 9679.6.
- En éliminant les utilisateurs ayant noter moins de 50 jeux nous avons une mrr de **10.2** en moyenne sur 8887.4.
- En éliminant les utilisateurs ayant noter moins de 100 jeux nous avons une mrr de **4.6** en moyenne sur 8261.0.
- En éliminant les utilisateurs ayant noter moins de 200 jeux nous avons une mrr de **3.0** en moyenne sur 7230.8.

En conclusion, l'algorithme de recommandation basé sur les jeux populaires offre une approche simple mais efficace pour générer des recommandations. Les jeux les plus populaires sont souvent des choix sûrs pour de nombreux utilisateurs, ce qui conduit à des recommandations potentiellement plus satisfaisantes. Par ailleurs on obtient de meilleures évaluations qu'avec SVD. Cependant, il est important de noter que cet algorithme présente également des limites, notamment en termes de personnalisation des recommandations pour les goûts individuels des utilisateurs. D'autres méthodes de recommandation plus avancées peuvent être explorées pour améliorer davantage les résultats.

4.5 LightFM

L'algorithme LightFM est utilisé pour effectuer une recommandation collaborative implicite. Contrairement à la recommandation collaborative explicite, l'approche implicite se base sur les interactions plutôt que sur les évaluations explicites.

LightFM est une implémentation en Python de plusieurs algorithmes de recommandation populaires, prenant en compte à la fois les feedbacks implicites

et explicites. Il inclut une implémentation efficace des pertes de classement BPR (Bayesian Personalized Ranking) et WARP (Weighted Approximate-Rank Pairwise). LightFM est facile à utiliser, rapide (grâce à l'estimation multi-thread) et produit des résultats de haute qualité.

Dans notre approche, nous utilisons l'implémentation BPR de LightFM pour effectuer la recommandation collaborative implicite. Le modèle se base sur une factorisation de la matrice, où une note est prédite de la manière suivante :

$$\hat{r}_{ui} = f(q_u p_i + b_u + b_i)$$

Nous considérons deux configurations pour les interactions :

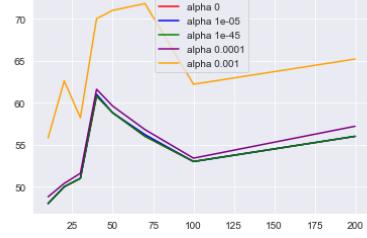
- Les interactions sont les notes supérieures ou égales à 10.
- Les interactions sont les notes supérieures ou égales à 7.

En utilisant ces deux configurations, nous entraînons le modèle BPR et évaluons ses performances en termes de Mean Reciprocal Rank (MRR) pour chaque seuil du nombre de jeux notés par les utilisateurs (10, 50, 100, 200). Le MRR mesure la qualité des recommandations en calculant la moyenne des inverses des rangs des jeux réellement pertinents dans les listes de recommandation. Pour chaque combinaison des paramètres alpha et du nombre de composantes, le modèle est entraîné sur les interactions d'entraînement (train_interactions) et évalué sur les interactions de test (test_interactions). Le but est de trouver la meilleure combinaison de paramètres (alpha et n_components) qui maximise le rang réciproque moyen. Pour chaque seuil on va chercher le meilleur modèle selon l'algorithme lightFM.

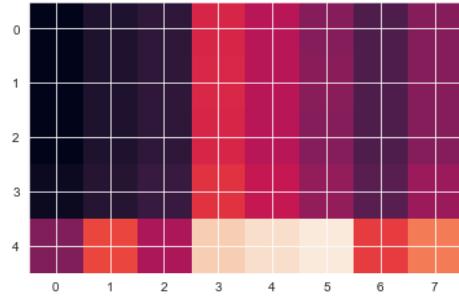
Pour les interactions comprenant les notes supérieures ou égales à 10

- Pour le seuil à 10, un des meilleurs modèles qu'on a obtenu a les paramètres suivants : alpha = 0, n_components = 10 et un rang réciproque moyen de 48.0.
- Pour le seuil à 50, un des meilleurs modèles qu'on a obtenu a les paramètres suivants : alpha = 0, n_component = 40 et un rang réciproque moyen de 10.2.
- Pour le seuil à 100 un des meilleurs modèles qu'on a obtenu a les paramètres suivants : alpha = 0, n_components = 70 et un rang réciproque moyen de 11.4.
- Pour le seuil à 200 un des meilleurs modèles qu'on a obtenu a les paramètres suivants : alpha = 0, n_components = 100 et un rang réciproque moyen de 6.0.

