L3 MIAGE – Java avancée – projet individuel

Modalités du rendement

- Un seul fichier zip avec l'ensemble de code finalement envoyer un fichier zip de format : *[votre_prénom]_[votre_nom]_projetJava.zip*
- La date limite est 6/12/2020 à 23h59.

Le projet consiste à faire une interface d'une version avancée de la base de données Movielens.

La base de données :

Dans cette base on trouve les tableaux suivants :

movie

Ce tableau contient les informations des films (le titre, l'année de production, …).

people

Ce tableau contient les noms des personnes qui travaillent dans le cinéma (acteurs, réalisateurs, …).

role

ce tableau fait le lien entre des personnes du tableau (people), et des films de tableau (movie), avec le rôle de chaque personne dans le film. Par exemple, la requête SQL suivante :

SELECT * FROM role INNER JOIN people ON people.peo_id = role.peo_id WHERE mov_id = 150;

Donne les noms de toutes les personnes qui ont travaillé dans le film numéro 150 avec le rôle de chacun, comme dans la figure en bas. Remarquez que rien n'empêche plusieurs personnes d'avoir le même rôle dans un film, ni une personne d'avoir plusieurs rôles dans le même film.

rol_id	rol_name	mov_id	peo_id	peo_id	peo_name
17558	director	150	889	889	ron howard
17559	writer (book)	150	890	890	jim lovell
17560	writer (book)	150	891	891	jeffrey kluger
17561	writer (screenplay)	150	892	892	william broyles jr.
17562	writer (screenplay)	150	893	893	al reinert
17563	actors	150	8	8	tom hanks
17564	actors	150	894	894	bill paxton
17565	actors	150	100	100	kevin bacon
17566	actors	150	895	895	gary sinise

Figure 1 le tableau role

tags

Ce tableau contient un ensemble de mots sélectionnés par des internautes pour être pertinents à décrire les contenus des films. Il y a en totale 1128 mots.

tags_relevance

Ce tableau forme un lien entre les mots clés de tableau tags et les films de tableau movie. Chaque ligne contient l'identifiant d'un film, l'identifiant d'un tag, et un score de « relevance » qui détermine à quel point ce tag est représentatif du film. Les valeurs de scores varient entre 0 et 1, ces scores ont été calculé par des experts du domaine qui ont collecté les avis des spectateurs ayant regardé les films. Nous n'allons pas entrer dans les détails de calcul, tous ce qui nous intéresse est de savoir qu'un score proche de 1 relève de la pertinence du terme à un film.

La figure suivante montre les scores pertinence de trois tags (3D, Alaska, alien) au film « toy story ». On remarque que le terme **3D** est représentatif pour ce film, **alien** beaucoup moins (la notion d'espace est légèrement abordée avec le personnage de Buzz Lightyear), et **Alaska** pas de tout. On peut dire que la pertinence commence à avoir de sens à partir de la valeur 0.1.

title	tag_word	relevance
toy story	3d	0.563
toy story	alaska	0.014
toy story	alien	0.13

Figure 2 tag pertinence

Le figure en bas regroupe l'ensemble de tableaux de la base.

