UNIVERSITÉ DE SHERBROOKE

IFT 599-799 - Science des données

TP 3 - Systèmes de recommandation

Automne 2024

Réalisé par

	Courriel	Matricule
Achraf Haijoubi	Achraf.Haijoubi@USherbrooke.ca	24 014 567
Amadou Selle Ndiaye	Amadou.Selle.Ndiaye@usherbrooke.ca	23 233 207
Ayman Zouhair	Ayman.Zouhair@USherbrooke.ca	24 170 433

Enseignant

Pr. Shengrui Wang

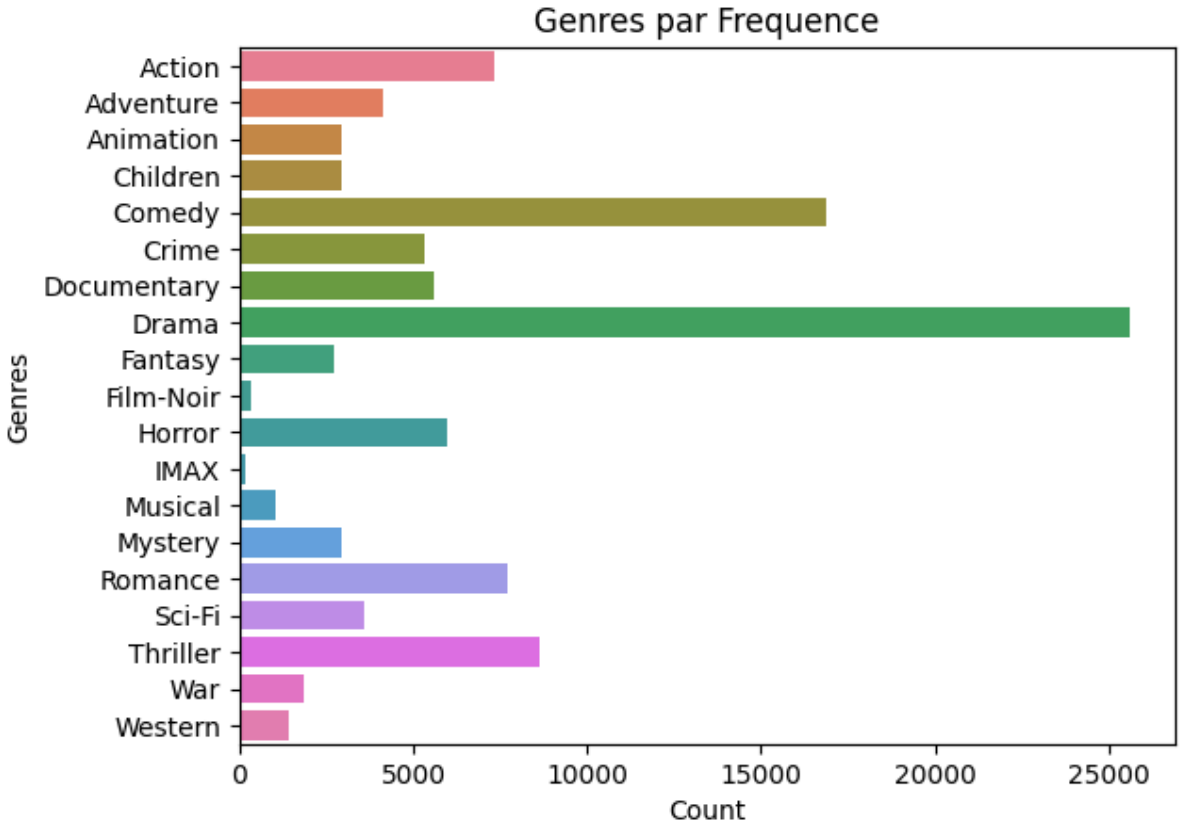
Table des matières

1 - Diagramme en bâton illustrant le nombre de films que l'on a par genre :

Méthodologie

La colonne **genres** du dataset ‘’movies.csv’’ a été analysée pour identifier les genres de films. Nous avons remarqué que plusieurs films avaient des genres composés hors ce serait plus correct d'explorer les genres par rapport à leur unicité.
Nous avons ainsi extrait tous les genres uniques en utilisant un **set()** après avoir parcouru la colonne **‘genres’** du dataset “**movies.csv**”. Les genres composés étaient séparés par un ’**|**’ nous obligeant à utiliser la méthode **.split()** pour la séparation. Ensuite, nous avons créé un dataframe ayant deux colonnes **Genres** dans lequel figure les genres uniques et **Count** qui contient la fréquence d'apparition des genres par rapport aux films en parcourant la colonne **‘genres’** de **“movies.csv”.**
Enfin, une visualisation des genres(en excluant les entrées sans genres (no genres listed)) a été réalisée à l'aide de la méthode .**barplot()** de Seaborn et Matplotlib.

