Assignment 7

CS532-s16: Web Sciences

Spring 2016

John Berlin

Generated on March 31, 2016

1

Question

```
1.  Find 3 users who are closest to you in terms of age,
gender, and occupation.  For each of those 3 users:

- what are their top 3 favorite films?
- bottom 3 least favorite films?

Based on the movie values in those 6 tables (3 users X (favorite +
least)), choose a user that you feel is most like you.  Feel
free to note any outliers (e.g., "I mostly identify with user 123,
except I did not like ‘‘Ghost’’ at all").

This user is the "substitute you".
```

Answer

The code for this is found in listing 1 The top three users like me are:

```
user(551, 25, M, programmer)
top three movies:
(Young Guns (1988),5),
(Star Trek: First Contact (1996),5),
(Star Trek VI: The Undiscovered Country (1991),5)
bottom three movies:
(To Die For (1995),1),
(Naked Gun 33 1/3: The Final Insult (1994),1),
(Bram Stoker’s Dracula (1992),1)

user(595, 25, M, programmer)
top three movies:
(Willy Wonka and the Chocolate Factory (1971),5),
(Godfather, The (1972),5),
(Star Wars (1977),5)
bottom three movies:
(Phantom, The (1996),1),
(Down Periscope (1996),1),
(Volcano (1997),1)

user(622, 25, M, programmer) -Me
top three movies:
(Fargo (1996),5),
(Star Trek: First Contact (1996),5),
(Much Ado About Nothing (1993),5)
```

```
bottom three movies:
(Tin Cup (1996),1),
(Under Siege 2: Dark Territory (1995),1),
(Striptease (1996),1)
```

I choose user 622 because he had Fargo(ok Wade) and a Star Trek movie in his top three. Even though I Much Ado About Nothing is not a movie id watch all the time it was very good. His bottom three movies I had not seen and only heard of Under Siege 2,which I thought looked dumb. Thus I decided he was the closes fit to me.

2

Question

```
2.  Which 5 users are most correlated to the substitute you? Which
5 users are least correlated (i.e., negative correlation)?
```

Answer

```
Top 5 correlated users:
user(819, 59, M, administrator) correlation: 1.0
user(813, 14, F, student) correlation: 1.0
user(3, 23, M, writer) correlation: 1.0
user(762, 32, M, administrator) correlation: 0.891042111213631
user(909, 50, F, educator) correlation: 0.8703882797784892


Bottom 5 correlated users:
user(832, 24, M, technician) correlation: -1.000000000000004
user(242, 33, M, educator) correlation: -0.8017837257372769
user(31, 24, M, artist) correlation: -0.7660323462854247
user(462, 19, F, student) correlation: -0.745355992499929
user(565, 40, M, student) correlation: -0.7372097807744856
```

As seen in the 5 users who are most correlated to me I tend to like more older movies which in real life is true. Thus I am more correlated to older people except the 14yr old girl. Interesting.

3

Question

```
3.  Compute ratings for all the films that the substitute you
have not seen.  Provide a list of the top 5 recommendations for films
that the substitute you should see.  Provide a list of the bottom
```

5 recommendations (i.e., films the substitute you is almost certain to hate).

Answer

```
top five recommened movies:
Maya Lin: A Strong Clear Vision (1994) 5.0,
Tough and Deadly (1995) 5.0,
Prefontaine (1997) 5.0,
Aiqing wansui (1994) 5.0,
Saint of Fort Washington, The (1993) 5.0

bottom five recommened movies:
Girl in the Cadillac (1995) 1.0,
Turbo: A Power Rangers Movie (1997) 1.0,
Vie est belle, La (Life is Rosey) (1987) 1.0,
King of New York (1990) 1.0,
Hostile Intentions (1994) 1.0,
```

All of the movies recommended here I have not even heard of so I am wondering what is up.

4

Question

4. Choose your (the real you, not the substitute you) favorite and least favorite film from the data. For each film, generate a list of the top 5 most correlated and bottom 5 least correlated films. Based on your knowledge of the resulting films, do you agree with the results? In other words, do you personally like / dislike the resulting films?

Answer

I attempted the problem but it did not work well. I should have not tried to do this part in another language.