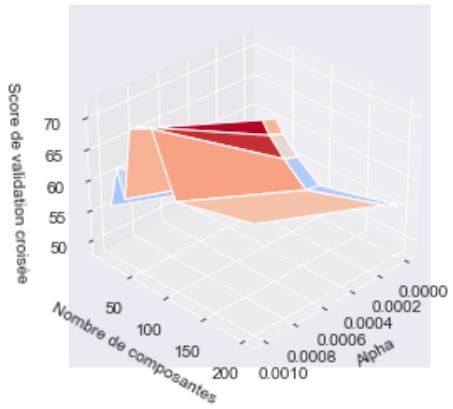
df_10 rel en fonction de ncomp pour des alpha dans [0, 1e-05, 1e-45, 0.0001, 0.001]



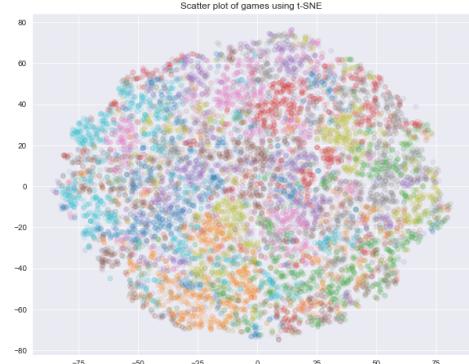
(a) Diagramme des courbes des moyennes des rangs réciproques



(c) Image des valeurs des moyennes des rangs réciproques

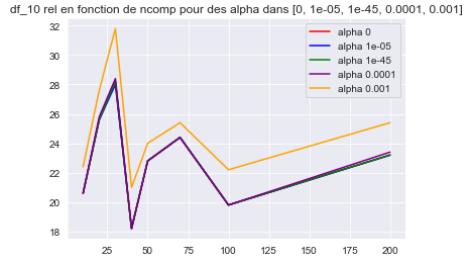


(b) Diagramme en 3D des scores de validation croisée

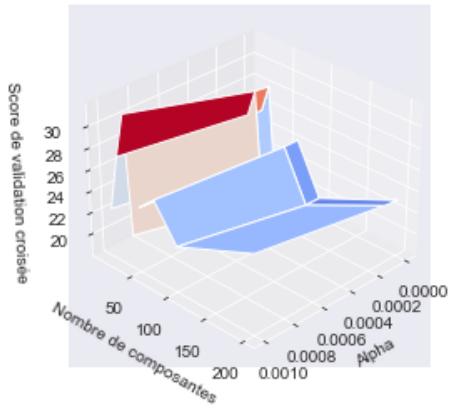


(d) TSNE

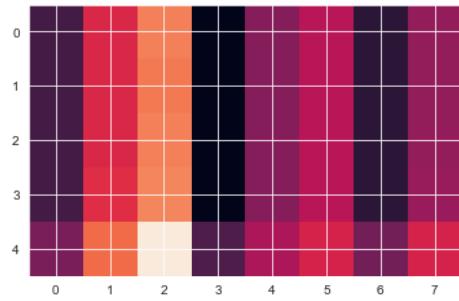
FIGURE 26 – Recherche des paramètres optimaux pour ground truth ≥ 10 pour $df_{avis} = 10$



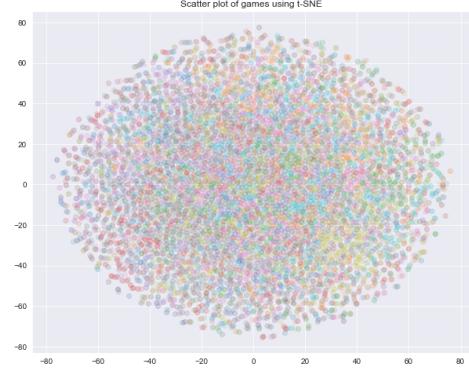
(a) Diagramme des courbes des moyennes des rangs réciproques



