Project 4: A Music Recommendation System with Social Context

James Hamski (james.hamski@spsmail.cuny.edu), Vuthy Nguy (vuthy.nguy@spsmail.cuny.edu)

July 10, 2016

1 Introduction

In this project we use the method outlined in the paper Social Network Collaborative Filtering (R. Zheng, D. Wilkinson, and F. Provost, 2008) to add context to a content-based recommender system.

Citation: Zheng, Rong and Wilkinson, Dennis and Provost, Foster, Social Network Collaborative Filtering (October 2008). Stern, IOMS Department, CeDER, Vol. , pp. -, 2008. Available at SSRN: http://ssrn.com/abstract=1303924

2 User-Based Recommendation System (without Social Context)

2.1 Data Preparation

Reading in needed datasets. Artist information will be joined to user-artist dataset for readability

```
user.artists.pairwise <- read_delim('hetrec2011-lastfm-2k/user_artists.dat', delim = "\t")

artists <- read_delim('hetrec2011-lastfm-2k/artists.dat', delim = "\t") %>% select(id, name)
```

```
## Warning: 1 parsing failure.
##  row  col           expected actual
## 1678 name delimiter or quote
```

Converting from Artist IDs to Artist Names for easier interpretation.

```
colnames(artists) <- c("artistID", "name")

#note: one duplicate artist
#artists$name[duplicated(artists$name)==TRUE]
user.artists.pairwise <- inner_join(user.artists.pairwise, artists, by="artistID")

user.artists.pairwise <- select(user.artists.pairwise, -artistID)

user.artists.pairwise$name <- strtrim(user.artists.pairwise$name, 100)

user.artists.pairwise[16060,]$name <- NA

#ggplot(user.artists.pairwise, aes(x = weight)) + geom_density() + theme_bw() + scale_x_log10()
```

The user-artist pairwise record contains multiple rows for a single artist. In order to understand the distribution of listens for artists, we group and sum for each artist.

```
artist.listens <- user.artists.pairwise %>%
  group_by(name) %>%
  summarise(total.listens = sum(weight))
```

```
## Warning: failed to assign NativeSymbolInfo for lhs since lhs is already
## defined in the 'lazyeval' namespace
```

```
## Warning: failed to assign NativeSymbolInfo for rhs since rhs is already
## defined in the 'lazyeval' namespace
```

