Basic Ideas of Finding Similarities

Ao Liu

April 12, 2016

Ideas

In order to recommend the similar movie based on the input of users, we try to develop an algorithm to calculate the degree of similarity between two movies.

```
data<-read.csv("datawithnameandgenre.csv")
head(data,5)
```

```
##   X V1 product_productid  review_userid review_helpfulness review_score
## 1 1  1        B003AI2VGA A141HP4LYPWMSR                7/7            3
## 2 2  2        B003AI2VGA A328S9RN3U5M68                4/4            3
## 3 3  3        B003AI2VGA A1I7QGUDP043DG               8/10            5
## 4 4  4        B003AI2VGA  AQJVNDW6YZFQS               3/11            1
## 5 5  5        B00006HAXW  AD4CDZK7D31XP              64/65            5
##                                       V1.1    V2
## 1                       The Virgin of Juarez Drama
## 2                       The Virgin of Juarez Drama
## 3                       The Virgin of Juarez Drama
## 4                       The Virgin of Juarez Drama
## 5 Rock Rhythm & Doo Wop: Greatest Early Rock Drama
```

First we try to decide which factor we can use to calculate the similarity, according to the characteristic of our dataset, we tend to use the reviewrs' scores, **if the scores of the two movies are close, then the two movies are considered to be similar**. However, we cannot ignore the fact that different movies are not review by exactly the same people, different people have different attitude towards the same movie, so it makes no sense if we simply take these scores and calculate the similarity.

A better approach is that we **only focus on the users who have watched both movies and take their scores into consideration**, which makes sense. Let v_1 be the common users' scores for the first movie and v_2 be the common users' scores for the second, using simple R programming function, we can get the two vectors. Then the similarity between the two movies become the similarity between v_1 and v_2: the more they overlap, the more similar they are.

```
#the first function gives the users who rate both movies
 common_reviewers_by_id <- function(movie1,movie2) {
   reviews1 <- subset(data, V1.1==movie1)
   reviews2 <- subset(data, V1.1==movie2)
   reviewers_sameset <- intersect(reviews1['review_userid'],
                                  reviews2['review_userid'])
   if (length(reviewers_sameset)<2) {
     NA
   }
   else{
     reviewers_sameset
   }
 }
```

```
  #the second function gives the ratings of these common users
  get_review_metrics <- function(movie,id){
    metrics<-subset(data,V1.1== movie & review_userid %in% id)
    metrics<-metrics[order(unique(metrics$review_userid)),]
    metrics
  }
```

The next question is how to calculate the degree they overlap, my solution is to to calculate the angle θ they form, **the smaller the θ, the greater they overlap**. According to a conclusion of mathematics, we can calculate the cosine of a movie with following equation:

$$cos(\theta) = \frac{\vec{v}_1 \bullet \vec{v}_2}{||\vec{v}_1||||\vec{v}_2||}$$

```
get_similarity <- function(movie1){
  id <- common_reviewers_by_id(movie1, insert)
  if (any(is.na(id))) {
    return (NA)
  }else{
    metrics1 <- get_review_metrics(movie1,id)
    metrics2 <- get_review_metrics(insert,id)
    #the helpfulness is multiplied by the corresponding rating score:
    product <- function(metrics){
      names <- names(metrics)
      temp1 <- sapply(as.matrix(metrics[names[1]]), function(x) eval(parse(text=x)))
      temp2 <- sapply(as.matrix(metrics[names[2]]), function(x) eval(parse(text=x)))
      temp1*temp2
    }
    similarity <- product(metrics1)%*%product(metrics2)/
      (product(metrics1)%*%product(metrics1)*product(metrics2)%*%product(metrics2))^0.5
    if (similarity == 1){
      similarity = NA
    }
    similarity
  }
}
```

Implementation:

Thus, when a user insert a movie name to our recommendation app, we calculate its similarity with all the other movies with the same genre(takes a while to complete computation...), and then we rank all the similarity values and then get the movie which shares the greatest similarity with the user's input.

Take the movie *"The Last Samurai"* as an example:

```
#how to use the function:
#inert a movie name, and then run the following function,
#the output is the most similar movie suggested by the algorithm
insert<-"The Last Samurai"
index<-which(data[,7]== insert)[1]
data2<-data[which(data[,8]==data[index,8]),]
system.time(similarity<-sapply(as.matrix(unique(data2[,7])),get_similarity))
```

```
##    user  system elapsed
##   0.973   0.072   1.050
```

```
similarity <- na.omit(similarity,insert)
names(similarity[which.max(similarity)])
```

```
## [1] "2 Days in Paris"
```

Improvements

There could be some more improvements to this algorithm, due to the limitation of our data set, we only search the movies with the same genre. However, we have to know that there are many reasons that a person loves a movie, not just genre, but also directors or the cast. If we have more information within our data set, we will **also search within the movie with the same director, cast, etc**, which I think will present a better search result to the user.

What's more, currently we only show one movie with the highest similarity, in order to give more choice to the user and avoid some weierd scenario where "the most similar" movie is not the one the user really like, we can **increse the number of movies in the search result**.