Recommendation based on Sentiment Analysis in Big-Data Environment
Yamini V. Nikam1 M.B.Vaidya2
1,2Department of Computer Engineering
1,2Amruthvahini College of Engineering, Sangmner, India

Abstract— With the rapid development of Internet, the number of users and items of recommender system is growing exponentially. As a result, the single-node machine implementing these algorithms is time-consuming and unable to meet the computing needs of large data sets. Therefore, implementing the algorithm distributed will reduce the required computing time. To improve the performance, we proposed a distributed collaborative filtering recommendation algorithm combining Sentimental Analysis on Hadoop.
Key words: Hadoop, Big-Data

I. INTRODUCTION
Collaborative Filtering (CF) is a promising technique in recommender systems. It provides personalized recommendations to users based on a database of user preferences, from which users having similar tastes are identified. It then recommends to a target user items liked by other, similar users.

CF-based recommender systems can be classified into two major types depending on how they collect user preferences: user-log based and ratings based. User-log based CF obtains user preferences from implicit votes captured through users’ interactions with the system (e.g. purchase histories as in Amazon.com). Ratings based CF makes use of explicit ratings users have given items (e.g. 5-star rating scale as in MovieLens). Such ratings are usually in or can easily be transformed into numerical values (e.g. A to E).

Some review hubs, such as the Internet Movie Database (IMDb), allow users to provide comments in free text format, referred to as user reviews. User reviews can also be considered as type of “user ratings”, although they are usually natural language texts rather than numerical values. While research on mining user preferences from reviews, a problem known as sentiment analysis or sentiment classification, is becoming increasingly popular in the text mining literature, its integration with CF has only received little research attention.

A. Hybrid Recommendation Algorithm Based On Hadoop:
The collaborative filtering is the most successful technique in use of recommendation and is the most widely used recommendation algorithm in recommender system. Compared with the traditional user-based, item-based collaborative filtering algorithms, Slope One algorithm is simpler and more efficient. However, the algorithm relies on the targeting predicting item ratings which are already rated by a large number of users. If users have no similar interests with the predicting user, the rating of the predicting user will encounter a lot of interference. With the rapid development of Internet, the number of users and items of recommender system is growing exponentially. As a result, the single-node machine implementing these algorithms is time-consuming and unable to meet the computing needs of large data sets. Therefore, implementing the algorithm distributed will reduce the required computing time.

II. SYSTEM ARCHITECTURE

The steps are as follows-
1) Data Preparation
   – POS Tagging
   – Negation Tagging
   – Feature generalization
2) Review Analysis
3) Opinion Dictionary Construction
   MapReduce Job-1: Work of the first MapReduce job is to collect all the user rated both the items.
   MapReduce Job-2: second MapReduce job will find the similarity between items using correlation formula.

III. ALGORITHM
A. Algorithm-1:
Map-1 Emit the user_id as key and (item and rating) as value
Job-1
Input:-
key-line offset
value- row of input file contains (item_id,user_id,rating)
Output:-
Key- user_id
Value-(item_id,rating) pair
Require: Input dataset containing User_id, Item_id, rating fields
B. Procedure:
- user_id, item_id, rating = line.split('t')
- key=user_id
- value=item_id
- value.append(rating)
- emit(key, value)

C. Algorithm-2:
Reducer-1 For each user, emit a row containing their "postings"(item , rating pairs)
Input:-
key- user_id
value- Sequence of (user_id , rating)
Output:-
Key- user_id
Value-row contain all posting of user (item_id , rating)

D. Procedure:
- item_count = 0
- item_sum = 0
- final = []
- for item_id, rating in values
  - 
    - item_count += 1
    - item_sum += rating
    - final.append((item_id, rating))
- Key=user_id
- Value=item_count, item_sum, final
- Emit (key, value)

E. Algorithm-3:
Map-2: The output drops the user from the key entirely, instead it emits the pair of items as the key
Job-2: Require Output of job-1 as input to the job-2.
Input:-
key-user_id
value- Sequence of (item_id, rating)
Output:-
Key- (item_id , item_id)
Value-(rating , rating)

F. Procedure:
- item_count , item_sum, ratings = values
- for item1, item2 in combinations(ratings, 2)
  - 
    - key=(item1[0], item2[0])
    - value=(item1[1], item2[1])
    - Emit=(key , value)
  - }

G. Algorithm-4:
Reduce-2: Sum components of each co rating pair across all users who rated both item x and item y, then calculate correlation similarity.
Input:-
key- (item_id , item_id)
value- sequence of rating pair(rating , rating)
Output:-
Key- (item_id , item_id)
Value-(similarity , n)

H. Procedure:
- sum_xx, sum_xy, sum_yy, sum_x, sum_y, n = (0.0, 0.0, 0.0, 0.0, 0.0)
- sum_x += item_x
- n += 1
- item_pair, co_ratings = pair_key, lines
- item_xname, item_yname = item_pair
- for item_x, item_y in lines:
  - 
    - sum_xx += item_x * item_x
    - sum_yy += item_y * item_y
    - sum_xy += item_x * item_y
    - sum_y += item_y
    - similarity = normalized_correlation(n, sum_xx, sum_x, sum_y, sum_xx, sum_yy)
- Key=(item_xname, item_yname)
- Value=(similarity)
- Emit=(key, value)
- }

IV. PERFORMANCE ANALYSIS

In this chapter, presenting the results of performance evaluation of practical implementation. Performance of system is measured by comparing with previous work, algorithm and time complexity. In this project, IMDB’s MovieLens dataset is used. First system performance of the data set analyzed and then algorithm. Our goal in the performance analysis is to optimize algorithm and compare results with other algorithms.

