Université d'Ottawa
Faculté de génie

École de science
d'informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

CSI2520 Programming Paradigms
CSI2520 Paradigmes de Programmation

Movie Recommendation System
Comprehensive assignment
(32%)

Winter 2025

Individual project

Part 1 due February 10th before 23:59

Part 2 due March 10th before 23:59

Part 3 due March 31st before 23:59

Part 4 due le April 7th before 23:59

Late assignment policy: minus 10% for each day late. For example: a project due Friday night but handed out on the Monday morning: -30%

**** This project will count towards your final mark if and only if you obtain at least 50% on your exams (midterm + final)***

Système de recommandation de Films
Projet intégrateur
(32%)

Hiver 2025

Ce projet est à réaliser de façon individuelle

Partie 1 due le 10 Février avant 23:59

Partie 2 due le 10 Mars avant 23:59

Partie 3 due le 31 Mars avant 23:59

Partie 4 due le 7 Avril avant 23:59

Pénalité de retard : perte de 10% de la note obtenue par jour de retard. Exemple : pour un projet dû le vendredi soir mais remis le lundi matin : -30%

**** Ce projet comptera dans votre note finale si et seulement si vous obtenez au moins 50% au cumul de vos examens (mi-session et final)***

Problem description

A good item recommendation engine is an important tool in music/movie streaming apps, online shopping websites, etc. The basic principle is quite simple: looking at your past streaming/shopping history, the objective is to find other users sharing similar transactions. Now if some of these users have liked items you also liked, then it is probably because you have similar preferences/interests than them. Therefore, by looking at their history of transactions, you have to identify items they have liked and that you have never streamed/bought. These are the items that will be recommended to you.

Based on this idea, we ask you to implement a simple algorithm for movie recommendations. This simple algorithm is described below. To test your programs, you will use a database of movie ratings generated by a large number of users.

Comparing users' preferences

In order to identify movies to recommend, the first step is to find users with similar preferences. This similarity between users can be estimated using the Jaccard index. Given two users, U_1 and U_2 , the sets of movies they liked, $L(U_1)$ and $L(U_2)$, and the sets of movies they did not like, $D(U_1)$ and $D(U_2)$, the similarity between these users can be evaluated as:

$$S(U1,U2) = |L(U_1) \cap L(U_2)| + |D(U_1) \cap D(U_2)| \,/\, |L(U_1) \cup L(U_2) \cup D(U_1) \cup D(U_2)|$$

which can be read as the number of common liked/disliked movies over the number of distinct movies viewed by the two users (intersection over union). This is a number between 0 and 1. Consider now a movie M that user U has not viewed, the probability that this user will like this movie can be defined as:

$$\text{score}(U,M) = \sum_{V \in L(M)} S(U,V)$$
$$P(U,M) = \text{score}(U,M) \,/\, |L(M)|$$

with $L(M)$ being the set of users who liked movie M. Basically, this equation expresses the fact that if all the users who liked movie M have preferences similar to user U then the probability that this user will also like that movie is very high. Note that, for this probability to be reliable, a movie must has been liked by a minimum number K of users.

From these equations, a brute-force approach to produce a list of N recommendations for a user would consist in finding the N unseen movies with the highest probability of being liked by this user. Obviously, several improvements could be applied to this method, but, in this assignment, this is the formula we will use.

The MovieLens dataset

To test your results, you will use the `ml-latest-small` dataset that contains 100,000 ratings applied to 9,000 movies by 600 users. This dataset can be found here:

https://grouplens.org/datasets/movielens/

The zip file you will find there contains two CSV (comma-separated values) files that are of interest to us. The first one is the list of movies. Each movie has an ID number, a title and a list of genres. For example:

```
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
```

The second file contains the ratings. Each line contains the ID of the user who made the rating, the ID of the rated movie, the rating and a timestamp (this later is not useful to us). For example:

```
1,1,4.0,964982703
1,3,4.0,964981247
1,6,4.0,964982224
2,318,3.0,1445714835
2,333,4.0,1445715029
```

The rating is a number between 1 and 5. In our case, we will consider that a user has liked a movie if the rating specified is over or equal to R.