4

Code

```
import java.io._
import java.nio.charset.CodingErrorAction
import java.util.StringJoiner

import scala.io.{Codec, Source}
```

```
6  import jberlin._
7
8  import scala.collection.mutable
9
10
11 object Main {
12
13   def writeToFile(name: String, lines: mutable.MutableList[String])
       = {
14     val fout = new File(name)
15     val fos = new FileOutputStream(fout)
16     val bw = new BufferedWriter(new OutputStreamWriter(fos))
17     lines.foreach(l => {
18       bw.write(l)
19       bw.newLine()
20     })
21
22     bw.close()
23   }
24
25   def umap(line: String): user = line.split("\\|")
26
27   def urmap(line: String): urating = line.split("\\s+")
28
29   def mvmap(line: String): (Int, movie) = {
30     def help(m: movie) = (m.mid, m)
31     val ar = line.replaceAll("\\|\\|", "\\|").split("\\|")
32     help((ar(0), ar(1)))
33   }
34
35
36   def findCloseToMe(users: List[user]) = {
37     val me = 622
38     val likeme = users.filter(u => u.age == 25 && u.job.equals("
       programmer")).slice(1, 4)
39     var ml = mutable.MutableList[String]()
40     likeme.foreach(u => {
41       println(u)
42       ml += u.toString
43       val mrs = u.mRatings.values.toList.map(ur => (ur.mname, ur.
       rating)).sortBy(_._2).reverse
44       ml += "top three movies: " + mrs.take(3).foldLeft(new
       StringJoiner(","))((jnr, ur) => jnr.add(ur.toString())).
       toString
45       ml += "bottom three movies: " + mrs.tail.drop(mrs.length - 4)
       .foldLeft(new StringJoiner(","))((jnr, ur) => jnr.add(ur.
       toString())).toString + "\n"
46       println(mrs.take(3))
47       println(mrs.tail.drop(mrs.length - 4))
48       println("==============================")
49     })
50     writeToFile("top3.txt", ml)
51     likeme.slice(2, 3).head
52   }
53
54   def persons(likeme: user, u: user): Double = {
55     val mkeys = likeme.mRatings.keySet intersect u.mRatings.keySet
```

```
56      if (mkeys.isEmpty) {
57        u.corToMe = 0.0
58        return 0.0
59      }
60      val myMovies = likeme.mRatings.filterKeys(k => mkeys contains k
        ).values.toList.sortBy(_.itemId)
61      val otherGuys = u.mRatings.filterKeys(k => mkeys contains k).
        values.toList.sortBy(_.itemId)
62
63      val (sum1, sum2, sum1sq, sum2sq, sump) = (myMovies zip
        otherGuys).foldLeft((0.0, 0.0, 0.0, 0.0, 0.0)) {
64        case ((as1, as2, asq1, asq2, asp), (mr, ur)) =>
65          (as1 + mr.rating, as2 + ur.rating, asq1 + math.pow(mr.
        rating, 2.0),
66            asq2 + math.pow(ur.rating, 2.0), asp + (mr.rating * ur.
        rating))
67      }
68      val n = mkeys.size
69      val num = sump - (sum1 * sum2 / n)
70      val den = math.sqrt((sum1sq - math.pow(sum1, 2.0) / n) * (
        sum2sq - math.pow(sum2, 2.0) / n))
71      val r = if (den == 0) 0.0 else num / den
72      u.corToMe = r
73      r
74    }
75
76    def pMovies(movie:List[(String,Int,Int)],other:List[(String,Int,
        Int)]) ={
77      val (sum1, sum2, sum1sq, sum2sq, sump) = (movie zip other).
        foldLeft((0.0, 0.0, 0.0, 0.0, 0.0)) {
78        case ((as1, as2, asq1, asq2, asp), (mr, ur)) =>
79          (as1 + mr._3, as2 + ur._3, asq1 + math.pow(mr._3, 2.0),
80            asq2 + math.pow(ur._3, 2.0), asp + (mr._3 * ur._3))
81      }
82      val n = movie.size
83      val num = sump - (sum1 * sum2 / n)
84      val den = math.sqrt((sum1sq - math.pow(sum1, 2.0) / n) * (
        sum2sq - math.pow(sum2, 2.0) / n))
85      val r = if (den == 0) 0.0 else num / den
86      r
87    }
88
89
90
91    def correlatedUsers(likeme: user, users: List[user]) = {
92      val usrCors = users.filter(u => u.id != likeme.id).map(u => (u,
         persons(likeme, u))).sortBy(_._2).reverse
93
94      var ml = mutable.MutableList[String]()
95
96      val top5cor = usrCors.take(5)
97        .foldLeft(new StringJoiner("\n"))((jnr, uc) => jnr.add(uc._1
        + " correlation: " + uc._2)).toString
98
99      ml += "Top 5 correlated users:\n" + top5cor + "\n"
100
101     val bottom5cor = usrCors.tail.drop(usrCors.length - 6)
```

```
102         .sortBy(_._2).foldLeft(new StringJoiner("\n"))((jnr, uc) =>
      jnr.add(uc._1 + " correlation: " + uc._2))
103
104     ml += "\nBottom 5 correlated users:\n" + bottom5cor + "\n"
105
106     writeToFile("topBottom5Correlated2.txt", ml)
107
108     usrCors
109   }
110
111   def movieRecomendations(likeme: user, correlation: List[(user,
      Double)]) = {
112     val mkeys = likeme.mRatings.keySet
113     correlation.filter(_._2 > 0.0).flatMap {
114       case (usr, cor) =>
115         val haventSeen = usr.mRatings.keySet.diff(mkeys)
116         usr.mRatings.filterKeys(k => haventSeen contains k)