(b) Diagramme en 3D des scores de validation croisée

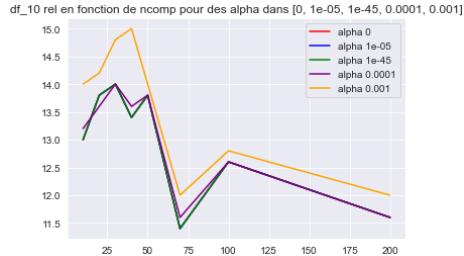


(c) Image des valeurs des moyennes des rangs réciproques

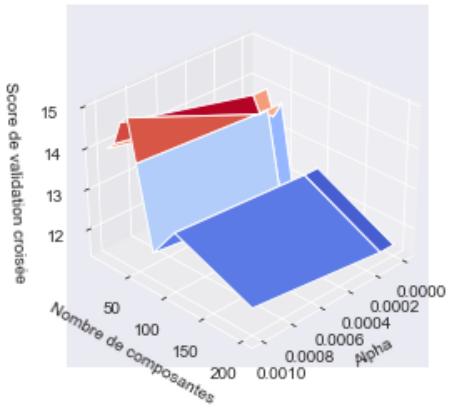


(d) TSNE

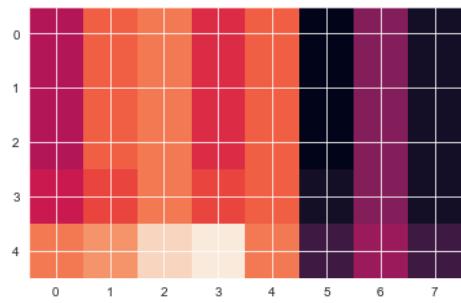
FIGURE 27 – Recherche des paramètres optimaux pour ground truth ≥ 10 pour $df_{avis} = 50$



(a) Diagramme des courbes des moyennes des rangs réciproques



(b) Diagramme en 3D des scores de validation croisée

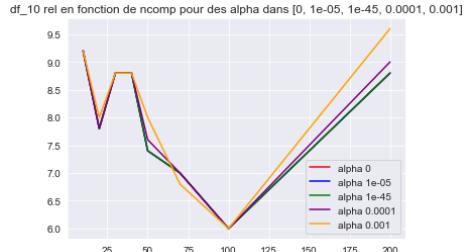


(c) Image des valeurs des moyennes des rangs réciproques

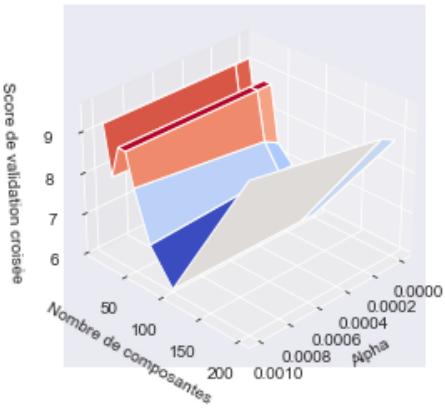


(d) TSNE

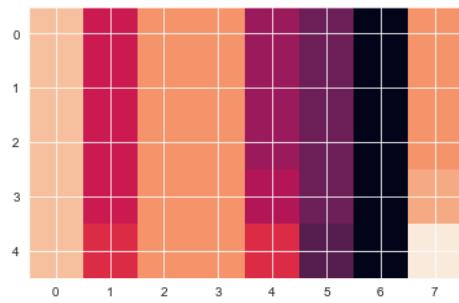
FIGURE 28 – Recherche des paramètres optimaux pour ground truth ≥ 10 pour $df_{avis} = 100$



