Abstract

In this paper, we use item-based and user-based collaborative filtering recommendation algorithms to predict user’s rating on the Movielens[3] dataset. We use RMSE as our metric to evaluating the performance of these two algorithm with different parameters. Besides, we implemented the scalable item-based recommender system over MapReduce. By comparing with the in-memory algorithm, we can find that the parallelized implementation is much more efficient on large datasets.

Methods

We use both item-based and user-based collaborative filtering in our project. We use Pearson correlation as our similarity function. The basic idea of CF is to use similar user or item’s rating to predict unrated items for a user. In both cases we choose k users or items with the highest similarity for further prediction. For user-based algorithm, we predict user’s rating to item using the following equation.[4]

\[
    r_{u,i} = \frac{\sum_{u' \in U} \left( r_{u,i} \right) \times \text{simil}(u, u')}{\sum_{u' \in U} \text{simil}(u, u')}
\]

In order to parallelize the process of building the recommender with high scalability, we present an implementation of item-based collaborative filtering based on MapReduce[1] in 3 phases:

Phase 1. Preprocessing: transform the raw data for the next phases

\[
    \text{map(line)} \rightarrow \text{list}(k: \text{userId}, v: \text{list(itemId, rating)})
\]

\[
    \text{reduce}(k: \text{userId}, v: \text{list(itemId, rating)}) \rightarrow \\
    k: \text{userId}, v: \text{list(itemId, rating)}
\]

Phase 2. Calculating Similarity: output item-item similarity matrix

\[
    \text{map}(k: \text{userId}, v: \text{list(itemId, rating)}) \rightarrow \\
    \text{list}(k: \text{itemId}, v: \text{list(itemId, rating)})
\]

\[
    \text{reduce}(k: \text{itemId}, v: \text{list(itemId, rating)}) \rightarrow \\
    k: \text{itemId}, v: \text{similarity}
\]

Phase 3. Recommendation: take all item similarities and user ratings as input, calculate the ratings for unrated items based on the similarities of neighbors.

\[
    \text{map}(k: \text{userId}, v: \text{list(itemId, rating)}) \rightarrow \\
    \text{reduce}(k: \text{itemId}, v: \text{list(itemId, rating)})
\]

We compared the running time of MapReduce implementation with single-machine in-memory implementation. As shown in Figure 1, the growth of running time of mapred is slower than in-mem, which implies better scalability – when dataset is getting larger, the mapred implementation will have better performance.

Results

In our single machine experiment, we performed both user-based and item-based CF on 1M Movielens[3] dataset, and we use RMSE as our criteria to evaluate the performance of the algorithms.

Figure 1 illustrates the comparison between the performance of user-based and item-based algorithm.

From Figure 2 we can see a very low overlapping rate between two algorithms’ result, only half of the recommended items are same when both algorithms recommend 1500 items to a user.

In order to test the scalability of the distributed algorithm, we applied item-based recommendation on Movielens 10M dataset based on Apache Hadoop 1.2.1 [2]. The experiments ran on 12 Linux boxes, each with 2 x 4 Xeon CPU cores (LS420 - 2.50GHz), 2 x 4GB DDR2 RAMs and 1 x 1TB SCSI disk.

Figure 4 demonstrates the affectivity of MapReduce implementation in the recommendation phase. As can be seen, along with the increase of the number of users to be recommended, the in-mem implementation leads a linear trend of running time, while mapred remains a constant and much lower running time.

Conclusions

From our experiment, we can see that although item-based and user-based algorithm have similar accuracy, the items they recommend are quite different. Thus, we can say that the two algorithms’ result are complementary. Item-based algorithm recommend items that are similar to a user’s current preference, so the recommender system is not likely to give innovative recommendation, while user-based algorithm is able to recommend totally new items to a user. However, User-based algorithm have "cold-start" problem, while Item-based algorithm can avoid the problem.

Besides, we implement item-based recommendation over MapReduce framework. The result shows that, the parallelized implementation is much more efficient than single-machine version on large datasets. Along with the coming of the Big Data era, the scalable recommender system will play a more important role in varieties of applications for modern business.

References