117           .values.toList.sortBy(_.itemId).map { case ur => (ur.
      itemId, cor, ur.rating * cor) }
118     }.groupBy(_._1).map { case (mid, ratings) =>
119       val (sumsim, sumweight) = ratings.foldLeft((0.0, 0.0)) {
120         case (acum, (movid, cor, ws)) => (acum._1 + cor, acum._2 +
      ws)
121       }
122       (mid, sumweight / sumsim)
123     }.toList.sortBy(_._2).reverse
124   }
125
126   def main(args: Array[String]) {
127
128
129     val usrfile = "ml-100k/u.user"
130     val usrReviews = "ml-100k/u.data"
131     val movief = "ml-100k/u.item"
132
133     implicit val codec = Codec("UTF-8")
134     codec.onMalformedInput(CodingErrorAction.REPLACE)
135     codec.onUnmappableCharacter(CodingErrorAction.REPLACE)
136
137     val users = Source fromFile usrfile getLines() map { case line
      => umap(line) } toList
138     val movies = Source.fromFile(movief).getLines().map { case line
       => mvmap(line) }.toMap
139     val userReviews = Source.fromFile(usrReviews).getLines().map {
140       case line =>
141         val rating = urmap(line)
142         movies.get(rating.itemId) match {
143           case Some(m) => rating.mname = m.mtitle
144           case None => println("Bad juju movieTitle -> rating")
145         }
146         rating
147     }.toList.groupBy(_.uid)
148
149     users.foreach(u => {
150       userReviews.get(u.id) match {
151         case Some(ratings: List[urating]) =>
152           //          u.mRatings = ratings.sortBy(_.rating).reverse
```

```
153             u.mRatings = ratings.map(ur => (ur.itemId, ur)).toMap
154           case None => println("Bad juju")
155         }
156       })
157
158       val likeme = findCloseToMe(users)
159       val correlation = correlatedUsers(likeme, users)
160       val myRecomendation = movieRecomendations(likeme, correlation)
161       var ml = mutable.MutableList[String]()
162
163       ml += "top five recommened movies: " + myRecomendation.take(5).
          foldLeft(new StringJoiner(","))( case (jnr,(mid, rating)) =>
164         movies.get(mid) match {
165           case Some(movie) => jnr.add(movie.mtitle +" "+ rating)
166         }
167       }.toString
168
169       ml += "bottom five recommened movies: " +  myRecomendation.drop
          (myRecomendation.length - 6).foldLeft(new StringJoiner(","))(
          case (jnr,(mid, rating)) =>
170         movies.get(mid) match {
171           case Some(movie) => jnr.add(movie.mtitle +" "+ rating)
172         }
173       }.toString + "\n"
174
175       writeToFile("topBottom5RecommendMovies.txt", ml)
176
177       println("==================")
178
179
180       val reviewsFlipped = users.flatMap(u => u.mRatings.values.map(
          ur => (ur.mname,ur.uid,ur.rating))).groupBy(_._1)
181
182       val clo = reviewsFlipped.get("Clockwork Orange, A (1971)")
          match {
183         case Some(x) => x
184       }
185
186       var mcor = reviewsFlipped.filterNot(it => it._1.equals("
          Clockwork Orange, A (1971)")).map(it => (it._1, pMovies(clo,it.
          _2))).toList.sortBy(it => it._2).reverse
187       ml = mutable.MutableList[String]()
188       ml += "top five correlated recommened movies: " + mcor.take(5).
          foldLeft(new StringJoiner(","))( case (jnr,(mid, rating)) =>
189
190           jnr.add(mid+" "+rating+"\n")
191       }.toString
192
193       ml += "bottom five recommened movies: " +  mcor.drop(mcor.
          length - 6).foldLeft(new StringJoiner(","))( case (jnr,(mid,
          rating)) =>
194         jnr.add(mid+" "+rating+"\n")
195       }.toString + "\n"
196
197       writeToFile("ClockworkOrangetopBottom5RecommendMovies.txt", ml)
198
199       println("==================")
```

```
200
201
202     mcor = reviewsFlipped.filterNot(it => it._1.equals("Jean de
        Florette (1986)")).map(it => (it._1, pMovies(clo,it._2))).
        toList.sortBy(it => it._2).reverse
203     ml = mutable.MutableList[String]()
204     ml += "top five correlated recommened movies: " + mcor.take(5).
        foldLeft(new StringJoiner(","))( case (jnr,(mid, rating)) =>
205
206       jnr.add(mid+" "+rating+"\n")
207     }.toString
208
209     ml += "bottom five recommened movies: " +  mcor.drop(mcor.
        length - 6).foldLeft(new StringJoiner(","))( case (jnr,(mid,
        rating)) =>
210       jnr.add(mid+" "+rating+"\n")
211     }.toString + "\n"
212
213     writeToFile("D3:TheMightyDuckstopBottom5RecommendMovies.txt",
        ml)
214
215     println("==================")
216
217
218
219
220     //474|Dr. Strangelove or: How I Learned to Stop Worrying and
        Love the Bomb
221     //179|Clockwork Orange, A (1971)
222
223
224   }
225 }
```

Listing 1: Recomender