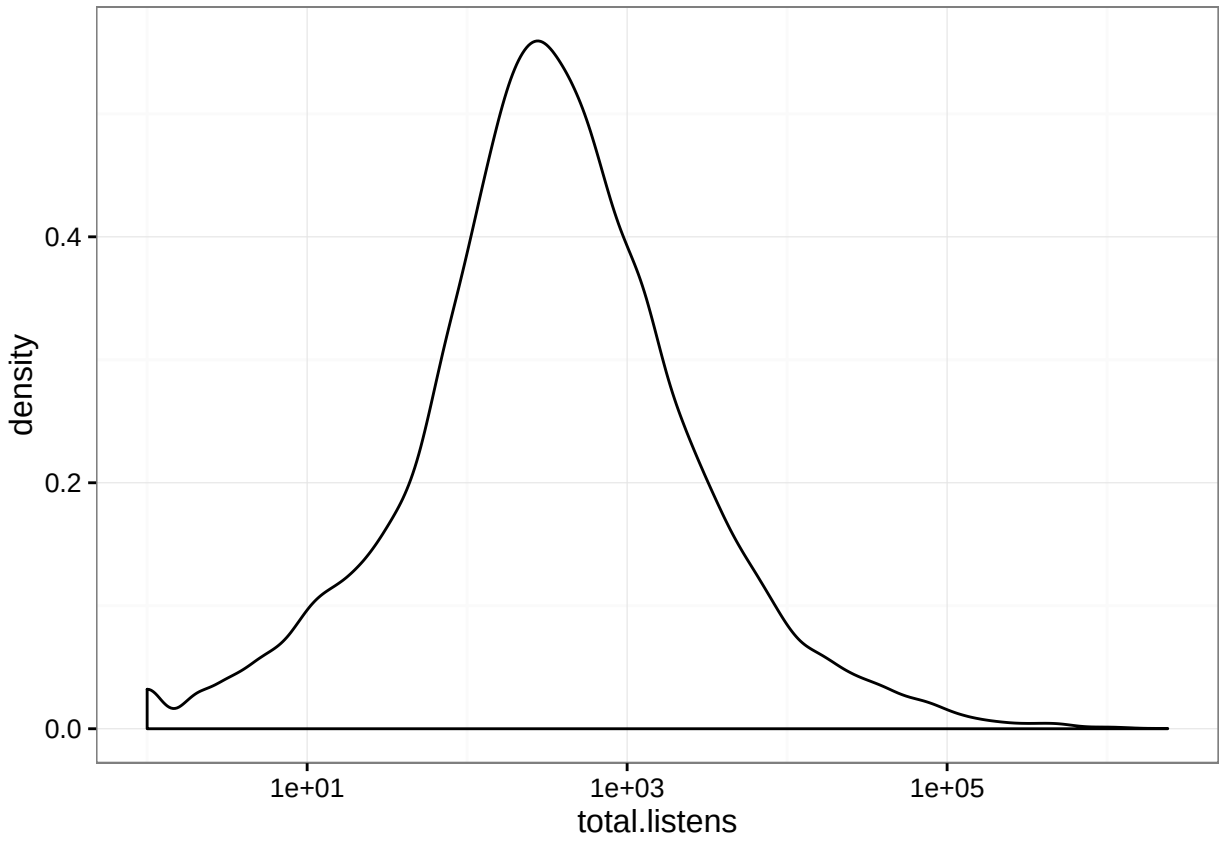
2.2 Exploratory Data Analysis

The median number for listens is 324 - quite a low number! The highest number of listens is Britney Spears with 2.3 million listens. We believe this dataset could be well modeled with a lognormal distribution.

```
summary(artist.listens$total.listens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     107     324    3858    1086 2393000
```

```
ggplot(artist.listens, aes(x = total.listens)) + geom_density() + theme_bw() + scale_x_log10()
```



Looking at the top 10 artists it is important to note that Britney Spears has nearly twice as many listens as the next most popular artist, Depeche Mode.

```
artist.listens %>% arrange(desc(total.listens)) %>% top_n(10) %>% kable()
```

```
## Selecting by total.listens
```

name	total.listens
Britney Spears	2393140
Depeche Mode	1301308
Lady Gaga	1291387
Christina Aguilera	1058405
Paramore	963449
Madonna	921198
Rihanna	905423
Shakira	688529
The Beatles	662116
Katy Perry	532545

We have decided to limit our recommender system to artist who have more than 900 total listens.

```
listens.cutoff <- 900
artist.cutoff <- artist.listens %>% filter(total.listens >= listens.cutoff)
```

2.3 User-Item Matrix Construction

Using the TidyR library the pairwise data was converted into a sparse matrix.

```
user.artists.matrix <- user.artists.pairwise %>%
  filter(name %in% artist.cutoff$name) %>% #filter artists with less than num_listens listens
  spread(key = name, value = weight) %>%
  distinct() %>%
  as.matrix()

rownames(user.artists.matrix) <- user.artists.matrix[,1]
user.artists <- user.artists.matrix[,-1]
dim(user.artists)
```

```
## [1] 1883 4608
```

2.3.1 Filtering out artists with less than 0.5% of the users listening to them

```
#Number of listeners cutoff rate: 0.5%
cutoff.rate <- 0.005

#Calculate cutoff numbner
n.users <- nrow(user.artists.matrix)
n.user.cutoff <- round(cutoff.rate * n.users)

#Empty vector to store counts
```

```
listener.count <- vector()

#For each artist, sum up number of users who listened to them
for (n in 2:ncol(user.artists.matrix)) {
  counts <- sum(!is.na(user.artists.matrix[,n]))
  listener.count <- c(listener.count, counts)
}

#Create logical vector for artists with less listeners than cutoff rate
listener.bool <- listener.count > n.user.cutoff

#Remove artists that didnt meet cutoff
mtx.cutoff <- user.artists.matrix[,listener.bool]

dim(user.artists.matrix)
```

```
## [1] 1883 4609
```

```
dim(mtx.cutoff)
```

```
## [1] 1883 1189
```

```
#new user.artists.matrix with
user.artists <- mtx.cutoff
```

2.3.2 Dealing with NAs

We tried two ways of dealing with NAs - imputing with column averages, and replacing with 0s.Note that due to the sparsity of the dataset, for most artists the column mean rounds to zero. In addition, the dataframe with column means was centered and scaled around 0 using the scale function. This gives us two views of the same dataset- one with NAs replaced with zeros which is relatively unmodified, and another with column means replacing NAs and normalized listen counts.

```
# http://stackoverflow.com/questions/25835643/replacing-missing-values-in-r-with-column-mean
user.artists.ave <- user.artists
for(i in 1:ncol(user.artists.ave)){
  user.artists.ave[is.na(user.artists.ave[,i]), i] <- mean(user.artists.ave[,i], na.rm = TRUE)
}

user.artists.ave <- scale(user.artists.ave)

user.artists[is.na(user.artists)] <- 0

user.artists.rrm <- new("realRatingMatrix", data = as(user.artists, "Matrix"))
user.artists.ave.rrm <- new("realRatingMatrix", data = as(user.artists.ave, "Matrix"))
```