Visualisation



Ce graphique aide à comprendre quelles catégories de films dominent la base de données, ce qui est utile pour orienter les analyses futures.

2 - Nettoyage : Extraction des jeux de données “movies_1.csv” et “ratings_1.csv” en ignorant les films avec un genre non listé :

Méthodologie

Pour le dataset “movies.csv”, nous avons juste supprimer toutes les lignes avec le label **‘(no genres listed)’** sur la colonne ‘**genres**’. Ensuite, nous avons enregistré le nouveau dataframe en utilisant la méthode
pandas de **to_csv()** en nommant le dataset généré **“movies_1.csv”**.

```
#Nouveau Dataframe sans les films avec genre non listés
data_movies_1.to_csv('movies_1.csv',index=False)
```

Pour le dataset “ratings.csv”, nous avons fait un filtre pour ne conserver que les utilisateurs ayant évalué au moins 5 000 films pour des questions de performance. Cela permet aussi de se concentrer sur des utilisateurs plus "actifs" susceptibles de fournir des données fiables pour la modélisation. Ainsi, nous sommes passés d’un shape de (25000095, 4) à un shape de (138974,4).

Pour enlever les ratings sur les genres non listés par contre, nous avons utilisé une sélection du dataframe “movies.csv” pour avoir les films avec un genre non listé. Ce dernier a été utilisé pour faire un merge avec “ratings.csv” sur la colonne “**movieId**”.

	userId	movieId	rating	timestamp	title	genres
0	187	1	3.5	1277374478	NaN	NaN
1	187	2	3.5	1277374864	NaN	NaN
2	187	3	3.0	1277839361	NaN	NaN
3	187	13	4.5	1277374351	NaN	NaN
4	187	19	4.5	1277373060	NaN	NaN
...	...	...	...	...	...	...
4192352	162516	194947	3.5	1551064100	NaN	NaN
4192353	162516	194951	4.0	1545794162	NaN	NaN
4192354	162516	195159	3.5	1571015852	NaN	NaN
4192355	162516	196997	3.5	1546313160	NaN	NaN
4192356	162516	199350	4.0	1553428563	NaN	NaN

4192357 rows × 6 columns

Après cette opération, nous avons reproduit la procédure de suppression effectuée sur “movies.csv” préalablement pour enlever les genres non listés.
Les colonnes non pertinentes (comme title et genres) ont aussi été supprimées du dataframe.

Enfin, nous avons effectué un mapping des ratings pour avoir des valeurs entières juste avant d'enregistrer le nouveau dataframe “ratings_1.csv” en utilisant la méthode **to_csv()** de pandas.

3 - Construction d’une base de données de films en contenu

Méthodologie

Tous les genres uniques ont été extraits de la colonne genres du dataset **movies_1**.

Certaines valeurs dans la colonne ‘genres’ sont une chaîne contenant plusieurs genres séparés par ” |”.La méthode **.split()** divise cette chaîne en une liste de genres individuels. Ces genres sont ajoutés dans un ensemble Python **(set)**, qui garantit que chaque genre n'est présent qu'une seule fois.

Ensuite, une matrice binaire est créée sous forme de dataframe avec des lignes correspondant à chaque **movieId** et des colonnes représentant les **genres uniques**.Tous les éléments de la matrice sont initialisés à **0**. Pour chaque film (chaque ligne du dataframe **data_movies_1**), on récupère les genres associés en divisant la chaîne de genres (.split('|')).

