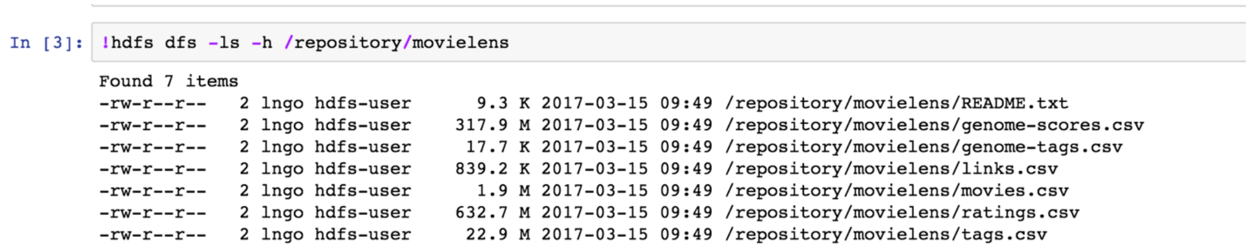
CPSC6770 Assignment #3 Report
Yuzhe Yang

Programming Questions

Utilize MapReduce to find the answers to these questions.

1. Find the mean, median, and standard deviation of the ratings for each of the movie genres. For each statistic (mean, median, or standard deviation), only use a single MapReduce program.

Note: the MovieLens dataset has been update in March 2017 by Dr. Ngo (as indicated in the screenshot below). Therefore, this might be the reason that the statistics I generated are different from the provided outputs which were calculated based on the Jan 2016 dataset version.



```
In [3]: !hdfs dfs -ls -h /repository/movielens

Found 7 items
-rw-r--r--   2 lngo hdfs-user      9.3 K 2017-03-15 09:49 /repository/movielens/README.txt
-rw-r--r--   2 lngo hdfs-user    317.9 M 2017-03-15 09:49 /repository/movielens/genome-scores.csv
-rw-r--r--   2 lngo hdfs-user     17.7 K 2017-03-15 09:49 /repository/movielens/genome-tags.csv
-rw-r--r--   2 lngo hdfs-user    839.2 K 2017-03-15 09:49 /repository/movielens/links.csv
-rw-r--r--   2 lngo hdfs-user      1.9 M 2017-03-15 09:49 /repository/movielens/movies.csv
-rw-r--r--   2 lngo hdfs-user    632.7 M 2017-03-15 09:49 /repository/movielens/ratings.csv
-rw-r--r--   2 lngo hdfs-user     22.9 M 2017-03-15 09:49 /repository/movielens/tags.csv
```

I implemented mapper for the genre (*genreMapper.py*) and reducer for each statistic (*meanGenreReducer.py, medianGenreReducer.py, stdevGenreReducer.py*) as well as all three statistics in one reducer (*genreReducer.py*). I also performed in Mapper Reduction optimization. All codes are in *solution* folder.

Map & reduce without optimization:
genreMapper.py: for each movie, map its genre with its rating and pass the pair through pipeline
sort: sort the data by genre name alphabetically.
meanGenreReducer.py: sum the ratings of the same genre and divide it by the number of movies within that genre.
genreReducer.py: for movies within the same genre, add them into a list and take statistics of the list to get mean, meidan and stdev of the genre.

Map & reduce with in-mapper reduction:
In order to reduce the amount of data pass through the pipeline, in-mapper reduction can be used for optimization.

meanGenreMapper.py: use a genre dictionary to calculate the sum and the total number of ratings for the same genre within the mapper and pass the information through pipeline in json format. E.g. Action {"total_count": 1, "total_rating": 1.0}
meanGenreReducer.py: for each genre, retrieve the total ratings and total number from the dictionary and calculate the average.

The answer to question number 1:

Genre	Mean	Median	Standard Deviation (Sample)
Action	3.4545	3.5	1.0721
Adventure	3.5071	3.5	1.0675
Animation	3.6105	4	1.0317
Children	3.4166	3.5	1.1011
Comedy	3.4175	3.5	1.085
Crime	3.6785	4	1.0132
Documentary	3.7228	4	1.022
Drama	3.6743	4	1.0025
Fantasy	3.503	3.5	1.0868
Film-Noir	3.9408	4	0.9155
Horror	3.2753	3.5	1.1521
IMAX	3.6371	4	1.0275
Musical	3.5439	4	1.0627
Mystery	3.6615	4	1.0119
(no genres listed)	3.208	3.5	1.2311
Romance	3.5425	4	1.0468
Sci-Fi	3.4552	3.5	1.0916
Thriller	3.5127	3.5	1.0399
War	3.8033	4	0.9969
Western	3.5716	4	1.0256

All outputs are included in *assignment3_Q1.ipynb* and *assignment3_mapreduce-optimized.ipynb.*

2. Using a single MapReduce program, identify the user who provides the most rating. Which genre does this user watch the most?

Map & reduce without optimization:

userMapper.py: for each user, create a user dictionary. For each movie, add the user and genre list to the user dictionary with user as key and genre list as value. Pass the user and json file of user dictionary through pipeline

userReducer.py: Iterate through the users and their genre lists to find out the user with most ratings. For that user, iterate through the genre list to find out the genre with most ratings.

Map & reduce with in-mapper reduction:

userMapper.py: for each user, create a user dictionary. The key is the user id and value is a dictionary with user rating count and user rated genre statistics. Specifically, the user rated genre statistics are stored in a dictionary which has genre name as key and genre count as value. E.g. 1

{"genre": {"Action": 1, "Drama": 3, "Thriller": 2, "Sci-Fi": 1, "Comedy": 2, "Horror": 1, "Romance": 2, "Crime": 1}, "count": 7}

userReducer.py: from the user dictionary, the user with most ratings is retrieved with most count. For that user, the most rated genre and the count are found by finding the information in the genre dictionary

After optimization, these programs are available for the yarn command. Otherwise, it experiences out of memory error, since the data is too large for the memory.

User Identification

```
186590 -- Total Rating Counts: 13250 -- Most Rated Genre: Drama - 8026
```

All outputs are included in *assignment3_Q2.ipynb* and *assignment3_mapreduce-optimized.ipynb.*

Submission
Submit all electronic copies of the MapReduce programs that you implement AND also submit a printed electronic document (or screenshot) that provides the answers to the above questions.

All python codes and ipython notebooks are submitted. I also implemented optimization for this assignment which is included in the *solution/ MapReduce_with_Optimization* folder. I also copied the ipython notebook for optimization out to the root folder for your convenience.