COLLABORATIVE FILTERING AS A MODEL OF GROUP DECISION-MAKING

You know that the low-tech way to get recommendations for products, movies, or entertaining web sites is to ask your friends. You also know that some of your friends have better "taste" than others, something you've learned over time by observing whether they usually like the same things as you. As more and more options become available, it becomes less practical to decide what you want by asking a small group of people, since they may not be aware of all the options. This is why a set of techniques called *collaborative filtering* was developed [1].

A collaborative filtering algorithm usually works by searching a large group of people and finding a smaller set with tastes similar to yours. It looks at other things they like and combines them to create a ranked list of suggestions. There are several different ways of deciding which people are similar and combining their choices to make a list ; this tutorial cover a few of these.

Which movie we can suggest to Anne ?

The ratings of seven movie critics, given between 0 and 5 on six movies, are given in the table below. The titles of these movies are "Lady in the waters" (Lady), "Snakes on the Plane" (Snakes), "Just My Luck" (Luck), "Superman Returns" (Superman), "You, Me and Dupree" (Dupree) and"The Night Listener" (Night). An empty cell means that the critic does not seen the movie and is not able to evaluate it.

	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, a student of Centrale Supelec, needs your help to choose between three movies "Snakes", "Superman" and "Night", the movie she could see based on your recommendations. She gave the following ratings to the other movies she already saw.

	Lady	Snakes	Luck	Superman	Dupree	Night
Anne	1.5		4.0		2.0	

Our objective is to provide recommendations to Anne, taking into account the ratings of the seven movie critics. Python language is only used here as an example in the implementation of this model, but you can choose another language (Java, C++, . . .).

1. The term collaborative filtering was first used by David Goldberg at Xerox PARC in 1992 in a paper called "Using collaborative filtering to weave an information tapestry." He designed a system called Tapestry that allowed people to annotate documents as either interesting or uninteresting and used this information to filter documents for other people. There are now more than hundreds of web sites that employ some sort of collaborative filtering algorithm for movies, music, books, dating, shopping, other web sites, podcasts, articles, and even jokes.

1. Build a dictionary `critiques` containing the seven movie critics and their ratings (the dictionary is only a suggestion for the representation of the "preferences" of movie critics. You can choose another data structure).

2. After collecting data about the things people like, you need a way to determine how similar people are in their tastes. You do this by comparing each person with every other person and calculating a *similarity score*. In our case, we should determine which persons are "similar" to Anne.

   (a) To compute a simply similarity score, we use the Manhattan distance or an Euclidean distance.

   Hence, if n represent the number of movies both rated by the critics x and y, then the similarity score between x and y is given by :

   - their Manhattan distance $d(x, y)$ defined by

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

   - their Euclidean distance $d(x, y)$ defined by

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

   where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are the ratings vectors of the n movies evaluated by x and y (the movies non evaluated by x or y are not taken into account in these formulas).

   We can generalize Manhattan Distance and Euclidean Distance to what is called the Minkowski Distance Metric [2].

   i. Build functions `sim_distanceManhattan` and `sim_distanceEuclidienne` which return a similarity score, based respectively on Manhattan distance and Euclidean distance, for given two persons.

   Indication : For instance, we can build a function
   `def sim_distanceEuclidienne(person1,person2)`
   where `person1` et `person2` are dictionaries containing ratings given by the users (critics)

   Example : the dictionary associated to the evaluations of Lisa Rose is obtained by :

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

   Hence we have :

   ```
   >>> sim_distanceEuclidienne(critiques['Lisa Rose'],
   critiques['Gene Seymour'])
   2.3979157616563596
   ```

   ii. For each distance above, build the function
   `recommend(nouveauCritique, Critiques)` returning a list of movies to recommend to the user nouveauCritique based on the tastes of the other critics.

   Indication : For instance, we can use the following function, based on Manhattan distance,
   `computeNearestNeighbor(nouveauCritique, critiques)`, returning a sorted list of critics close to nouveauCritique.