Ensuite, les colonnes correspondant à ces genres sont mises à 1 pour la ligne correspondant au movieId du film. Ce dataframe est appelé **“content_matrix”**

Exemple de données :

Un extrait du dataframe pourrait ressembler à ceci :

movieId	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	IMAX	Musical
1	0	1	1	1	1	0	0	0	1	0	0	0	0
2	0	1	0	1	0	0	0	0	1	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	1	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0	0

4 - Construire la matrice de profil des utilisateurs P

Méthodologie

Pour cette partie, nous avons commencé par créer le dictionnaire **user_profile.** Après, nous avons regroupé les ratings par utilisateur avec le dataset “**data_ratings_1**.**csv**”.Chaque itération traite un utilisateur (**user_id**) et les films qu'il a notés (**user_movies**).
Pour chaque utilisateur, le vecteur de profil (**profil_vector**) est calculé en multipliant leurs notes (**user_movies['rating']**) par les vecteurs de genres des films correspondants dans la

“content_matrix”. La multiplication produit une matrice où chaque ligne représente les genres pondérés par la note donnée par l'utilisateur.
Ensuite, la somme (**.sum(axis=0)**) des lignes donne un vecteur représentant la préférence de l'utilisateur pour chaque genre.
Ensuite, la norme du vecteur est calculée avec **np.linalg.norm(profile_vector)**.
Si la norme est supérieure à 0, le vecteur est normalisé en divisant par sa norme.Sinon, il est laissé tel.
Enfin, le dictionnaire **user_profile** associera chaque **userId** à un vecteur de profil normalisé basé sur leurs évaluations et les genres des films qu'ils ont notés avant d’être converti en un dataframe, où :

- Les indices (index) correspondent aux **userId**.
- Les colonnes correspondent aux **genres**.
- Les valeurs représentent les préférences pondérées et normalisées de chaque utilisateur pour chaque genre.

Exemple de données :

Un extrait du dataframe pourrait ressembler à ceci :

	Action	Adventure	Animation	Children	Comedy	Crime	\
187	0.359040	0.251590	0.089820	0.105782	0.484121	0.174398	
426	0.282674	0.150844	0.021972	0.034648	0.266618	0.260280	
541	0.295741	0.313502	0.257117	0.199604	0.339439	0.173103	
548	0.390475	0.172558	0.022309	0.048070	0.409086	0.240349	
626	0.270488	0.279734	0.070095	0.164056	0.519475	0.118258	
...	...	...	...	...	...	...	

5 - Préparation des données pour l’évaluation (et comparaison) des systèmes de recommandation développés

Pour évaluer et comparer les systèmes de recommandation, nous avons choisi la division simple des données. Cette division consiste à subdiviser le fichier ratings1.csv en deux ensembles distincts d’une façon aléatoire :

- 80% des données ont été attribuées au fichier d’apprentissage (**ratings apprendissage.csv**).
- 20% des données restantes ont été attribuées à un fichier d’évaluation (**ratings evaluation.csv**).

Ainsi, les deux fichiers obtenus contiennent **109013** pour l’ensemble d’apprentissage et **27254** pour l’ensemble d’évaluation.

6 - Filtrage collaboratif basé sur utilisateurs

Pour cette question, nous implémentons un système de filtrage collaboratif basé sur la similarité entre utilisateurs pour prédire les évaluations des films dans un ensemble de test. Pour chaque paire d'utilisateurs, nous calculons la similarité en utilisant la formule de corrélation de Pearsons et nous la stockons dans un dictionnaire user similarity , en prenant en compte uniquement les films évalués en commun. Ensuite, pour chaque évaluation à prédire dans l'ensemble de test, le code

calcule une estimation en effectuant une somme pondérée des évaluations des utilisateurs similaires, où les poids sont les valeurs de similarité entre les utilisateurs.

L'évaluation de la précision des prédictions se fait par le calcul de l'erreur quadratique moyenne **(RMSE)**, qui mesure la différence entre les évaluations estimées et les évaluations réelles. Une valeur plus faible du RMSE indique une meilleure performance du modèle.

Le **RMSE** obtenu est de **0.8243**, ce qui indique que le système propose des recommandations relativement proches des préférences des utilisateurs.

```
RMSE : 0.8243008575354488
```

7 - Filtrage basé sur contenu en utilisant le clustering :

Dans cette partie, nous appliquons l'algorithme de clustering **KMeans** sur les profils d'utilisateurs afin de regrouper ceux ayant des préférences similaires. Pour déterminer le nombre optimal de clusters **k**, nous combinons **KMeans** avec le coefficient de **silhouette**, qui mesure la qualité des regroupements. Cette approche consiste à effectuer plusieurs clusterings sur les profils d'utilisateurs et à identifier le **k** qui maximise le score de silhouette, assurant ainsi une séparation optimale des données. Une fois le **k optimal** déterminé (dans notre étude, **k=2**, correspondant à un score de silhouette de **0,32103**), les résultats du clustering sont exploités pour développer un système de recommandation basé sur le contenu. Les suggestions de films tiennent compte des préférences des utilisateurs appartenant à leur cluster respectif.

```
Meilleur nombre de clusters : 2
Scores de silhouette pour chaque k : [(2, 0.32103403775401357), (3, 0.3131990902145277),
```

```
(4, 0.2639469983207576), (5, 0.2474550645013966), (6, 0.2855881223753658), (7, 0.29245073561114243),
```

```
 (8, 0.23363578086579484), (9, 0.16831304349464424), (10, 0.1648440799008041)]
```

