

# Week11 - Recommendation Systems

James Quacinella

11/04/2015

## Exercise 9.3.1

Figure 9.8 is a utility matrix, representing the ratings, on a 1–5 star scale, of eight items, a through h, by three users A, B, and C. Compute the following from the data of this matrix.

- Treating the utility matrix as boolean, compute the Jaccard distance between each pair of users.
- Repeat Part (a), but use the cosine distance.
- Treat ratings of 3, 4, and 5 as 1 and 1, 2, and blank as 0. Compute the Jaccard distance between each pair of users.
- Repeat Part (c), but use the cosine distance.
- Normalize the matrix by subtracting from each nonblank entry the average value for its user.
- Using the normalized matrix from Part (e), compute the cosine distance between each pair of users.

## Answer

Prep:

```
require(lsa)
```

```
## Loading required package: lsa
## Loading required package: SnowballC
```

```
# Utility matrix for problem
utility.matrix <- matrix(c(4, 0, 2,
                           5, 3, 0,
                           0, 4, 1,
                           5, 3, 3,
                           1, 1, 0,
                           0, 2, 4,
                           3, 1, 5,
                           2, 0, 3), nrow=3)

# Convert user row into boolean vector
make.boolean <- function(user) {
  return(lapply(user, function(x) {return(as.integer(x != 0));}));
}

# Jaccard distance
dist.jaccard <- function(x, y) {
  return(1 - sum(x & y) / sum(x | y));
}
```

```

# Cosine distance
dist.cosine <- function(x, y) {
  return(1 - cosine(x,y));
}

# For part a), create boolean version of utility matrix
utility.matrix.boolean <- data.frame()
utility.matrix.boolean <- rbind(utility.matrix.boolean, make.boolean(utility.matrix[1,]))
utility.matrix.boolean <- rbind(utility.matrix.boolean, make.boolean(utility.matrix[2,]))
utility.matrix.boolean <- rbind(utility.matrix.boolean, make.boolean(utility.matrix[3,]))
colnames(utility.matrix.boolean) <- c('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h')

# Example of Jaccard distances working (from book)
#dist.jaccard(c(4,0,0,5,1,0,0), c(5,5,4,0,0,0,0)) # 0.8
#dist.jaccard(c(4,0,0,5,1,0,0), c(0,0,0,2,4,5,0)) # 0.5

```

a) Treating the utility matrix as boolean, compute the Jaccard distance between each pair of users

```
dist.jaccard(utility.matrix.boolean[1,], utility.matrix.boolean[2,])
```

```
## [1] 0.5
```

```
dist.jaccard(utility.matrix.boolean[1,], utility.matrix.boolean[3,])
```

```
## [1] 0.5
```

```
dist.jaccard(utility.matrix.boolean[2,], utility.matrix.boolean[3,])
```

```
## [1] 0.5
```

b) Repeat Part (a), but use the cosine distance.

```

# cosine(c(4,0,0,5,1,0,0), c(5,5,4,0,0,0,0)) # 0.37986
# cosine(c(4,0,0,5,1,0,0), c(0,0,0,2,4,5,0)) # 0.322

dist.cosine(as.matrix(utility.matrix)[1,], as.matrix(utility.matrix)[2,]) # 0.6010408

```

```
##           [,1]
## [1,] 0.3989592
```

```
dist.cosine(as.matrix(utility.matrix)[1,], as.matrix(utility.matrix)[3,]) # 0.6149187
```

```
##           [,1]
## [1,] 0.3850813
```

```
dist.cosine(as.matrix(utility.matrix)[2,], as.matrix(utility.matrix)[3,]) # 0.5138701
```

```
##           [,1]
## [1,] 0.4861299
```

- c) Treat ratings of 3, 4, and 5 as 1 and 1, 2, and blank as 0. Compute the Jaccard distance between each pair of users.

```
# Treat ratings of 3, 4, and 5 as 1 and 1, 2, and blank as 0.
make.grouped <- function(user) {
  return(lapply(user, function(x) {
    if(x==0||x==1||x==2) {
      return(0);
    } else {
      return(1);
    }
  }));
}

utility.matrix.grouped <- data.frame()
utility.matrix.grouped <- rbind(utility.matrix.grouped, make.grouped(utility.matrix[1,]))
utility.matrix.grouped <- rbind(utility.matrix.grouped, make.grouped(utility.matrix[2,]))
utility.matrix.grouped <- rbind(utility.matrix.grouped, make.grouped(utility.matrix[3,]))
colnames(utility.matrix.grouped) <- c('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h')

utility.matrix.grouped
```

```
##    a b c d e f g h
## 2  1 1 0 1 0 0 1 0
## 21 0 1 1 1 0 0 0 0
## 3  0 0 0 1 0 1 1 1
```

```
dist.jaccard(utility.matrix.grouped[1,], utility.matrix.grouped[2,]) # 0.6
```

```
## [1] 0.6
```

```
dist.jaccard(utility.matrix.grouped[1,], utility.matrix.grouped[3,]) # 0.6666667
```

```
## [1] 0.6666667
```

```
dist.jaccard(utility.matrix.grouped[2,], utility.matrix.grouped[3,]) # 0.8333333
```

```
## [1] 0.8333333
```

- d) Repeat Part (c), but use the cosine distance.

```
dist.cosine(as.matrix(utility.matrix.grouped)[1,], as.matrix(utility.matrix.grouped)[2,]) # 0.5773503
```

```
##           [,1]
## [1,] 0.4226497
```

```
dist.cosine(as.matrix(utility.matrix.grouped)[1,], as.matrix(utility.matrix.grouped)[3,]) # 0.5
```

```
##      [,1]
## [1,] 0.5
```

```
dist.cosine(as.matrix(utility.matrix.grouped)[2,], as.matrix(utility.matrix.grouped)[3,]) # 0.2886751
```

```
##      [,1]
## [1,] 0.7113249
```

e) Normalize the matrix by subtracting from each nonblank entry the average value for its user.

```
utility.matrix.normalized <- utility.matrix;
utility.matrix.normalized[1,] <- utility.matrix.normalized[1,] - mean(utility.matrix.normalized[1,]);
utility.matrix.normalized[2,] <- utility.matrix.normalized[2,] - mean(utility.matrix.normalized[2,]);
utility.matrix.normalized[3,] <- utility.matrix.normalized[3,] - mean(utility.matrix.normalized[3,]);
utility.matrix.normalized
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,] 1.50 2.50 -2.50 2.50 -1.50 -2.50 0.50 -0.50
## [2,] -1.75 1.25 2.25 1.25 -0.75 0.25 -0.75 -1.75
## [3,] -0.25 -2.25 -1.25 0.75 -2.25 1.75 2.75 0.75
```

f) Using the normalized matrix from Part (e), compute the cosine distance between each pair of users.

```
dist.cosine(as.matrix(utility.matrix.normalized)[1,], as.matrix(utility.matrix.normalized)[2,]) # 1.046374
```

```
##      [,1]
## [1,] 1.046374
```

```
dist.cosine(as.matrix(utility.matrix.normalized)[1,], as.matrix(utility.matrix.normalized)[3,]) # 1.037662
```

```
##      [,1]
## [1,] 1.037662
```

```
dist.cosine(as.matrix(utility.matrix.normalized)[2,], as.matrix(utility.matrix.normalized)[3,]) # 1.288179
```

```
##      [,1]
## [1,] 1.288179
```