2.4 Constructing a User-based Recommender System

First, we construct a user-based similarity matrix using the recommenderlab package. This will allow us to have a “sans-context” recommendation system to compare results when we use our recommendation system. We do this for both the averaged NAs and zerod NAs user-artist matrices.

```
# note igraph also has a function 'similarity' so the package must be specified.

sim.matrix.base.ave <- recommenderlab::similarity(user.artists.ave.rrm, method = "cosine", which = "use

sim.matrix.base.zeros <- recommenderlab::similarity(user.artists.rrm, method = "cosine", which = "users
```

3 Social Network Data

```
user.friends <- read_delim('hetrec2011-lastfm-2k/user_friends.dat', delim = "\t")
```

Now we ensure the analysis only goes forward with users that were contained in the dataset of user / artist listens.

```
user.IDs <- unique(user.artists.pairwise$userID)

length(user.IDs)
```

```
## [1] 1891
```

```
user.friends <- filter(user.friends, userID %in% user.IDs)

length(unique(user.friends$userID))
```

```
## [1] 1891
```

```
user.friends$friend <- 1
user.friends.matrix <- user.friends %>% spread(key = userID, value = friend)
user.friends.matrix[is.na(user.friends.matrix)] <- 0

rownames(user.friends.matrix) <- user.friends.matrix$friendID
user.friends.matrix <- user.friends.matrix[,-1]
user.friends.matrix <- user.friends.matrix[-1,]
dim(user.friends.matrix)
```

```
## [1] 1891 1891
```

```
user.graph <- graph.adjacency(as.matrix(user.friends.matrix), mode = "undirected")

#plot(user.graph)
```

3.1 Graph Analysis

The proportion of present edges from all possible edges in the network.

```
edge_density(user.graph)
```

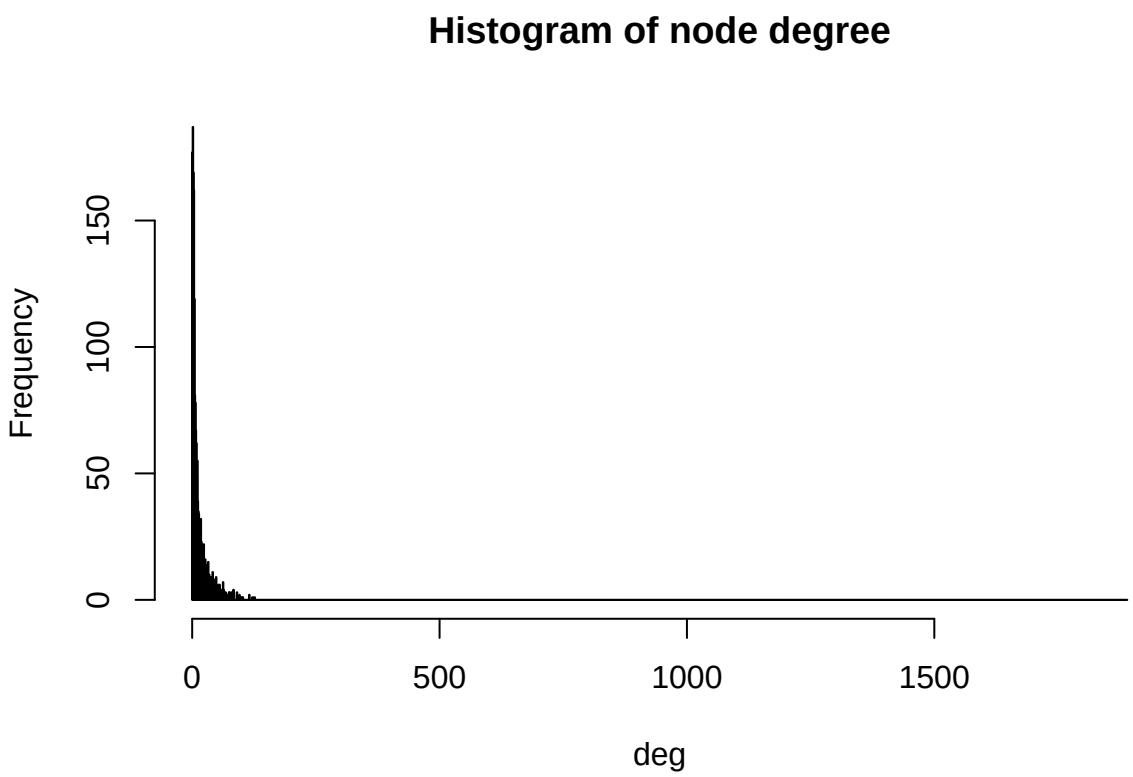
```
## [1] 0.007813116
```

A network diameter is the longest geodesic distance (length of the shortest path between two nodes) in the network.