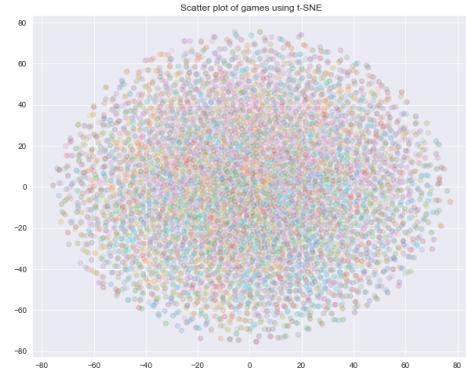
(a) Diagramme des courbes des moyennes des rangs réciproques



(b) Diagramme en 3D des scores de validation croisée



(c) Image des valeurs des moyennes des rangs réciproques

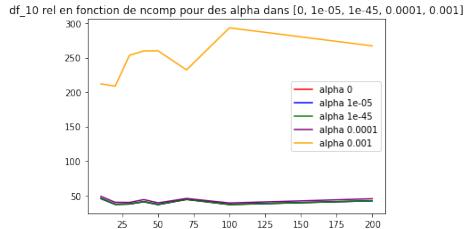


(d) TSNE

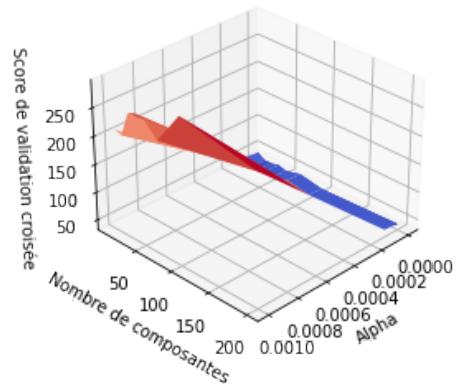
FIGURE 29 – Recherche des paramètres optimaux pour ground truth ≥ 10 pour $df_{avis} = 200$

Pour les interactions comprenant les notes supérieures ou égales à 7

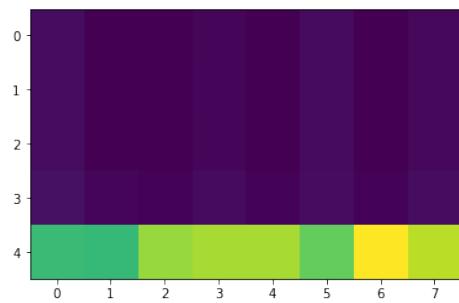
- Pour le seuil à 10, un des meilleurs modèles qu'on a obtenu a les paramètres suivants : alpha = 0, n_components = 50 et un rang réciproque moyen de 34.8.
- Pour le seuil à 50 un des meilleurs modèles qu'on a obtenu a les paramètres suivants : alpha = 0, n_components = 10 et un rang réciproque moyen de 9.8
- Pour le seuil à 100 un des meilleurs modèles qu'on a obtenu a les paramètres suivants : alpha = 0, n_components = 40 et un rang réciproque moyen de 5.0.
- Pour le seuil à 200 un des meilleurs modèles qu'on a obtenu a les paramètres suivants : alpha = 0, n_components = 30 et un rang réciproque moyen de 3.0.



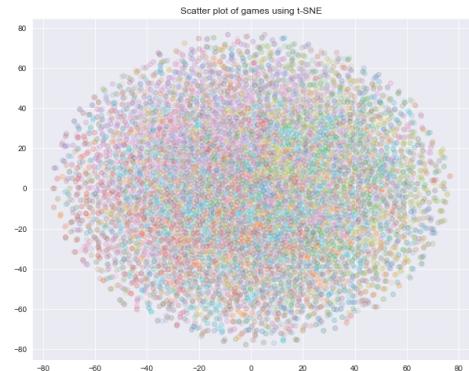
(a) Diagramme des courbes des moyennes des rangs réciproques



