Faculté des Sciences de Sfax

Rapport de Projet

Recommandation Basée sur la Similarité
(Approche centrée item)

Par

Ahmedou Yahye Gleiguem Kheyri

étudiant en Master Recherche Informatique

25 février 2025

Table des matières

1 Introduction

Ce rapport présente l'analyse d'un systÃĺme de recommandation basé sur la similarité entre films. Nous utilisons des évaluations de films ainsi que leurs caractéristiques pour prédire les notes manquantes. Le document est découpé en plusieurs ETAPEs :
— Chargement des données.
— Fusion des données.
— Création de la matrice utilisateur-film.
— Normalisation des notes.
— Calcul des similarités (Pearson et Cosinus).
— Prédiction des notes et comparaison des méthodes.
— Analyse finale et conclusion.

2 Presentation Mathematique

2.1 Normalisation des Notes

Pour reduire le biais des utilisateurs, on centre les notes autour de la moyenne :

$$r_{norm}(u,i) = r(u,i) - \bar{r_i} \tag{1}$$

ou $\bar{r_i}$ est la moyenne des notes pour l'item i.

2.2 Mesures de Similarite

— **Pearson** : Mesure la correlation lineaire entre les vecteurs centres

$$sim_p(A,B) = \frac{\sum(a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum(a_i - \bar{a})^2}\sqrt{\sum(b_i - \bar{b})^2}} \tag{2}$$

— **Cosinus** : Mesure l'angle entre les vecteurs

$$sim_c(A,B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{3}$$

3 Présentation des Données

Les fichiers utilisés sont :

movies-Rene (1).csv

```
movieId;title;genres
1;Item1;"Adventure|Animation|Children|Comedy|Fantasy"
2;Item2;"Adventure|Children|Fantasy"
3;Item3;"Comedy|Romance"
4;Item4;"Adventure|Children|Fantasy"
```

rating-Rene (1).csv

```
userId;movieId;rating;timestamp
1;1;;964982703
1;2;2;964981247
1;3;7;964981247
1;4;8;964981247
2;1;4;964982224
2;2;1;964982224
2;3;;964982224
2;4;7;964982224
3;1;3;964983815
3;2;8;964983815
3;3;;964983815
3;4;4;964983815
4;1;4;964982931
4;2;1;964982931
4;3;6;964982931
4;4;;964982931
```

4 ETAPE 1 : Chargement des Données

Objectif : Importation des evaluations utilisateurs et des metadonnees des films. Les donnees brutes contiennent des valeurs manquantes (NaN) qu'il faudra traiter.

4.1 Code

```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

```

```python
# ----------------------------
# ETAPE 1 : Chargement des donn\ées
# ----------------------------
print("\n[ETAPE 1/7] Chargement des donn\ées...")
ratings = pd.read_csv('rating-Rene (1).csv', sep=';',
    na_values=[''])
movies = pd.read_csv('movies-Rene (1).csv', sep=';')

print(f"* {len(ratings)} \évaluations charg\ées")
print(f"* {len(movies)} films dans le catalogue")
print("Apercu des \évaluations :")
print(ratings.head())
```

Listing 1 – Chargement des données

4.2 Output

Output - Etape 1

[ETAPE 1/7] Chargement des données...
* 16 évaluations chargées
* 4 films dans le catalogue
Apercu des évaluations :

	userId	movieId	rating	timestamp
0	1	1	NaN	964982703
1	1	2	2.0	964981247
2	1	3	7.0	964981247
3	1	4	8.0	964981247
4	2	1	4.0	964982224

5 ETAPE 2 : Fusion des Données

Processus : Jointure des evaluations avec les metadonnees des films via l'identifiant movieId. Permet d'enrichir les donnees d'evaluation avec les genres des films.