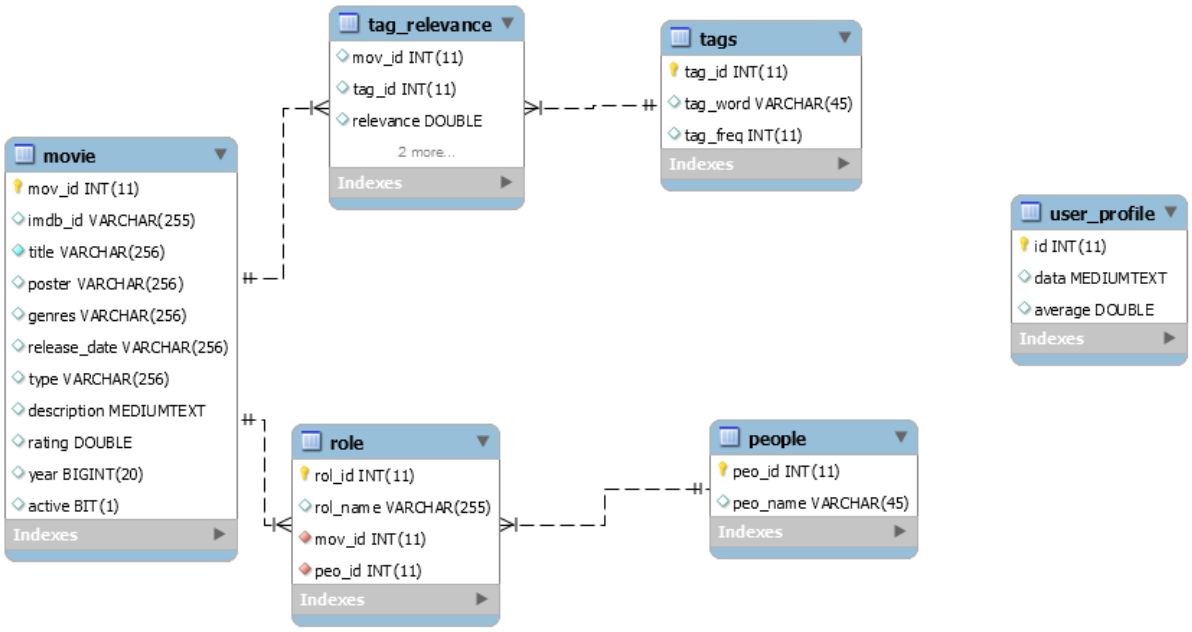


Figure 3 L'ensemble de la base de données.

Le projet :

Le projet consiste à utiliser la base de données pour faire trois interfaces graphiques :

Affiche de film :

Il s'agit d'une interface graphique pour afficher les informations d'un film, cette interface doit contenir :

- Le titre du film
- L'affiche (si une image existe)
- Les genres
- Le synopsis
- La note moyenne (rating)
- L'année de production
- Les personnes qui ont participé à la réalisation du film et leurs rôles respectifs.

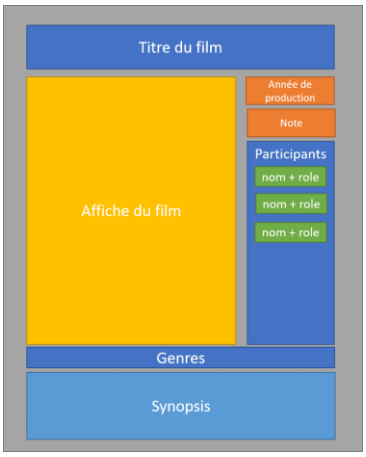


Figure 4 exemple de l'interface film

Les noms de participants doivent être cliquables. Cliquer sur le nom d'un participant nous amène vers une interface graphique qui regroupe ses films.

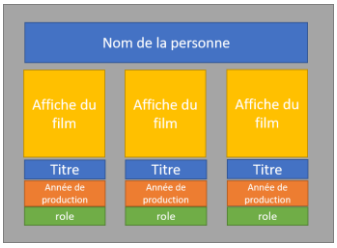


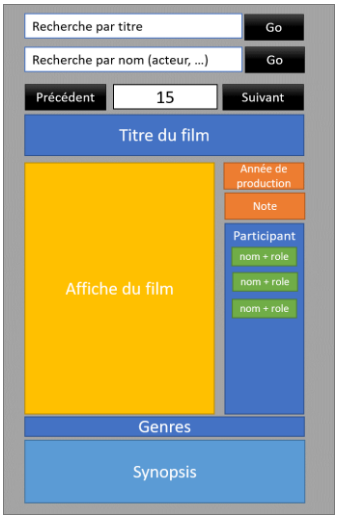
Figure 5 Interface d'un personnage

Quand on clique sur l'un des films de figure 5, on a le film qui s'affiche dans une figure 4.

