German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher

Concepts of Programming Languages, Spring term 2018
Project Description
"Recommender System"
Deadline: 30.04.2017 - 11:59 pm

The project will put your knowledge in Haskell to the test. Before proceeding, make sure that you read each section carefully. Enjoy.

Project Description

Collaborative Filtering is one of the techniques used in recommender systems. The aim of the project is to provide an implementation of a recommender system based on collaborative filtering.

The technique is based on the user-item ranking. The similarity between objects is used to predict missing user-item rankings. In more details, this is how the technique work:

a) Users give ratings for some of the items.

b) Similarities between items are calculated.

c) The algorithm also takes into account the users who rated the same item to calculate similarities.

d) The system then predicts user-item ratings.

Example

For example, if we find that on average people rate item 2 with `1` more than item 3 and they give the same rating for both items 1 and 2. In case we know that a user rated item 1 with `4` and rated item 3 with `2`. In case the user did not rate item 2, it could be estimated as follows:

- According to item 3, it is $2 + 1 = 3$.
- According to item 1, it is 4

The rating is calculated as the average between 3 and 4 (3.5).

Bigger Example

	Item 1	Item 2	Item 3
User A	3	5	?
User B	4	3	3

In the example shown above, user A rated item 1 with and item 2 with 5 but did not rate item 3. Whereas, user B rated items 1, 2 and 3 with the values 4, 3 and 3 respectively. In case we need to predict the rating user A would give to item 3 the follows should be done:

a) We will be relying on user B since according to the data we have, this is the only user who rated item 3.

b) We will try o find a pattern between the way user B rated item 3 and relate it to the rating user B gave to the other items

c) The differences between the rating given to item 3 and the ratings given to the rest of the items is calculated as follows:

   1. User B rated item 1 with 4 and item 3 with 3. Thus the difference is $3 - 4 = -1$.
   2. User B rated item 2 with 3 and item 3 with 3. Thus the difference is $3 - 3 = 0$.

d) The calculated differences will be used to project a possible estimate for the rating that user A might give to item 3 similar to the previous example:

   1. If we use the difference calculated for item 1, the rating would be $3 - 1 = 2$
   2. If we use the difference calculated for item 2, the rating would be $5 + 0 = 5$.
   3. We thus use the avergae: $\frac{5+2}{2} = 3.5$.

Functions

Te rest of the description will show the functions that should be implemented.

dis

The function `dis`:
`dis :: Eq a => [a] -> [a]` takes a list of values and returns a list of distinct values.
Examples:

```
> dis [1,3,2,3,4,1]
[2,3,4,1]

> dis [I "Lenovo", I "Lenovo", I "Sofa"]
[I "Lenovo",I "Sofa"]
```

fromRatingsToItems

The function `fromRatingsToItems`:
`fromRatingsToItems :: Eq a => [(b,a,c)] -> [a]` takes a list of ratings of users to items and returns the list of items.
Examples:

```
> fromRatingsToItems [(U "Nada" , I "Laptop Lenovo", 5), (U "Ahmed" , I "Laptop Lenovo", 3),
(U "Carole" , I "Honda", 5), (U "Nada" , I "Honda", 4),(U "Ahmed" , I "Honda", 1)]

[I "Laptop Lenovo",I "Honda"]
```

fromRatingsToUsers

The function `fromRatingsToItems`:
`fromRatingsToUsers :: Eq a => [(a,b,c)] -> [a]` takes a list of ratings of users to items and returns

the list of existing users.
Examples:

```
> fromRatingsToUsers [(U "Nada" , I "Laptop Lenovo", 5), (U "Ahmed" , I "Laptop Lenovo", 3),
(U "Carole" , I "Honda", 5), (U "Nada" , I "Honda", 4),(U "Ahmed" , I "Honda", 1)]

[U "Carole",U "Nada",U "Ahmed"]
```

hasRating

The function `hasRating`:
`hasRating :: (Eq a, Eq b) => a -> b -> [(a,b,c)] -> Bool` takes a user, item and a list of ratings. It returns True if the user rated this item and False otherwise.
Examples:

```
> hasRating (U "Ahmed") (I "Sofa") [(U "Nada" , I "Laptop Lenovo", 5),
(U "Ahmed" , I "Laptop Lenovo", 3), (U "Carole" , I "Honda", 5), (U "Nada" , I "Honda", 4),
(U "Ahmed" , I "Honda", 1)]

False

> hasRating (U "Ahmed") (I "Honda") [(U "Nada" , I "Laptop Lenovo", 5),
(U "Ahmed" , I "Laptop Lenovo", 3), (U "Carole" , I "Honda", 5), (U "Nada" , I "Honda", 4),
(U "Ahmed" , I "Honda", 1)]


True
```

getRating

The function `hasRating`:
`getRating :: (Eq a, Eq b) => a -> b -> [(a,b,c)] -> c` takes a user, item and a list of ratings. It returns the rating given by this user to this item.
Examples:

```
> getRating(U "Ahmed") (I "Honda") [(U "Nada" , I "Laptop Lenovo", 5),
(U "Ahmed" , I "Laptop Lenovo", 3), (U "Carole" , I "Honda", 5), (U "Nada" , I "Honda", 4),
(U "Ahmed" , I "Honda", 1)]

1

> getRating(U "Ahmed") (I "Sofa") [(U "Nada" , I "Laptop Lenovo", 5),
(U "Ahmed" , I "Laptop Lenovo", 3), (U "Carole" , I "Honda", 5), (U "Nada" , I "Honda", 4),
(U "Ahmed" , I "Honda", 1)]

Program error: No given rating
```