2. $d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{\frac{1}{r}}$. When $r = 1$, the formula is Manhattan Distance and when $r = 2$, the formula is Euclidean Distance

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

By testing these functions, we should have :

```
>>> 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')]
 >>> recommend('Lisa Rose', Critiques)
 []
 >>> recommend('Toby', Critiques)
 [('Lady', 1.5), ('Luck', 4.0)]
```

iii. Finding a good critic to read is great, but what Anne really wants is a movie recommendation right now. Anne could just look at the person who has tastes most similar to her and look for a movie he likes that Anne has not seen yet, but that would be too permissive. Such an approach could accidentally turn up reviewers who haven't reviewed some of the movies that Anne might like. It could also return a reviewer who strangely liked a movie that got bad reviews from all the other critics returned by topMatches.

To solve these issues, you need to score the items by producing a weighted score (global score) that ranks the critics. The procedure which determine the recommendations to suggest to Anne is describe in two steps below. For a given movie a, let us denote by $\mathcal{C}(a)$ the list of all the critics which gave a rating to a, and $x(a)$ the rating given to a by the critic x.

A. **Step 1** : For each movie a not seen by Anne, compute the quantities

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

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

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

The quantity $s'(a)$ translates the fact that a person similar to Anne will more contribute, to the global score, than a person which is different to her.

By using a Manhattan distance, we have for the movie "Night" non seen by 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. **Step 2** : The movie to recommend to Anne will be the movie with the highest global score $s'(a)$.

From these explanations, build the function `Bestrecommend` suggesting to Anne a recommendation between the three movies "Snakes", "Superman" and "Night".

iv. A slightly more sophisticated way to determine the similarity between people's interests is to use a *Pearson correlation coefficient*. The correlation coefficient is a measure of how well two sets of data fit on a straight line. The formula for this is more complicated than the Euclidean distance score, but it tends to give better results in situations where the data is not well normalized for example, if critics' movie rankings are routinely more harsh than average.

Hence, if n represent the number of movies both rated by the critics x and y, then the similarity score between x and y is given by the Pearson correlation coefficient $p(x, y)$ as follows :

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

where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are the ratings vectors of the n movies evaluated by x and y (the movies non evaluated by x or y are not taken into account in this formula).

This function will return a value between -1 and 1 inclusive. A value of 1 means that the two people have exactly the same ratings for every item. -1 indicates perfect disagreement. It is an increasing function, i.e. unlike with the distance metric, you don't need to change this value to get it to the right scale in the computation of $s'(a)$.

Build the function `PearsonRecommend` suggesting to Anne, by using the Pearson correlation coefficient, a recommendation between the three movies "Snakes", "Superman" and "Night".

Indication : One may use the following function determining the Pearson correlation coefficient between two users.

```
def pearson(person1, person2):
    sum_xy=0
    sum_x=0
    sum_y=0
    sum_x2=0
    sum_y2=0
    n=0
    for key in person1:
        if key in person2:
            n += 1
            x=person1[key]
            y=person2[key]
            sum_xy +=x*y
            sum_x += x
            sum_y += y
            sum_x2 += x**2
             sum_y2 += y**2
    denominator = sqrt(sum_x2 - (sum_x**2) / n) *
                  sqrt(sum_y2 - (sum_y**2) / n)
    if denominator == 0:
        return 0
    else:
        return (sum_xy - (sum_x * sum_y) /n ) / denominator
```

v. By using as a similarity score the Cosine similarity, suggesting to Anne, by using the Pearson correlation coefficient, a recommendation between the three movies "Snakes", "Superman" and "Night". We recall the following formula of

Cosine between two users x and y :

$$cos(x,y) = \frac{\sum_{i=1}^{n} x_i \, y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \times \sqrt{\sum_{i=1}^{n} y_i^2}}$$

where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are the ratings vectors of the n movies evaluated by x and y (the movies non evaluated by x or y are not taken into account in this formula).

The cosine similarity rating ranges from 1 indicated perfect similarity to -1 indicate perfect negative similarity. Note that this function, which is very popular in text mining, is an increasing function.

Which similarity measure to use ?

- If the data is subject to grade-inflation (different users may be using different scales) use Pearson.
- If your data is dense (almost all attributes have nonzero values) and the magnitude of the attribute values is important, use distance measures such as Euclidean or Manhattan.
- If the data is sparse consider using Cosine similarity.