5.1 Code

```python
print("\n[ETAPE 2/7] Fusion des donn\ées...")
df = pd.merge(ratings, movies, on='movieId', how='inner')
print(f"* Dataset fusionn\é : {df.shape[0]} lignes")
```

```
print("Colonnes disponibles :", list(df.columns))
```

Listing 2 – Fusion des données

5.2 Output

Output - ETAPE 2

[ETAPE 2/7] Fusion des données...
* Dataset fusionné : 16 lignes
Colonnes disponibles : ['userId', 'movieId', 'rating', 'timestamp', 'title', 'genres']

6 ETAPE 3 : Création de la Matrice Utilisateur-Film

Structure : Creation d'une matrice creuse de taille $n_{utilisateurs} \times n_{films}$ ou chaque cellule contient la note attribuee. Le taux de remplissage initial est de 75%.

$$M = \begin{bmatrix} - & 4 & 3 & 4 \\ 2 & 1 & 8 & 1 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4}$$

6.1 Code

```
print("\n[ETAPE 3/7] Cr\éation de la matrice utilisateur-film ...")
matrix = df.pivot_table(index='title', columns='userId', values='rating')
print("\nMatrice brute (5 premières lignes) :")
print(matrix.head())
print(f"\n* Dimensions : {matrix.shape[0]} films x {matrix.shape[1]} utilisateurs")
print(f"* Taux de remplissage : {(1 - matrix.isna().mean().mean()):.1%}")
```

Listing 3 – Création de la matrice utilisateur-film

6.2 Output

Output - ETAPE 3

[ETAPE 3/7] Création de la matrice utilisateur-film...

Matrice brute (5 premiÃÍres lignes) :

	1	2	3	4
Item1	NaN	4.0	3.0	4.0
Item2	2.0	1.0	8.0	1.0
Item3	7.0	NaN	NaN	6.0
Item4	8.0	7.0	4.0	NaN

* Dimensions : 4 films x 4 utilisateurs
* Taux de remplissage : 75.0%

7 ETAPE 4 : Normalisation des Notes

Impact : Reduit le biais systematique entre utilisateurs. Permet une comparaison plus equitable entre les films.
Exemple de normalisation :

$$M_{norm} = M - \bar{M} = \begin{bmatrix} - & 0.33 & -0.66 & 0.33 \\ -1 & -2 & 5 & -2 \end{bmatrix} \tag{5}$$

7.1 Code

```
print("\n[ETAPE 4/7] Normalisation des notes...")
matrix_norm = matrix.subtract(matrix.mean(axis=1), axis=0)
print("\nMatrice normalis\ée (exemple) :")
print(matrix_norm.loc[['Item1', 'Item2'], [1, 4]])
print("\nExplication : Les notes sont centr\ées autour de la
    moyenne de chaque film")
```

Listing 4 – Normalisation des notes

7.2 Output

Output - ETAPE 4

[ETAPE 4/7] Normalisation des notes...

Matrice normalisée (exemple) :

	1	4
Item1	NaN	0.33
Item2	-1.00	-2.00

Explication : Les notes sont centrées autour de la moyenne de chaque film

8 ETAPE 5 : Calcul des Similarités

Comparaison :
— Pearson sensible aux tendances lineaires
— Cosinus sensible Ãă l'amplitude des notes
Resultats typiques :

$$S_{Pearson} = \begin{bmatrix} 1 & -0.93 \\ -0.93 & 1 \end{bmatrix}, \quad S_{Cosinus} = \begin{bmatrix} 1 & 0.40 \\ 0.40 & 1 \end{bmatrix} \tag{6}$$

8.1 Code

```
print("\n[ETAPE 5/7] Calcul des similarit\és entre films...")

# M\éthode Pearson
print("\nCalcul des corr\élations de Pearson...")
item_similarity_pearson = matrix_norm.T.corr()
print("Matrice de similarit\é Pearson (extrait) :")
print(item_similarity_pearson.head())

# M\éthode Cosinus
print("\nCalcul des similarit\és cosinus...")
matrix_norm_zero = matrix_norm.fillna(0)
item_similarity_cosine = pd.DataFrame(
    cosine_similarity(matrix_norm_zero),
    index=matrix_norm.index,
    columns=matrix_norm.index
)
print("Matrice de similarit\é Cosinus (extrait) :")
print(item_similarity_cosine.head())
```

Listing 5 – Calcul des similarités entre films

8.2 Output

Output - ETAPE 5

[ETAPE 5/7] Calcul des similarités entre films...