Generating recommendations

You are asked to recommend N movies to a given user. To do so, you will consider all movies and all users in the dataset, on which you will apply the formula given above. Once you get the probability of being liked for all movies, you will sort the recommendations and retain only the first N.

There are different algorithms that can be used to accomplish this task with a complexity that depends on the number of users and movies and on the memory constraints. We will adopt here a simple version, that is not the most efficient but that should be the easiest to implement.

The algorithm

```
The input is the dataset of movies and ratings
and the ID of user U.

for each movie M in the dataset:
  if the movie M not been viewed by U:
     if the movie M has been liked by at least K users:
          score(U,M)=0
          |L(M)|= 0

          for each user V, V≠U:
               if V liked movie M:
                    compute S(U,V)
                    score(U,M)+= S(U,V)
                    |L(M)|++

          Create recommendation with P(U,M)= score(U,M)/|L(M)|
```

You must use this exact algorithm in your project.

Recommended values

For your experiments, use the following values:

- K=10
- R= 3.5
- N= 20

Programming

You have to write programs under different paradigms that solve different this problem. You will receive specific instructions for each language.

Each program will be marked as follows:

Program produces the correct result	[3 points]
Adherence to programming paradigm	[2 points]
Quality of programming (structures, organisation, etc)	[1.5 point]
Quality of documentation (comments and documents)	[1.5 point]

You must submit all your source files together with the outputs of your program (in a text file). All your files must include a header showing student name and number. These files must be submitted in a zip file.

Problème à résoudre

Un bon système de recommandation d'items est un outil important dans les applications de streaming de musique/films, dans les sites de commerce en ligne, etc. Le principe de base est assez simple : en analysant votre historique d'achats passé, l'objectif est de trouver d'autres utilisateurs ayant effectué des transactions similaires. Si ces utilisateurs ont aimé les mêmes articles que vous, c'est probablement parce que vous avez des préférences/intérêts similaires aux leurs. Par conséquent, en examinant leur historique de transactions, vous êtes en mesure d'identifier les items qu'ils ont aimés et que vous n'avez jamais acheté ou écouté. Ce sont ces items qui devraient vous être recommandés.

En vous basant sur cette idée, nous vous demandons de mettre en œuvre un algorithme simple de recommandation de films. Cet algorithme est décrit ci-dessous. Pour tester vos programmes, vous utiliserez une base de données d'évaluations de films générée par des utilisateurs.

Comparer les préférences des utilisateurs :

Afin d'identifier les films à recommander, la première étape consiste à trouver des utilisateurs ayant des préférences similaires. Cette similarité entre utilisateurs peut être estimée en utilisant l'index de Jaccard. Soient deux utilisateurs, U1 et U2, les ensembles de films qu'ils ont aimés, L(U1) et L(U2), et les ensembles de films qu'ils n'ont pas aimé, D(U1) et D(U2), la similarité entre ces utilisateurs peut être évaluée comme suit :

$$S(U1,U2) = |L(U_1) \cap L(U_2)| + |D(U_1) \cap D(U_2)| \,/\, |L(U_1) \cup L(U_2) \cup D(U_1) \cup D(U_2)|$$

Cette formule peut être interprétée comme le nombre de films aimés/non-aimés en commun divisé par le nombre de films distincts vus par les deux utilisateurs (intersection sur union). Cette valeur est un nombre compris entre 0 et 1. Considérons maintenant un film M que l'utilisateur U n'a pas vu, la probabilité que cet utilisateur aime ce film peut être définie comme suit :

$$\text{score}(U,M) = \sum_{V \in L(M)} S(U,V)$$
$$P(U,M) = \text{score}(U,M) \,/\, |L(M)|$$

avec L(M) étant l'ensemble des utilisateurs ayant aimé le film M. Fondamentalement, cette équation exprime le fait que si tous les utilisateurs ayant aimé le film M ont des préférences similaires à celles de l'utilisateur U, alors la probabilité que cet utilisateur aime également ce film est très élevée. Notez que pour que cette probabilité soit fiable, un film doit avoir été aimé par un nombre minimum K d'utilisateurs.

