Project 1

Charley Ferrari

June 19, 2016

Ruffomendation Engine: A Dog Recommender

This recommendation engine uses collaborative and content-based techniques to recommend dogs to users. I have scraped data from Animal Planet's Dogs101 database to get content on different breeds of dog using BeautifulSoup. I also generated some random ratings for a list of 63 users.

First, I'll set up my data. I'll read in my dogs and ratings matrices. I'll save a version of ratings preserving the NAs, but otherwise will replace the NAs with 0 for hand coding calculations. I'll also make a realRatingMatrix object out of the ratings matrix with NAs.

```
dogs <- read.csv('dogs.csv', row.names=1)
ratings <- read.csv('ratings.csv', row.names=1)

ratings <- t(ratings)
dogs <- as.matrix(dogs)

ratingsUnScaled <- ratings

ratings <- t(scale(t(ratings)))

ratingsna <- ratings

################################################################################
ratings2 <- matrix(ratings, nrow=dim(ratings)[1], ncol=dim(ratings)[2],
                   dimnames = list(users = dimnames(ratings)[[1]],
                                   dogs = dimnames(ratings)[[2]]))

reclabRatings <- as(ratings2, 'realRatingMatrix')
################################################################################

ratings[is.na(ratings)] <- 0
```

Next, I will create a new matrix, mult, that will determine how users rate different qualities in dogs. This is simply done by multiplying the ratings and dogs matrices above, dividing by number of ratings for each user to make it an average.

```
lengthna <- function(x){
  return(length(x[!is.na(x)]))
}

numRatingsInv <- 1/aaply(ratingsna, 1, lengthna)

mult <- sweep(t(ratings %*% dogs), MARGIN=2, numRatingsInv, '*')
```

Then, I will create a user similarity matrix and a dog similarity matrix:

```
sumsquares <- function(x){
  return(sqrt(sum(x^2)))
}

userSim <- aaply(ratings, 1, sumsquares)
#userSim <- aaply(mult, 2, sumsquares)
dogSim <- aaply(dogs, 1, sumsquares)

useruser <- (ratings %*% t(ratings)) / outer(userSim, userSim, '*')
dogsdogs <- (dogs %*% t(dogs)) / outer(dogSim, dogSim, '*')
```

These matrices match what we get when we create similarity matrices in reccomenderlab. As you can see below:

```
useruserRL <- as.matrix(similarity(reclabRatings, method='cosine', which='users'))

kable(useruser[1:8, 1:8])
```

	Alberto	Alex	Arlene	Beryl	Bonnie	Bret	Chris	Cindy
Alberto	1.0000000	0.0085313	-0.0393360	0.0088760	-0.0363809	-0.0198798	0.0868448	0.0252330
Alex	0.0085313	1.0000000	0.1912373	-0.0051840	-0.1413746	0.0530590	0.0409492	-0.0722869
Arlene	-0.0393360	0.1912373	1.0000000	-0.0076730	-0.0812471	0.0810726	0.0386777	-0.0331392
Beryl	0.0088760	-0.0051840	-0.0076730	1.0000000	-0.0362160	0.1038731	0.0331140	0.1485239
Bonnie	-0.0363809	-0.1413746	-0.0812471	-0.0362160	1.0000000	0.0313063	0.0495739	-0.0684008
Bret	-0.0198798	0.0530590	0.0810726	0.1038731	0.0313063	1.0000000	0.0719399	-0.0664211
Chris	0.0868448	0.0409492	0.0386777	0.0331140	0.0495739	0.0719399	1.0000000	-0.0814343
Cindy	0.0252330	-0.0722869	-0.0331392	0.1485239	-0.0684008	-0.0664211	-0.0814343	1.0000000

```
kable(useruserRL[1:8, 1:8])
```

	Alberto	Alex	Arlene	Beryl	Bonnie	Bret	Chris	Cindy
Alberto	0.0000000	0.0085313	-0.0393360	0.0088760	-0.0363809	-0.0198798	0.0868448	0.0252330
Alex	0.0085313	0.0000000	0.1912373	-0.0051840	-0.1413746	0.0530590	0.0409492	-0.0722869
Arlene	-0.0393360	0.1912373	0.0000000	-0.0076730	-0.0812471	0.0810726	0.0386777	-0.0331392
Beryl	0.0088760	-0.0051840	-0.0076730	0.0000000	-0.0362160	0.1038731	0.0331140	0.1485239
Bonnie	-0.0363809	-0.1413746	-0.0812471	-0.0362160	0.0000000	0.0313063	0.0495739	-0.0684008
Bret	-0.0198798	0.0530590	0.0810726	0.1038731	0.0313063	0.0000000	0.0719399	-0.0664211
Chris	0.0868448	0.0409492	0.0386777	0.0331140	0.0495739	0.0719399	0.0000000	-0.0814343
Cindy	0.0252330	-0.0722869	-0.0331392	0.1485239	-0.0684008	-0.0664211	-0.0814343	0.0000000