Calcul des corrélations de Pearson...
Matrice de similarité Pearson (extrait) :

	Item1	Item2	Item3	Item4
Item1	1.00	-1.00	1.00	-0.93
Item2	-1.00	1.00	-1.00	0.93
Item3	1.00	-1.00	1.00	-0.93
Item4	-0.93	0.93	-0.93	1.00

Calcul des similarités cosinus...
Matrice de similarité Cosinus (extrait) :

	Item1	Item2	Item3	Item4
Item1	1.00	-0.98	0.74	-0.85
Item2	-0.98	1.00	-0.29	0.40
Item3	0.74	-0.29	1.00	-0.85
Item4	-0.85	0.40	-0.85	1.00

9 ETAPE 6 : Prédictions des Notes et Comparaison

Methodologie : Pour chaque film non note, on calcule une moyenne ponderee des notes des films similaires :

$$\hat{r}_{u,i} = \bar{r_i} + \frac{\sum_{j \in N(i)} sim(i,j) \cdot (r_{u,j} - \bar{r_j})}{\sum_{j \in N(i)} |sim(i,j)|} \tag{7}$$

ou $N(i)$ sont les voisins les plus proches de l'item i.

9.1 Code

```python
def item_based_recommendation(user_id, similarity_matrix, method_name):
    print(f"\nAnalyse pour l'utilisateur {user_id} ({method_name})")
    user_ratings = matrix_norm[user_id]

    # Films non not\és
    unwatched = user_ratings[user_ratings.isna()].index
    print(f"\n* Films non \évalu\és : {list(unwatched)}")

    predictions = []
    for movie in unwatched:
        if movie not in similarity_matrix:
            continue
        # Calcul des similarit\és
        sim_scores = similarity_matrix[movie].reset_index(name='similarity')
        merged = pd.merge(user_ratings.dropna().reset_index(), sim_scores, on='title')

        if not merged.empty:
            top_sims = merged.nlargest(2, 'similarity')
            predicted = np.average(top_sims[user_id], weights=top_sims['similarity'])
            predictions.append((movie, predicted))

    return sorted(predictions, key=lambda x: x[1], reverse=True)[:3]

# Comparaison des m\éthodes
print("\n[ETAPE 6/7] Comparaison des m\éthodes...")

# M\éthode Pearson
print("\n" + "="*50)
reco_pearson = item_based_recommendation(4, item_similarity_pearson, "Pearson")
print("\nR\ésultats Pearson :")
for film, note in reco_pearson:
    note_reelle = note + matrix.mean(axis=1)[film]
    print(f" {film :<45} : {note:.2f} (norm)  {note_reelle:.2f}/5")

# M\éthode Cosinus
print("\n" + "="*50)
reco_cosine = item_based_recommendation(4, item_similarity_cosine, "Cosinus")
print("\nR\ésultats Cosinus :")
for film, note in reco_cosine:
```

```
40      note_reelle = note + matrix.mean(axis=1)[film]
41      print(f" {film} : {note:.2f} (norm)  {note_reelle:.2f}/5"
            )
```

Listing 6 – Prédictions des notes et comparaison

9.2 Output

Output - ETAPE 6

[ETAPE 6/7] Comparaison des méthodes...

==========

Analyse pour l'utilisateur 4 (Pearson)

* Films non évalués : ['Item4']

Résultats Pearson :
Item4 : 31.57 (norm) 37.91/5

==========

Analyse pour l'utilisateur 4 (Cosinus)

* Films non évalués : ['Item4']

Résultats Cosinus :
Item4 : 0.04 (norm) 6.37/5

10 ETAPE 7 : Analyse Finale

10.1 Code et Output d'Analyse Finale

```
1 print("\n[ETAPE 7/7] Analyse finale :")
2 print("\nObservations cl\és :")
3 print("1. La m\éthode Pearson donne g\én\éralement des pr\é
      dictions plus \élev\ées")
4 print("2. Le cosinus est plus conservateur car sensible aux
      notes nulles")
5 print("3. Item4 est fortement recommand\é dans les deux m\é
      thodes")
6 print("\nConclusion : La corr\élation de Pearson semble mieux
       adapt\ée")
7 print("pour ce jeu de donn\ées clairsem\é avec normalisation"
      )
```

