**Exploring Recommender Systems and MapReduce**

Learning Objectives and Assignment Structure

The assignment consists of multiple parts spread over several weeks. Part-1 and Part-2 are pre-requisites for Part-3. Part-1 and Part-2 can be done in isolation as independent stand alone assignments.

1. Basic of MapReduce
2. Foundations of recommender systems (user-based, item-based, content-based)
3. Expressing components of a recommender system with MapReduce
4. Basics of MapReduce

Details of this part are in the sub-folder mapreduce/ a5-mapreduce-mrjob.docx . In consists of 3 problems to be solved with the MapReduce paradigm using Yelp’s library MrJob.

1. Foundations of Recommender Systems

Background reading for this is Chapter 2 “Making Recommendations” of the book Collective Intelligence by Toby Segaran. My pedological contribution is the spreadsheet recosys/user-based-and-item-based-recommendation.xlsx where students need to express the core algorithms for user-based and item-based recommendation in a spreadsheet. Class experience has been that this helps significantly lay the conceptual foundation for how recommender systems work.

1. MapReduce + Recommender Systems

With the background of Part-1 and Part-2, students explore implementing parts of a recommender system using MapReduce.

Background Reading:

1. the CACM article “Recommender Systems—Beyond Matrix Completion”
2. Amazon.com recommendations

Both are available in the folder recosys.

We will be working with the movielens data set available at <https://grouplens.org/datasets/movielens/> . For testing purposes download <http://files.grouplens.org/datasets/movielens/ml-latest-small.zip> . You will find 5 files: README.txt, links.csv, movies.csv, ratings.csv, tags.csv. You will primarily use the contents of ratings.csv which has the structure: userId,movieId,rating,timestamp.

Overall approach

We will base similarity between two movies on their Pearson correlation <https://en.wikipedia.org/wiki/Pearson_correlation_coefficient> . For each pair of movies compute the Pearson correlation coefficient. For example, if we have two movies m3 and m5, compute the correlation from the following table:

|  |  |  |
| --- | --- | --- |
|  | M3 | M5 |
| U2 | 1 | 3 |
| U5 | 4 | 2 |
| U11 | 5 | 1 |
| U17 | 3 | 4 |
| U21 | 4 | 2 |

We compute the correlation in a multi-stage mapreduce process:

Stage 1:

* Mapper1:
Input: userID, movieID, rating
Emit: Key: userID, Value: (movieID, rating)
* Reducer 1:
Input: key: userID, value: [(movieID,rating), ... ]
For each user, produce all combinations of pairs of movies. Be sure to ensure each pair is produced only once i.e., m1,m4 or m4,m1 and not both
Emit: Key (mi, mj) value: ri, rj

Stage 2: At this stage, actual users don’t matter any more

* Reducer2:
Input: Key: (mi, mj) pairs value: [(ri, rj), …]
We now have the movie vectors for computing the correlation
Output: correlation between mi, mj

At this point using the MapReduce framework we have computed the large item-item similarity matrix based on correlation.

Given a movie mk find movies that are similar to it and rank order them in descending order based on the correlation coefficient. Your code needs to run along the following lines, to find movies similar to the movie with ID 23.

% python mrjob-recosys.py -m 23