Now, lets begin using these recommendation engines to make some recommendations. Using a content-based strategy, lets see how Alberto would rate the Afghan:

```
userMeans <- aaply(ratingsUnScaled, 1, mean, na.rm=TRUE)

genrecContent <- function(name, dog){
  if(is.na(ratingsna[name, dog])){
    return((dogs[dog,] %*% mult[,name])/
```

```
            (userSim[name]*dogSim[dog]))
  }
}

predictionContent <- genrecContent('Alberto', 'Afghan Hound') + userMeans['Alberto']

predictionContent
```

```
##          [,1]
## [1,] 3.014108
```

Recommenderlab doesn't have an option to take in content that I could find, so there's no real comparison we can use.

Now lets look at using a user to user collaborative strategy. I'll create a function that takes in the number of nearest neighbors and generates a collaborative based recommendation. Lets try to find out how Alberto would rate an Afghan using this method:

```
genrecCollabUU <- function(name, dog, k){
  namelist <- names(sort(useruserRL[name,], decreasing = TRUE))
  ratinglist <- names(ratingsna[!is.na(ratingsna[,dog]),dog])
  knearest <- ratinglist[order(match(ratinglist,namelist))][1:k]

  Rating <- ((useruserRL[name, knearest] %*% ratings[knearest, dog]) /
    sum(useruserRL[name, knearest])) + userMeans[name]

  return(Rating)
}

handPredictionCollabUU <- genrecCollabUU('Alberto', 'Afghan Hound', 5)

handPredictionCollabUU
```

```
##         [,1]
## [1,] 2.48754
```

Lets look at this similarity with a dog to dog collaborative strategy:

```
dogsdogsRL <- as.matrix(similarity(reclabRatings, method='cosine', which='items'))

dogMeans <- aaply(ratingsUnScaled, 2, mean, na.rm=TRUE)

genrecCollabDD <- function(name, dog, k){
  namelist <- names(sort(dogsdogsRL[dog,], decreasing = TRUE))
  ratinglist <- names(ratingsna[name,!is.na(ratingsna[name,])])
  knearest <- ratinglist[order(match(ratinglist,namelist))][1:k]

  Rating <- ((dogsdogsRL[dog, knearest] %*% ratings[name, knearest]) /
    sum(dogsdogsRL[knearest, dog])) + dogMeans[dog]

  return(Rating)
}
```

```
handPredictionCollabDD <- genrecCollabDD('Alberto', 'Afghan Hound', 5)

handPredictionCollabDD
```

```
##          [,1]
## [1,] 2.469856
```

Next, lets see what we get from the recommenderlab library, using both user based and item based collaboration.

```
recc_model <- Recommender(data = reclabRatings, method = "UBCF")

recom <- predict(recc_model, reclabRatings, type="ratings")

recom <- as(recom, 'matrix')

rlPredictionUBCF <- recom['Alberto', 'Afghan Hound'] + userMeans['Alberto']

rlPredictionUBCF
```

```
##  Alberto
## 2.902953
```

```
recc_model <- Recommender(data = reclabRatings, method = "IBCF")

recom <- predict(recc_model, reclabRatings, type="ratings")

recom <- as(recom, 'matrix')

rlPredictionIBCF <- recom['Alberto', 'Afghan Hound'] + userMeans['Alberto']

rlPredictionIBCF
```

```
##  Alberto
## 2.926942
```

So to summarize what we have:

```
display <- data.frame(types = c('User-User by Hand', 'Dog-Dog by Hand',
                                'User-User in RL', 'Dog-Dog in RL',
                                'Content-Based by hand'),
                      ratings = c(handPredictionCollabUU, handPredictionCollabDD,
                                  rlPredictionUBCF,rlPredictionIBCF, predictionContent))

kable(display)
```

types	ratings
User-User by Hand	2.487540
Dog-Dog by Hand	2.469856
User-User in RL	2.902953
Dog-Dog in RL	2.926942
Content-Based by hand	3.014108

This is all random data, so the conclusions we draw might not be really relevant. It's good to see all the collaborative ratings below the average rating of 3.013. This suggests they're all picking up on the signal that Alberto might not like an Afghan Hound.

I'd be interested to look at this with real data to see what should be happening when comparing content-based methods with collaborative methods. I'm guessing the random way I created this data means that these two approaches shouldn't be expected to match.