(b) Diagramme en 3D des scores de validation croisée

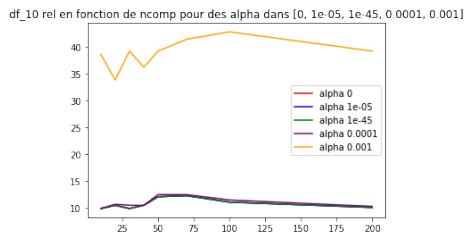


(c) Image des valeurs des moyennes des rangs réciproques

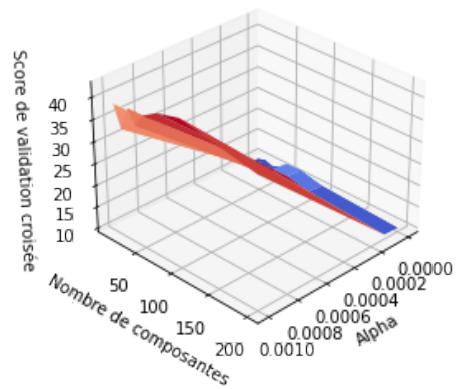


(d) TSNE

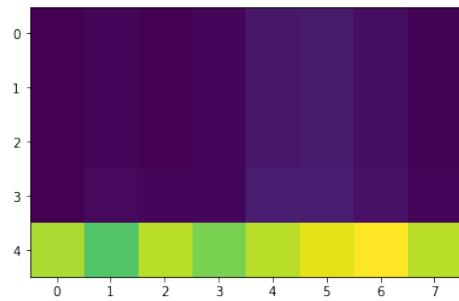
FIGURE 30 – Recherche des paramètres optimaux pour ground truth ≥ 7 pour $df_{avis} = 10$



(a) Diagramme des courbes des moyennes des rangs réciproques



(b) Diagramme en 3D des scores de validation croisée

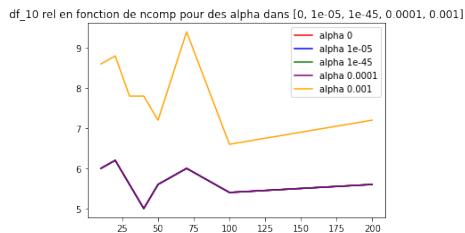


(c) Image des valeurs des moyennes des rangs réciproques

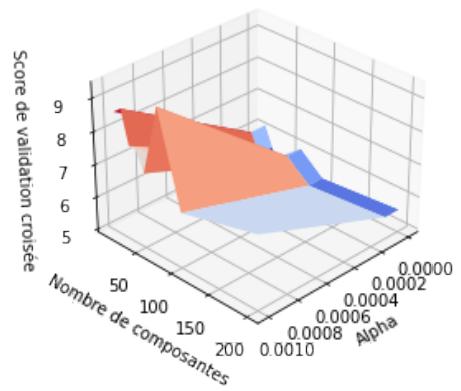


(d) TSNE

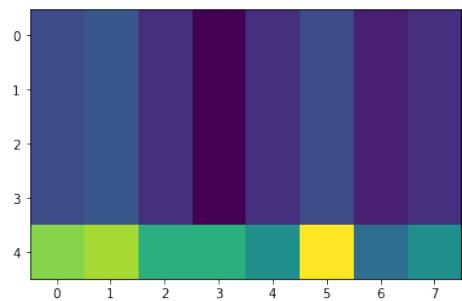
FIGURE 31 – Recherche des paramètres optimaux pour ground truth ≥ 7 pour $df_{avis} = 50$



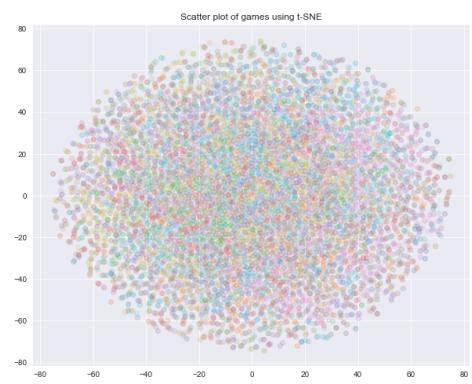
(a) Diagramme des courbes des moyennes des rangs réciproques