A. Data Set:
MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota. The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. Members of the GroupLens Research Project are involved in many research projects related to the fields of information filtering, collaborative filtering, and recommender systems. The project is lead by professors John Riedl and Joseph Konstan. The project began to explore automated collaborative filtering in 1992, but is most well known for its world wide trial of an automated collaborative filtering system for Usenet news in 1996. The technology developed in the Usenet trial formed the base for the formation of Net Perceptions, Inc., which was founded by members of GroupLens Research. Since then the project has expanded its scope to research overall information filtering solutions, integrating in content-based methods as well as improving current collaborative filtering technology.

This data set consists of:
- 100,000 ratings (1-5) from 943 users on 1682 movies.
- Each user has rated at least 20 movies.
- Simple demographic info for the users (age, gender, occupation, zip)

The data was collected through the MovieLens website (movielens.umn.edu) during the seven-month period from September 19th, 1997 through April 22nd, 1998. This data has been cleaned up - users who had less than 20 ratings or did not have complete demographic information were removed from this data set.
V. RESULTS ANALYSIS

1) For dataset size 12500 reviews

<table>
<thead>
<tr>
<th>Recommendation Type</th>
<th>MAE</th>
<th>RMSE</th>
<th>AvgP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Recommendation</td>
<td>1.0238</td>
<td>1.2529</td>
<td>0.697</td>
</tr>
<tr>
<td>Collaborative Recommendation</td>
<td>0.8093</td>
<td>1.0347</td>
<td>0.7785</td>
</tr>
<tr>
<td>SO+ Collaborative Recommendation</td>
<td>0.7952</td>
<td>1.0224</td>
<td>0.7988</td>
</tr>
<tr>
<td>Hadoop Recommendation</td>
<td>0.7690</td>
<td>0.9683</td>
<td>0.7715</td>
</tr>
</tbody>
</table>

Table 1:

2) For dataset size 25,000 reviews

<table>
<thead>
<tr>
<th>Recommendation Type</th>
<th>MAE</th>
<th>RMSE</th>
<th>AvgP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Recommendation</td>
<td>1.0648</td>
<td>1.2971</td>
<td>0.6819</td>
</tr>
<tr>
<td>Collaborative Recommendation</td>
<td>0.8490</td>
<td>1.0837</td>
<td>0.8016</td>
</tr>
<tr>
<td>SO+ Collaborative Recommendation</td>
<td>0.7945</td>
<td>1.0248</td>
<td>0.8096</td>
</tr>
<tr>
<td>Hadoop Recommendation</td>
<td>0.7950</td>
<td>0.9688</td>
<td>0.8115</td>
</tr>
</tbody>
</table>

Table 2:

3) For dataset size 50,000 reviews

<table>
<thead>
<tr>
<th>Recommendation Type</th>
<th>MAE</th>
<th>RMSE</th>
<th>AvgP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Recommendation</td>
<td>1.0874</td>
<td>1.2718</td>
<td>0.6817</td>
</tr>
<tr>
<td>Collaborative Recommendation</td>
<td>0.8502</td>
<td>1.0490</td>
<td>0.802</td>
</tr>
<tr>
<td>SO+ Collaborative Recommendation</td>
<td>0.7535</td>
<td>0.9693</td>
<td>0.8085</td>
</tr>
<tr>
<td>Hadoop Recommendation</td>
<td>0.7422</td>
<td>0.9493</td>
<td>0.8115</td>
</tr>
</tbody>
</table>

Table 3:

4) For dataset size 1,000,000 reviews

<table>
<thead>
<tr>
<th>Recommendation Type</th>
<th>MAE</th>
<th>RMSE</th>
<th>AvgP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Recommendation</td>
<td>0.9983</td>
<td>1.2424</td>
<td>0.6641</td>
</tr>
<tr>
<td>Collaborative Recommendation</td>
<td>0.8179</td>
<td>1.0263</td>
<td>0.7981</td>
</tr>
<tr>
<td>SO+ Collaborative Recommendation</td>
<td>0.7203</td>
<td>0.9322</td>
<td>0.8119</td>
</tr>
<tr>
<td>Hadoop Recommendation</td>
<td>0.7154</td>
<td>0.9314</td>
<td>0.8149</td>
</tr>
</tbody>
</table>

Table 4:

VI. CONCLUSION

With the scale of recommender system continues to expand, the number of users and items of recommender system is growing exponentially. Many algorithm need to meet the computing needs of large data sets. In this paper, the parallel hybrid recommendation algorithm can run on a number of low-cost computers. It improves speed and solves high recommendation performance under large data sets. Experiments show that the improved parallel hybrid recommendation algorithm compared with the former one improves the speed and reduces the time consumption.

We proposed a rating inference approach to integrating sentiment analysis and CF. Such approach transforms user preferences expressed as unstructured, natural language texts into numerical scales that can be understood by existing CF algorithms.

VII. FUTURE WORK

As noted, our rating inference approach transforms textual reviews into ratings to enable easy integration of sentiment analysis and CF. We nonetheless recognize the possibility to perform text-based CF directly from a collection of user reviews. A possible solution is to model text-based CF as an information retrieval (IR) problem, having reviews written by a target user as the “query” and those written by other similar users as the “relevant documents”, from which recommendations for the target user can be generated. This remains as an interesting research direction for future work.

VIII. REFERENCES


