Dauphine | PSL UNIVERSITÉ PARIS

M1 MSI
INITIATION ALGORITHMIQUE ET PROGRAMMATION
LANGAGE PYTHON
TRAVAUX PRATIQUES
Brice Mayag

TP3 : INITIATION AUX SYSTÈMES DE RECOMMANDATION (FILTRAGE COLLABORATIF)

Le filtrage collaboratif est une technique de personnalisation sur les sites marchands qui permet de faire des recommandations d'achat à un internaute en comparant ses préférences sous forme d'évaluations produits (livres, disques, films, jeux,...) aux jugements portés par des internautes au profil similaire sur des produits qui n'ont pas encore été achetés par le prospect à qui est faite la recommandation.

Le terme "collaboratif" est utilisé car la technique nécessite que l'internaute ait déclaré préalablement ses goûts à travers une évaluation de produits (livres lus, films vus, etc.). Le principe est de dire "vous devriez aimer ce livre ou ce film car les internautes ayant les mêmes goûts que vous l'ont aimé".

Quel film proposé à Anne ?

Les évaluations de sept Critiques cinématographiques, données entre 0 et 5 sur six films, sont reportées dans le tableau ci-dessous. Les films évalués ont pour titre : "Lady in the waters" (Lady), "Snakes on the Plane" (Snakes), "Just My Luck" (Luck), "Superman Returns" (Superman), "You, Me and Dupree" (Dupree) et "The Night Listener" (Night). Une case vide signifie que le critique n'a pas vu le film en question et donc est incapable de l'évaluer.

	Lady	Snakes	Luck	Superman	Dupree	Night
Lisa Rose	2.5	3.5	3.0	3.5	2.5	3.0
Gene Seymour	3.0	3.5	1.5	5.0	3.5	3.0
Michael Phillips	2.5	3.0		3.5		4.0
Claudia Puig		3.5	3.0	4.0	2.5	4.5
Mick Lasalle	3.0	4.0	2.0	3.0	2.0	3.0
Jack Matthews	3.0	4.0		5.0	3.5	3.0
Toby		4.5		4.0	1.0	

Anne, une étudiante dauphinoise souhaite aller au cinéma et hésite entre les trois films "Snakes", "Superman" et "Night". Ses préférences parmi les films déjà vus sont résumées par le tableau suivant :

	Lady	Snakes	Luck	Superman	Dupree	Night
Anne	1.5		4.0		2.0	

L'objectif de cet exercice est de faire des recommandations à Anne en tenant compte des notes des sept premiers Critiques. Pour cela, on va construire un petit algorithme intelligent pour résoudre ce problème.

1. Construire un dictionnaire `Critiques` contenant les huit Critiques (y compris "Anne") des films et leurs notes.

   `Critiques` sera donc un dictionnaire de dictionnaire. Par exemple, le dictionnaire associé aux évaluations de Lisa Rose sera obtenu en tapant :

   ```
   >>> Critiques['Lisa Rose']
   {'Lady': 2.5, 'Snake':3.5, 'Luck': 3.0, 'Superman': 3.5,
   'Dupree': 2.5, 'Night': 3.0}.
   ```

2. Après avoir réuni les données sur les préférences des personnes, nous avons besoin d'un mécanisme permettant de déterminer celles ayant des goûts similaires à Anne. Pour cela, nous allons comparer chaque personne à toutes les autres en calculant un "score de similarité" ou "score de similitude".

   Pour calculer simplement un score de similarité, nous utiliserons la distance de Manhattan ou la distance euclidienne.

   Ainsi, si n représente le nombre de films pour lesquels les Critiques x et y ont attribué une note, alors le score de similarité entre x et y sera assimilé à :

   - leur distance de Manhattan $d(x, y)$ définie par la formule

   $$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

   - leur distance euclidienne $d(x, y)$ définie par la formule

   $$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

   où $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ sont les vecteurs notes des n films pour lesquels x et y ont attribué une note (les films pour lesquels aucune note n'a été attribuée ne seront pas pris en compte dans cette formule).

   (a) Analysez le code suivant qui retourne un sore de similarité, basé sur la distance de Manhattan, entre deux critiques :

   ```
   # Distance de similarite de Manhattan entre deux utilisateurs. Les utilisateurs
   # d'evaluations

   def sim_manhattan(person1, person2):
     distance = 0
     for key in person1:
       if key in person2:
          distance += abs(person1[key] - person2[key])
     return distance
   ```

   Exemple : La mesure de similarité, basée sur la distance de Manhattan, entre "Lisa Rose" et "Gene Seymour" sera :

   ```
   >>> sim_distanceManhattan(Critiques['Lisa Rose'],
   Critiques['Gene Seymour'])
   4.5
   ```

(b) Construire la fonction `sim_distanceEuclidienne` qui retourne un score de similarité, basées sur la distance euclidienne, pour deux personnes données.

3. Pour chacune des distances ci-dessus, construire la fonction `recommend(nouveauCritique, Critiques)` qui retourne la liste des films à recommander à l'utilisateur `nouveauCritique` en fonction des goûts des autres Critiques. Ne pas oublier que `Critiques` est un dictionnaire de dictionnaires.

Indication : On pourra utiliser la fonction suivante, basée ici sur la distance de Manhattan, `computeNearestNeighbor(nouveauCritique, Critiques)`, qui retourne une liste triée des Critiques proches de nouveauCritique.

```
def computeNearestNeighbor(nouveauCritique, Critiques):
  distances=[]
  for critique in Critiques:
    if critique!=nouveauCritique:
      distance=sim_manhattan(Critiques[critique],Critiques[nouveauCritique])
      distances.append((distance,critique))
  distances.sort()
  return distances
```

En testant cette fonction, vous obtiendrez par exemple la liste des personnes proches de "Lisa Rose", en terme de distance de Manhattan :

```
>>> computeNearestNeighbor('Lisa Rose', Critiques)
[(1.5, 'Michael Phillips'), (2.0, 'Claudia Puig'), (2.5, 'Anne'),
(3.0, 'Mick LaSalle'), (3.0, 'Toby'), (3.5, 'Jack Matthews'),
(4.5, 'Gene Seymour')]
```

Ainsi, on voit que "Michael Phillips" est la personne la plus proche de "Lisa Rose" avec une similarité de 1.5, et "Gene Seymour" est la personne la plus éloignée avec une similarité de 4.5. Par conséquent, la fonction `recommend`, recommandera à "Lisa Rose" les films qu'elle n'a pas vus, mais que "Michael Phillips" a aimés.

```
>>> recommend('Lisa Rose', Critiques)
[]
```

On obtient une liste vide car "Lisa Rose" a en fait vu tous les films.

En répétant le même procédé pour "Toby", on obtient les résultats suivants :

```
>>> computeNearestNeighbor('Toby', Critiques)
[(1.0, 'Anne'), (2.0, 'Michael Phillips'), (2.5, 'Claudia Puig'),
(2.5, 'Mick LaSalle'), (3.0, 'Lisa Rose'), (4.0, 'Jack Matthews'),
(4.5, 'Gene Seymour')]
>>> recommend('Toby', Critiques)
[('Lady', 1.5), ('Luck', 4.0)]
```

On recommande donc à "Toby" les films `('Lady', 1.5)` et `('Luck', 4.0)` avec pour chacun de ces films, la note attribuée par "Anne", la personne la plus proche de "Toby" en termes de similarité basée sur la distance de Manhattan.