formMatrixUser

The function `formMatrixUser`:
`formMatrixUser :: (Eq a, Eq b, Fractional c) => b -> [a] -> [(b,a,c)] -> [Rating c]` takes a user, list of available items and a list of ratings. It returns a list with the rating given by this user to each item. The rating is given through the type `Rating` as shown in the below example.
Examples:

```
> formMatrixUser (U "Carole")[I "Laptop Lenovo",I "Honda"]
[(U "Nada" , I "Laptop Lenovo", 5),
(U "Ahmed" , I "Laptop Lenovo", 3), (U "Carole" , I "Honda", 5), (U "Nada" , I "Honda", 4),
(U "Ahmed" , I "Honda", 1),(U "Carole", I "Laptop Lenovo" , 4)]

[R 4.0,R 5.0]


> formMatrixUser (U "Carole")[I "Laptop Lenovo",I "Honda"]
[(U "Nada" , I "Laptop Lenovo", 5), (U "Ahmed" , I "Laptop Lenovo", 3),
(U "Nada" , I "Honda", 4),(U "Ahmed" , I "Honda", 1),
(U "Carole", I "Laptop Lenovo" , 4)]

[R 4.0,NoRating]
```

formMatrix

The function `formMatrix`:
`formMatrix :: (Eq a, Eq b, Fractional c) => [b] -> [a] -> [(b,a,c)] -> [[Rating c]]` takes the list of available users, the list of available items and a list of ratings. It returns a matrix of the rating given by each user to each item using the typr `Rating`.
Examples:

```
>formMatrix [U "Carole",U "Nada",U "Ahmed"] [I "Laptop Lenovo",I "Honda"]
[(U "Nada" , I "Laptop Lenovo", 5), (U "Ahmed" , I "Laptop Lenovo", 3),
(U "Nada" , I "Honda", 4),(U "Ahmed" , I "Honda", 1),(U "Carole", I "Laptop Lenovo" , 4)]

[[R 4.0,NoRating],[R 5.0,R 4.0],[R 3.0,R 1.0]]


> formMatrix [U "Karim",U "Carole",U "Nada",U "Ahmed"] [I "Laptop Lenovo",I "Honda"]
[(U "Nada" , I "Laptop Lenovo", 5), (U "Ahmed" , I "Laptop Lenovo", 3),
(U "Nada" , I "Honda", 4),(U "Ahmed" , I "Honda", 1),(U "Carole", I "Laptop Lenovo" , 4)]

[[NoRating,NoRating],[R 4.0,NoRating],[R 5.0,R 4.0],[R 3.0,R 1.0]]
```

numberRatingsGivenItem

The function `numberRatingsGivenItem`:
`numberRatingsGivenItem :: (Fractional a, Num b) => Int -> [[Rating a]] -> b` takes the index of an item and the matrix of ratings and returns the number of ratings given to this item.
Examples:

```
> numberRatingsGivenItem 0 [[NoRating,R 5.0],[R 5.0,R 4.0],[R 3.0,R 1.0]]
2

> numberRatingsGivenItem 1 [[NoRating,R 5.0],[R 5.0,R 4.0],[R 3.0,R 1.0]]
3
```

differeneRatings

The function `differeneRatings :: Fractional a => Rating a -> Rating a -> a` takes 2 ratings and returns the difference between them.

Examples:

```
> differeneRatings NoRating (R 5)
0.0

> differeneRatings (R 5) NoRating
0.0

> differeneRatings (R 5) (R 3)
2.0

> differeneRatings (R 3) (R 7)
-4.0
```

matrixPairs

The function `matrixPairs :: Num a => a -> [(a,a)]` takes a number and constructs a lis of pairs os all the numbers between 0 and the input number -1.
Examples:

```
>matrixPairs 3
[   (0,0),(0,1),(0,2),
    (1,0),(1,1),(1,2),
    (2,0),(2,1),(2,2)]


> matrixPairs 6
[   (0,0),(0,1),(0,2),(0,3),(0,4),(0,5),
    (1,0),(1,1),(1,2),(1,3),(1,4),(1,5),
    (2,0),(2,1),(2,2),(2,3),(2,4),(2,5),
    (3,0),(3,1),(3,2),(3,3),(3,4),(3,5),
    (4,0),(4,1),(4,2),(4,3),(4,4),(4,5),
    (5,0),(5,1),(5,2),(5,3),(5,4),(5,5)]
```

dMatrix

The function `dMatrix :: Fractional a => [[Rating a]] -> [a]` takes the ratings' matrix and returns a matrix for the items ratings differences. The value at row i and column j represents the sum of the differences between the ratings given to item i and item j by the same user.
Examples:

```
> dMatrix [[NoRating,R 5.0],[R 5.0,R 4.0],[R 3.0,R 1.0]]

[0.0,3.0,-3.0,0.0]
```

The value 3 in the first row was calculated as (5-4) four user 2+(3-1) for user 3

freqMatrix

The function `freqMatrix :: (Num a, Fractional b) => [[Rating b]] -> [a]` takes the ratings' matrix and returns a matrix for the frequency of items' ratings. The value at row i and column j represents number of users who rated both item i and j.
Examples:

```
>freqMatrix [[NoRating,R 5.0],[R 5.0,R 4.0],[R 3.0,R 1.0]]

[2,2,2,3]
```

diffFreqMatrix

The function `diffFreqMatrix :: Fractional a => [[Rating a]] -> [a]` takes the ratings' matrix and returns a matrix for the average differences (using the frequencies).
Examples:

```
> diffFreqMatrix [[NoRating,R 5.0],[R 5.0,R 4.0],[R 3.0,R 1.0]]

[0.0,1.5,-1.5,0.0]
```

predict

The function `predict` is used to predict the rating a user would have given an item. It takes the ratings matrix and the index of the user and the index of the item whose rating is missing. The prediction is used as the average prediction based on all the differences relevant to this item.