Listing 7 – Analyse finale

10.2 Output

Output - ETAPE 7

[ETAPE 7/7] Analyse finale :

11 Comparaison et Perspectives d'Amélioration

11.1 Comparaison des Méthodes

La corrélation de Pearson apparaÃőt comme mieux adaptée pour ce jeu de données, car la normalisation permet de centrer les notes et de mettre en évidence les relations linéaires entre les films. La méthode cosinus, quant a elle, est plus sensible aux valeurs manquantes.
Observations clés :
1. La méthode Pearson donne généralement des prédictions plus élevées
2. Le cosinus est plus conservateur car sensible aux notes nulles
3. Item4 est fortement recommandé dans les deux méthodes

11.2 Perspectives d'Amélioration

— Explorer d'autres techniques de normalisation ou de pondération des notes.
— Augmenter la taille du jeu de données pour améliorer la robustesse des prédictions.
— Tester des méthodes hybrides pour pallier les limites de chaque approche.

11.3 Conclusion

Conclusion : L'approche basee sur la correlation de Pearson montre des resultats prometteurs pour des donnees clairsemees, mais une approche hybride permettrait de beneficier des avantages des deux methodes.

A Code Complet (vertion plus pratique)

```python
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

# Chargement des donnees
ratings = pd.read_csv('rating-Rene (1).csv', sep=';',
    na_values=[''])
movies = pd.read_csv('movies-Rene (1).csv', sep=';')

# Fusion des donnees
df = pd.merge(ratings, movies, on='movieId', how='inner')

# Creation de la matrice utilisateur-film
matrix = df.pivot_table(index='title', columns='userId',
    values='rating')

# Normalisation de la matrice
matrix_norm = matrix.subtract(matrix.mean(axis=1), axis=0)

# Calcul des similarites
# Pearson
item_similarity_pearson = matrix_norm.T.corr()

# Cosine (remplacer NaN par 0)
matrix_norm_zero = matrix_norm.fillna(0)

item_similarity_cosine = pd.DataFrame(
    cosine_similarity(matrix_norm_zero),
    index=matrix_norm.index,
    columns=matrix_norm.index
)

# Fonction de recommandation generique
def item_based_recommendation(user_id, similarity_matrix,
    n_similar=2, n_reco=3):
    user_ratings = matrix_norm[user_id]
    unwatched_movies = user_ratings[user_ratings.isna()].
        index

    predictions = []

    for movie in unwatched_movies:
        if movie not in similarity_matrix:
            continue

```

```
        # Recuperer les similarites (Similarites du film
            cible avec les autres films)
        sim_scores = similarity_matrix[movie].reset_index(
            name='similarity')

        # Fusionner avec les films notes par l'utilisateur
        merged = pd.merge(
            user_ratings.dropna().reset_index(),
            sim_scores,
            on='title'
        )

        if merged.empty:
            continue

        # Filtrer les similarites non-nulles
        valid_sims = merged[merged['similarity'].notna()]
        if len(valid_sims) < 1:
            continue

        # Top N similarites
        top_sims = valid_sims.nlargest(n_similar, 'similarity
            ')

        # Prediction ponderee
        predicted_rating = np.average(top_sims[user_id],
            weights=top_sims['similarity'])
        predictions.append((movie, predicted_rating))

    # Tri et retour
    predictions.sort(key=lambda x: x[1], reverse=True)
    return predictions[:n_reco]

# Comparaison pour l'utilisateur 4
print("=== Resultats Pearson ===")
reco_pearson = item_based_recommendation(4,
    item_similarity_pearson)
for film, note in reco_pearson:
    print(f"- {film} : {note:.3f} (norm) / {note + matrix.
        mean(axis=1)[film]:.3f} (reel)")

print("\n=== Resultats Cosinus ===")
reco_cosine = item_based_recommendation(4,
    item_similarity_cosine)
for film, note in reco_cosine:
    print(f"- {film} : {note:.3f} (norm) / {note + matrix.
        mean(axis=1)[film]:.3f} (reel)")
```

Listing 8 – Code complet de l'implémentation