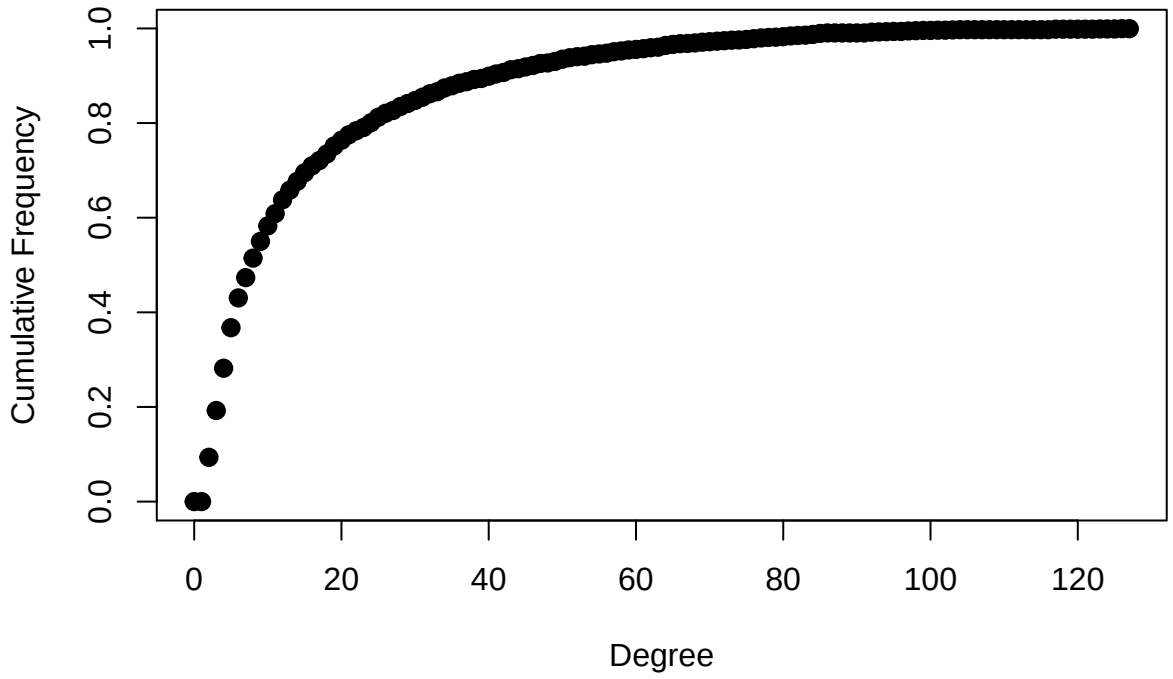
```
diameter(user.graph)
```

```
## [1] 9
```

```
# http://kateto.net/networks-r-igraph
deg <- degree(user.graph, mode="all")
#plot(user.graph, vertex.size=deg*3)
hist(deg, breaks=1:vcount(user.graph)-1, main="Histogram of node degree")
```




```
deg.dist <- degree_distribution(user.graph, cumulative=T, mode="all")
plot( x=0:max(deg), y=1-deg.dist, pch=19, cex=1.2, xlab="Degree", ylab="Cumulative Frequency")
```




```
mean_distance(user.graph, directed=F)
```

```
## [1] 3.409826
```

3.1.1 Communities

```
cfg <- cluster_fast_greedy(as.undirected(user.graph))
plot(cfg, as.undirected(user.graph))
```

```
ceb <- cluster_edge_betweenness(user.graph)
plot(ceb, user.graph)
```

3.2 Shortest Paths using Dijkstra's algorithm

Computing the distance matrix:

```
D <- shortest.paths(user.graph, algorithm = "dijkstra")
D[D == Inf] <- NA
```

The paper assumes that social influence will decay exponentially as the social-network distance increases. Therefore, the distance matrix is transformed to the influence matrix $I = (ist)s, t = 1, 2, \ldots, M$ via: $i_s t = exp(-d_s t)$.

Computing the influence matrix:

```
I <- exp(D * (-1))
dim(I)
```

```
## [1] 1891 1891
```

3.3 Adding Context via the Influence Matrix

```
sim.matrix.base.zeros.m <- as.matrix(sim.matrix.base.zeros)
dim(sim.matrix.base.zeros.m)
```

```
## [1] 1883 1883
```

```
#sim.matrix.zeros.I <- I %*% sim.matrix.base.zeros.m
```