ex44

August 12, 2022

[1]:
```
# Solution Ex. 44
```

[1]:
```
#inputPathWatched = "/data/students/bigdata-01QYD/ex_data/Ex44/data/
↪watchedmovies.txt"
#inputPathPreferences = "/data/students/bigdata-01QYD/ex_data/Ex44/data/
↪preferences.txt"
#inputPathMovies = "/data/students/bigdata-01QYD/ex_data/Ex44/data/movies.txt"
#outputPath = "res_out_Ex44/"
#threshold = 0.5

inputPathWatched = "data/Ex44/data/watchedmovies.txt"
inputPathPreferences = "data/Ex44/data/preferences.txt"
inputPathMovies = "data/Ex44/data/movies.txt"
outputPath = "res_out_Ex44/"
threshold = 0.5
```

[2]:
```
# Read the content of the watched movies file
watchedRDD = sc.textFile(inputPathWatched)
```

[3]:
```
# Select only userid and movieid
# Define an RDD or pairs with movieid as key and userid as value
movieUserPairRDD = watchedRDD.map(lambda line:  (line.split(",")[1], line.
↪split(",")[0]))
```

[4]:
```
# Read the content of the movies file
moviesRDD = sc.textFile(inputPathMovies)
```

[5]:
```
# Select only movieid and genre
# Define an RDD of pairs with movieid as key and genre as value
movieGenrePairRDD = moviesRDD.map(lambda line:  (line.split(",")[0], line.
↪split(",")[2]))
```

[6]:
```
# Join watched movies with movies
joinWatchedGenreRDD = movieUserPairRDD.join(movieGenrePairRDD)
```

[7]:
```
# Select only userid (as key) and genre (as value)
```

```
usersWatchedGenresRDD = joinWatchedGenreRDD.map(lambda pair: (pair[1][0],␣
 ↪pair[1][1]))
```

[8]:
```
# Read the content of preferences.txt
preferencesRDD = sc.textFile(inputPathPreferences)
```

[9]:
```
# Define an RDD of pairs with userid as key and genre as value
userLikedGenresRDD = preferencesRDD.map(lambda line: (line.split(",")[0], line.
 ↪split(",")[1]))
```

[10]:
```
# Cogroup the lists of watched and liked genres for each user
# There is one pair for each userid
# the value contains the list of genres (with repetitions) of the
# watched movies and the list of liked genres
userWatchedLikedGenres = usersWatchedGenresRDD.cogroup(userLikedGenresRDD)
```

[13]:
```
#userWatchedLikedGenres.mapValues(lambda v: (list(v[0]), list(v[1]))).collect()
```

[14]:
```
def misleadingProfileFunc(userWatchedLikedGenresLists):
    # Store in a local list the "small" set of liked genres
    # associated with the current user
    likedGenres = list(userWatchedLikedGenresLists[1][1])

    # Iterate over the watched movies (the genres of the watched movies)and␣
 ↪count
    # - The number of watched movies for this user
    # - How many of watched movies are associated with a not liked genre
    numWatchedMovies = 0
    notLiked = 0

    for watchedGenre in userWatchedLikedGenresLists[1][0]:
        numWatchedMovies = numWatchedMovies+1
        if watchedGenre not in likedGenres:
            notLiked = notLiked+1

    # Check if the number of watched movies associated with a non-liked genre
    # is greater that threshold%
    if float(notLiked) > threshold * float(numWatchedMovies):
        return True
    else:
        return False
```

[15]:
```
# Filter the users with a misleading profile
misleadingUsersListsRDD = userWatchedLikedGenres.filter(misleadingProfileFunc)
```

[16]:
```
# Select only the userid of the users with a misleading profile
misleadingUsersRDD = misleadingUsersListsRDD.keys()
```

```
[18]: #misleadingUsersRDD.collect()
```

```
[113]: misleadingUsersRDD.saveAsTextFile(outputPath)
```