(b) Diagramme en 3D des scores de validation croisée

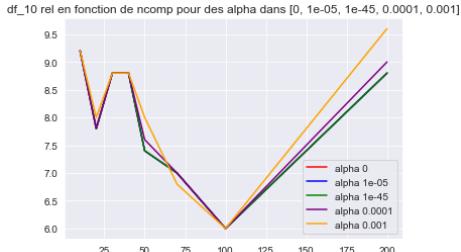


(c) Image des valeurs des moyennes des rangs réciproques

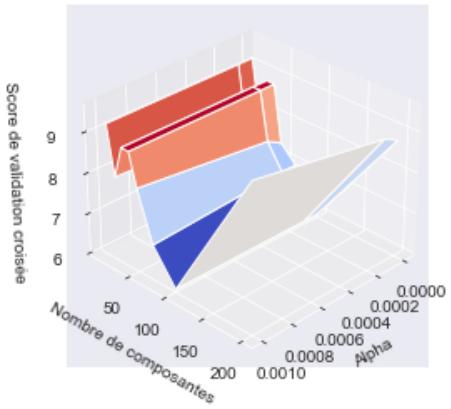


(d) TSNE

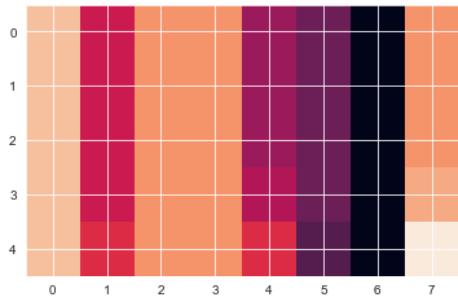
FIGURE 32 – Recherche des paramètres optimaux pour ground truth ≥ 7 pour $df_{avis} = 100$



(a) Diagramme des courbes des moyennes des rangs réciproques



(b) Diagramme en 3D des scores de validation croisée



(c) Image des valeurs des moyennes des rangs réciproques



(d) TSNE

FIGURE 33 – Recherche des paramètres optimaux pour ground truth ≥ 7 pour $df_{avis} = 200$

Les résultats montrent que le modèle BPR obtient de bonnes performances de recommandation implicite. Les MRR moyens diminuent à mesure que le seuil du nombre de jeux notés par les utilisateurs augmente, indiquant une amélioration des performances lorsque l'on se concentre sur les utilisateurs ayant fourni un plus grand nombre de préférences. Cela suggère que le modèle est capable de capturer les préférences implicites des utilisateurs et de fournir des recommandations pertinentes.

En conclusion, utilisant le modèle BPR de LightFM pour la recommandation collaborative implicite, nous exploitons les interactions entre les utilisateurs et les jeux plutôt que les évaluations explicites. Cela permet une recommandation plus flexible et polyvalente, adaptée à différents types d'interactions. Les performances

du modèle démontrent son efficacité dans la capture des préférences des utilisateurs et la génération de recommandations pertinentes.

5 Conclusion

En conclusion, notre projet a permis d'explorer la classification des sentiments et de comparer différents algorithmes de recommandation tels que SVD (Singular Value Decomposition) et Popular Items BPR (Bayesian Personalized Ranking). Fusionner l'analyse de sentiments avec les méthodes de recommandation afin de développer un système de recommandation plus personnalisé et précis est important car la recommandation basée uniquement sur les comportements d'achat ou de navigation ne tient pas vraiment en compte des préférences des utilisateurs. C'est pourquoi l'analyse de sentiments est une approche clé pour améliorer la précision des recommandations en prenant en compte les émotions et les sentiments des utilisateurs.

Cependant, nous n'avons pas eu l'opportunité de traiter la problématique de transfert de notre classifieur vers d'autres sources de données, comme Twitter par exemple. Ce serait une perspective intéressante pour étendre notre projet et explorer comment notre modèle de classification des sentiments peut être adapté à de nouveaux domaines ou types de données.

En ce qui concerne les perspectives d'amélioration futures, nous suggérons l'utilisation de Graph Neural Networks (Réseaux Neuronaux Graphiques). Ces réseaux sont capables de capturer les relations complexes entre les entités d'un réseau, ce qui pourrait améliorer la précision et la personnalisation des systèmes de recommandation en prenant en compte les émotions et les préférences des utilisateurs de manière plus complète.