À partir de ces équations, une approche brute pour produire une liste de N recommandations pour un utilisateur consiste à trouver les N films non vus ayant la probabilité la plus élevée d'être aimés par cet utilisateur. Évidemment, plusieurs améliorations pourraient être apportées à cette méthode, mais, dans ce devoir, nous allons simplement utiliser cette formule.

La base de données MovieLens

Afin de tester vos programmes, vous utiliserez la base de données `ml-latest-small` contenant 100,000 évaluations de 9,000 films effectuées par 600 utilisateurs. Cette base se trouve ici :

https://grouplens.org/datasets/movielens/

Le fichier zip qui s'y trouve contient deux fichier CSV (*comma-separated values*) d'intérêt. Le premier contient la liste des films. Chaque film est spécifié par son identificateur, un titre et une liste de genres. Par exemple:

```
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
```

Le second fichier contient les évaluations. Chaque ligne contient l'identificateur d'un utilisateur ayant effectué une évaluation, l'identificateur associé au film, l'évaluation et un *timestamp* (ce dernier ne sera pas utilisé ici). Par exemple:

```
1,1,4.0,964982703
1,3,4.0,964981247
1,6,4.0,964982224
2,318,3.0,1445714835
2,333,4.0,1445715029
```

L'évalutaion est un nombre entre 1 et 5. Dans notre cas, nous considèrerons qu'un utilisateur a aimé un film si son évaluation est supérieure ou égale à R.

Génération de recommendations

On vous demande de recommander N films à un utilisateur donné. Pour ce faire, vous devrez prendre en compte tous les films et tous les utilisateurs de la base de données, et appliquer la formule donnée ci-dessus. Une fois que vous avez calculé la probabilité que chaque film soit aimé par l'utilisateur U, vous triez les recommandations et ne retenez que les N premiers films.

Il existe différents algorithmes qui peuvent être utilisés pour accomplir cette tâche, dont la complexité varie selon le nombre d'utilisateurs et de films, ainsi que les contraintes de mémoire. Nous adopterons ici une version simple, qui n'est pas la plus efficace, mais qui devrait être la plus facile à mettre en œuvre.

The algorithm

```
The input is the dataset of movies and ratings
and the ID of user U.

for each movie M in the dataset:
  if the movie M not been viewed by U:
     if the movie M has been liked by at least K users:
          score(U,M)=0
          |L(M)|= 0

          for each user V, V≠U:
               if V liked movie M:
                    compute S(U,V)
                    score(U,M)+= S(U,V)
                    |L(M)|++

          Create recommendation with P(U,M)= score(U,M)/|L(M)|
```

C'est cet exact algorithme que vous devez utiliser dans votre projet.

Valeur recommandées

Pour vos expériences, utiliser les valeurs suivantes:

- K=10
- R= 3.5
- N= 20

Programmation

Vous devez écrire une série de programmes sous les différents paradigmes de programmation afin de résoudre le problème proposé. Vous recevrez des instructions spécifiques pour chacun des langages.

Chaque programme sera corrigé en suivant la grille suivante :

Le programme produit le bon résultat	[3 points]
Adhérence au paradigme de programmation	[2 points]
Qualité de la programmation (structure, organisation, etc)	[1.5 point]
Qualité de la documentation (commentaires, documents)	[1.5 point]

Il faut soumettre tous les fichiers sources ainsi que les sorties affichées par votre programme (dans un fichier texte). Tous vos fichiers doivent inclure un entête incluant votre nom et numéro d'étudiant. Tous ces fichiers doivent être soumis dans un fichier zip.