UNIVERSITY OF ILLINOIS AT CHICAGO, INFORMATION AND DECISION SCIENCES

Item-based Collaborative Filtering for Recommendation

GOAL

It is to understand item-based collaborative filtering for building an efficient recommendation system under a large amount of movie rating data: MovieLens data.

SOFTWARE

Hadoop 1.0.3
Java 1.6+

STEPS

1. Download the MovieLens 100K data from `http://grouplens.org/datasets/movielens/`.
2. Two jobs (2 mappers and 2 reducers): write your own Mappers and Reducers to compute similarity scores for all pairs of movies. Similarity measure could be Cosine similarity or Pearson correlation. The input is the dataset and the output are files in which each line would be like:

   movieID_1, movieID_2, similarity_score. (You don't need to generate movieID_2, movieID_1, similarity_score, because they are the same. The item similarity matrix is symmetric.)

For Cosine similarity of two vectors $\vec{X}$ and $\vec{Y}$, $S_{XY} = \frac{\sum_{i=1}^{n} X_i * Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}}$
For Pearson correlation similarity of two vectors $\vec{X}$ and $\vec{Y}$, $S_{XY} = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2}\sqrt{\sum (Y_i - \bar{Y})^2}}$

All calculation in this step would be offline (one-time effort). The results can be stored for future recommendation use.

The key-value pairs of two mappers and reducers are described below:

Your first mapper:
customerID, movieID, rating, timestamp ⇒ key: customerID, value: (movieID, rating)

Your first reducer:
customerID, (movieID, rating) ⇒ key: customerID, value would be a list of (movieID, rating): (movieID_1, rating_1) (movieID_2, rating_2) ⋯ (movieID_n, rating_n)

Your second mapper:
customerID, (movieID_1, rating_1) (movieID_2, rating_2) ⋯ (movieID_n, rating_n) ⇒ key: a pair of movies: (movieID_i, movie_j), value: a pair of corresponding ratings: (rating_i, rating_j)

Your second reducer:
(movieID_i, movie_j), (rating_i, rating_j) ⇒ key: a pair of movies: (movieID_i, movieID_j), value would be similarity score: S_{ij}.

3. (Sequential) write a simple java class to make top k (k could be an user input parameter in your program) movie recommendations for a given customer ID using weighted average strategy (please see slides). The inputs may contain k and similarity score generated from previous steps. The output would be top k movie names.

4. Write a README file to explain all source codes and how to run your programs.

Data Description

u.data —— The full u data set, 100000 ratings by 943 users on 1682 items. Each user has rated at least 20 movies. Users and items are numbered consecutively from 1. The data is randomly ordered. This is a tab separated list of user id | item id | rating | timestamp. The time stamps are unix seconds since 1/1/1970 UTC

u.item —— Information about the items (movies); this is a tab separated list of movie id | movie title | release date | video release date | IMDb URL | unknown | Action | Adventure | Animation | Children's | Comedy | Crime | Documentary | Drama | Fantasy | Film-Noir | Horror | Musical | Mystery | Romance | Sci-Fi | Thriller | War | Western | The last 19 fields are the genres, a 1 indicates the movie is of that genre, a 0 indicates it is not; movies can be in several genres at once. The movie ids are the ones used in the u.data data set.

SUBMISSION

You need to submit one zip file, which contain all source codes (*.java files) and README file. The name of your zip file would be following the format like: Firstname_Lastname_Assignment4.zip. Please follow this format and submit it through the blackboard. Please try to comment your codes for critical sections and make your codes as readable as possible.