Interface principale et barres de recherche :

Ceci est l'interface principale. Dans cette interface on affiche un film avec plusieurs fonctionnalités supplémentaires :

- Le bouton suivant qui permet d'afficher le film suivant
- Le bouton précédent qui permet d'afficher le film précédent.
- L'edit text de numéro du film qui permet d'accéder directement à un film en tapant son numéro suivi par la touche entrée.
- La barre de recherche par titre qui permet de cherche un film par son titre. Si la recherche trouve plusieurs titres correspondants au texte écrit, il affiche une liste permettant à l'utilisateur de choisir le film qu'il voudrait afficher.



Algorithme pour trouver un film à regarder :

Avant d'aller dans les détails de l'algorithme, il faut comprendre et développer une méthode de calcul de similarité entre films :

Comment calculer la similarité entre films

Grace au système de tags et de leur pertinence expliqué dans la partie de base de données, on peut considérer un film comme un vecteur de tags. Alors, pour calculer la similarité entre deux films il suffit

de calculer l'angle entre leurs vecteurs. Autrement dit, calculer la similarité cosinus entre les deux films selon l'équation :

$$\cos\theta = \frac{A.B}{\|A\|.\|B\|} = \frac{\sum_{i=1}^{n} A_i.B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}.\sqrt{\sum_{i=1}^{n} {B_i}^2}}$$

Par exemple on va prendre 3 films avec 3 tags pour montrer comment on calcule et on interprète cette similarité :

Film / tag	**3d**	**Alaska**	**Alien**
Toy Story	0.563	0.014	0.13
Balto	0.083	0.856	0.014
A bug's life	0.435	0.016	0.032

La similarité entre « toy story » est « Balto » :

$$cos\theta = \frac{(0.563 \times 0.083) + (0.014 \times 0.856) + (0.13 \times 0.014)}{\sqrt{(0.563)^2 + (0.014)^2 + (0.13)^2}.\sqrt{(0.083)^2 + (0.856)^2 + (0.014)^2}}$$
$$= \frac{0.060533}{0.58 \times 0.86} = 0.122$$

La valeur 0.122 montre que les deux films ne sont pas très similaires selon ces trois critères.

Maintenant essayons pour « toy story » est « Bug's life » :

$$cos\theta = \frac{(0.563 \times 0.435) + (0.014 \times 0.016) + (0.13 \times 0.032)}{\sqrt{(0.563)^2 + (0.014)^2 + (0.13)^2}.\sqrt{(0.435)^2 + (0.016)^2 + (0.032)^2}}$$
$$= \frac{0.25}{0.58 \times 0.44} = 0.988$$

La valeur est bien élevée, synonyme d'une similarité élevée entre les deux films.

Similarité

Ecrivez la méthode cosinus_sim qui calcule la similarité entre deux films à base de l'ensemble de 1128 tags de la base de données. Normalement, le tableau tag_relevance contient 1128 valeurs par film, mais il faut aussi prévoir le cas où un film en possède moins de valeurs voire zéro valeurs.

Dans le cas où il manque des valeurs, la méthode s'adapte à la taille de la liste d'intersection entre les deux listes (les valeurs en commun).

Dans le cas où aucune valeur n'existe en commun entre les deux films, la similarité égale à zéro.

Algorithme pour trouver un film

Dans cette partie, on va développer un algorithme qui aide l'utilisateur à trouver un film qu'il aimerait regarder depuis une liste de films :

1. Si la taille de la liste égale à zéro on imprime un message pour expliquer qu'on ne peut pas proposer un film
2. Si la taille de la liste égale à 1, on affiche le seul choix à l'utilisateur
3. Si la taille de la liste égale à 2, on affiche les deux films l'un à coté de l'autre exactement comme la figure 6 en bas, mais avec les boutons « je préfère … » désactivés.