Après avoir déterminé les clusters à l'aide de l'algorithme **KMeans** (avec un **k optimal** de **2** dans notre cas), nous avons exploité ces regroupements pour générer des recommandations. Enfin, pour évaluer les performances de notre système de recommandation, nous nous sommes basés sur l'erreur quadratique moyenne (**RMSE**) qui repose sur la comparaison entre les notes réelles attribuées aux films et les notes moyennes des films recommandés dans chaque cluster. Le **RMSE** obtenu est de **1.0522**, ce qui indique que le système propose des recommandations relativement proches des préférences des utilisateurs.

```
RMSE du système de recommandation : 1.0522
```

Comparaison des deux systèmes de recommandation développés :

Dans cette étude, nous avons développé deux systèmes de recommandation : le premier repose sur un filtrage collaboratif basé sur les utilisateurs, et le second sur un filtrage basé sur le contenu utilisant le clustering. Nous avons ensuite comparé ces deux systèmes en termes de performances, en analysant les erreurs RMSE obtenues pour chacun, ainsi qu'en termes de temps de calcul.

1-Performances en termes de performances (RMSE) :
La recommandation collaborative a montré de meilleurs résultats en termes de performances, avec un RMSE de 0.8243, contre 1.0522 pour la recommandation basée sur le clustering.

2- Temps de calcul :
Le système de recommandation basé sur le clustering est plus rapide que l'approche collaborative. Une fois les clusters établis, les recommandations sont générées à partir de moyennes pré-calculées. En revanche, l'approche collaborative, bien qu'elle offre une meilleure précision, est plus coûteuse en termes de temps en raison de la construction de la matrice de similarité entre utilisateurs et des prédictions personnalisées.

8 - Proposition d'un algorithme de filtrage basé sur contenu, une approche simple (Bayésienne)

Nous avons opté pour l'approche bayésienne parce qu'elle permet d'incorporer des connaissances préexistantes (probabilités a priori) dans les calculs.
Ces probabilités peuvent provenir d'expertises, d'analyses passées, ou d'une base de données historique.
Cela est utile lorsque le dataset est incomplet ou déséquilibré.
Exemple : Si nous savons qu'un utilisateur aime principalement les genres "Action" et "Science-Fiction", l'approche bayésienne peut intégrer cette information pour affiner les recommandations.

Dans ce contexte, ceci représente notre exemple pour cette approche:

Entrées :
- Vecteur des genres du nouveau film (1×d)
- Dataset des ratings : userId, movieId, rating
- Matrice binaire des genres des films (content_matrix)
- Profils utilisateurs : user_profiles

Sorties :
- Liste des utilisateurs susceptibles d'aimer le film

a. Créer les profils utilisateurs (`user_profiles`) Pu :
   - Grouper le dataset des ratings par `userId`
   - Calculer un vecteur pondéré des genres pour chaque utilisateur en fonction des films notés

- Normaliser chaque vecteur utilisateur

b. Calculer la probabilité pour chaque utilisateur :
Pour chaque utilisateur `u` :
i. Appliquer la formule bayésienne pour estimer

$$P(ru, i|Pu) = P(ru, i) \prod_{j \epsilon xi} P(xi, j|ru, i)$$

ii. Stocker le résultat.

c. Filtrer les utilisateurs :
- Comparer les résultats obtenus pour chaque utilisateur à un seuil (ou sélectionner le top-N).
- Construire une liste des utilisateurs intéressés.

d. Retourner la liste des utilisateurs intéressés.

RÉFÉRENCES

Références théoriques

IFT599_799_Thème_6_partie 1

IFT599_799_Thème_6_partie2

Références Programmation

seaborn.barplot — seaborn 0.13.2 documentation [1]
pandas.DataFrame.merge — pandas 2.2.3 documentation [2]
pandas 2.2.3 documentation [3]
Comment enregistrer un DataFrame Pandas au format CSV ? | DataCamp [4]