Dauphine | PSL
UNIVERSITÉ PARIS

M1 MSI
INITIATION ALGORITHMIQUE ET PROGRAMMATION
LANGAGE PYTHON
TRAVAUX PRATIQUES

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 dictionnaires. 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) Construire la fonction `sim_manhattan` qui retourne un score de similarité, basées sur la distance de Manhattan, pour deux personnes données.

   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 `recommendProche(nouveauCritique, Critiques)` qui retourne la liste ordonnée des films à recommander à l'utilisateur `nouveauCritique`, liste déterminée uniquement en fonction des goûts de l'utilisateur proche de `nouveauCritique`. Ne pas oublier que `Critiques` est un dictionnaire de dictionnaires qui permettra de calculer, à l'intérieur de la fonction `recommendProche`, les distances de chaque utilisateur par rapport à `nouveauCritique`.

   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 `recommendProch` recommandera à “Lisa Rose” les films qu’elle n’a pas vus, mais que “Michael Phillips” a aimés.

```
>>> recommendProche('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')]
>>> recommendProche('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.

4. La procédure de recommandation précédente permet de consulter les critiques de la personne ayant des goûts similaires à ceux de Anne et de sélectionner un film qu’elle n’a pas encore vu à partir de ces goûts, mais cette approche n’est pas très juste. En effet, elle pourrait mener à des critiques qui n’ont pas donné d’avis sur les films que Anne souhaiterait voir. Elle pourrait également désigner quelqu’un qui a aimé un film ayant eu mauvaise presse chez les autres personnes.

   Pour résoudre ce problème, nous devons évaluer les éléments en produisant un score global qui classe les critiques. La procédure qui déterminera les recommandations à faire à Anne se déroulera en deux étapes. Pour la décrire, notons pour un film a donné, $\mathcal{C}(a)$ la liste des critiques ayant attribué une note à a, et $x(a)$ la note attribuée à a par le critique x.

   (a) **Etape 1** : Pour chaque film a non vu par Anne, calculer les quantités

$$total(a) = \sum_{x \in \mathcal{C}(a)} \frac{1}{1 + d(x, Anne)} \times x(a)$$

$$s(a) = \sum_{x \in \mathcal{C}(a)} \frac{1}{1 + d(x, Anne)}$$

$$s'(a) = \frac{total(a)}{s(a)}$$

La quantité $s'(a)$ permettra de conclure qu'une personne similaire à Anne contribuera de manière plus importante au score global qu'une personne qui lui est différente.

En utilisant la distance de Manhattan, on obtient pour le film "Night" non vu par Anne :

- $\mathcal{C}(a) = \{$Lisa Rose, Gene Seymour, Michael Phillips, Claudia Puig, Mick LaSalle, Jack Matthews$\}$
- $total(a) = \frac{4}{1+1} + \frac{4.5}{1+1.5} + \frac{3}{1+2.5} + \frac{3}{1+3} + \frac{3}{1+3.5} + \frac{3}{1+5.5} = 6.53534799$
- $s(a) = 1.81178266$
- $s'(a) = 3.6071$

(b) **Etape 2** : Le film à recommander à Anne sera le film qui aura la plus grande somme $s'(a)$.

A partir de ces explications, écrire la fonction `Bestrecommend` qui permettra de proposer à Anne une recommandation entre les trois films "Snakes", "Superman" et "Night".

5. Le tableau ci-après représente 8 critiques et 8 films qu'ils ont notés. Les notes vont de 0 à 5.

	Angelica	Bill	Chan	Dan	Hailey	Jordyn	Sam	Veronica
Blues Traveler	3.5	2	5	3	-	-	5	3
Broken Bells	2	3.5	1	4	4	4.5	2	-
Deadmau5	-	4	1	4.5	1	4	-	-
Norah Jones	4.5	-	3	-	4	5	3	5
Phoenix	5	2	5	3	-	5	5	4
Slightly Stoopid	1.5	3.5	1	4.5	-	4.5	4	2.5
The Strokes	2.5	-	-	4	4	4	5	3
Vampire Weekend	2	3	-	2	1	4	-	-

A l'aide des fonctions de recommandation développées, et pour chacune des mesures de similarité (distance de Manhattan, distance euclidienne), proposez à chaque critiques un classement des films qu'il n'a pas encore vus.