4. Si la taille de la liste est supérieure à 2, on choisit un film (A) au hasard, puis calcule sa similarité avec l'ensemble de films dans la liste. On sélectionne (B) le film qui ressemble le moins à (A), et on affiche les deux côte-à-côte.

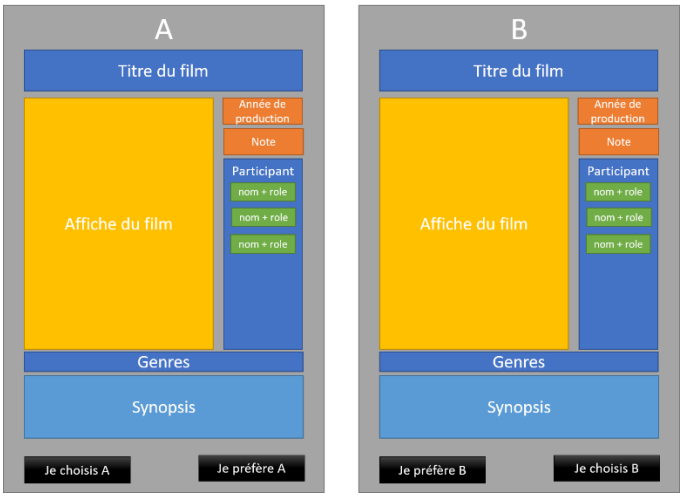


Figure 6 choisir un film

5. L'utilisateur a quatre choix :
    a. Si l'utilisateur appuie sur « je choisis A » ou « je choisis B », c'est qu'il a trouvé ce qu'il cherche et le jeu termine.
    b. Dans le cas ou aucun de deux films ne correspond pas exactement à ce que l'utilisateur cherche, ce dernier est invité à déclarer une préférence pour l'un des deux films, en appuyant sur « je préfère A » ou « je préfère B ».
6. Lorsque l'utilisateur donne une préférence à l'un des deux films, on calcule la similarité entre ce film et le reste de la liste des films, on ordonne la liste par la similarité au film sélectionné. Puis on supprime la queue de la liste (on garde que la moitié qui ressemble le plus au film sélectionné). La nouvelle liste sera notre nouvelle liste d'entrée, on repart à l'étape 1 avec cette nouvelle liste plus courte.

L'idée de l'algorithme est de continuer à couper la liste par 2 à chaque étape en considérant la préférence de l'utilisateur, jusqu'à ce qu'il choisisse un film, ou que la taille de la liste devient inférieure à deux.

Exemple : supposons que la liste de films était composée seulement de premiers 8 films du tableau movie

toy story
jumanji
grumpier old men
waiting to exhale
father of the bride part ii
heat
Sabrina
Tom and huck

Et que l'algorithme a choisi le film « toy story » comme référence. On calcule la similarité entre ce film et les autres 7 films, ce qui donne le tableau suivant :

Film	Similarité avec toy story
jumanji	`0.764`

grumpier old men	0.689
waiting to exhale	0.635
father of the bride part ii	0.653
heat	0.633
Sabrina	0.640
Tom and huck	0.689

Le film qui ressemble moins à « toy story » dans cette liste est « heat », alors on propose à l'utilisateur de choisir entre « toy story » et « heat ».

Supposons maintenant que l'utilisateur a cliqué sur « je préfère toy story ». Dans ce cas, il faut éliminer les 4 films sur 8 qui ressemble moins au choix de l'utilisateur (heat, waiting to exhale, sabrina, father of the bride part ii).

La nouvelle liste contient 4 films : (toy story, jumanji, grumpier old men, tom and huck).

L'étape suivante de l'algorithme serait de choisir un film au hasard parmi ces 4 films, trouver son opposé, et ainsi de suite.