2015 IEEE International Conference on Computational Intelligence & Communication Technology

Building a Distributed Generic Recommender Using Scalable Data Mining Library

Lavannya Bhatia
Department of CSE,
JSS Academy of Technical Education,
Noida, India
lavannyabhatia@gmail.com

S.S.Prasad
Department of CSE,
JSS Academy of Technical Education,
Noida, India
profssprasad@jssaten.ac.in

***Abstract*— Recommender systems produce list of recommended items through content based or collaborative or hybrid combination of these two approaches. The paper presents a generic approach for performing collaborative filtering using data mining techniques to discover relationships among users and items. Using generic model techniques a single recommender system can produce recommendations about a variety of items. The methodologies reported for development of recommender systems are not efficient for generic application. The difference in the implementations of recommender depends upon how they analyze the big input data to recognize the similarity between users and items that indicates the relevant preferences for that user. Generic user based recommender works with data model encapsulating recommender input data in Apache Mahout which is extensible data mining library. The recommender system framework can use any similarity metric. We have chosen Pearson correlation because the computation would be fast. One of the parameters of user based recommender is User-Neighborhood. Fixed-size Neighborhood has the advantage that the recommendations are based on fewer similar users. Hadoop software library allows distributed processing of big data across multiple clusters of nodes. The paper describes an implementation using Apache Mahout and Hadoop and also explores feasible augmentation that can enhance efficiency of recommendation. This paper reports successful implementation of generic recommender.**

***Keywords*—Big data, collaborative filtering, data mining, generic recommender, Hadoop, machine learning, Mahout, recommender systems**

I. INTRODUCTION

Recommender systems are referred as the information filtering systems which attempt to predict the rating or preference which user would assign to an item. It is difficult for users to find relevant items as the count of preference increases and they become overwhelmed with high volume, variety and velocity of data. In order to assist users to cope with this problem, a recommender system can be used to find more related items in shorter time. Relevance of a recommended item to meet the user need can be estimated using data mining algorithms. It increases user fidelity to better interpret what the user wants. It learns from users' behaviors and recommends an item in which a user may be interested. It scans through big amount of data to recognize user preferences. Data mining algorithms are useful to evaluate similarities between items and users. The fundamental methodology used for developing recommender system is collaborative filtering.

Collaborative filtering [1] is an approach for resolving the task of recommending an item using the strength of preference specified by user. It does not require the knowledge about the properties of each item to create recommendations. Generic recommender system framework is not concerned about the kind of item. It recognizes association between users and items and makes connections to forecast the relevant item which matches with user interest. The input to algorithm for collaborative filtering consists of user, item and rating to build recommendations using any of the following ways:

A. *User based recommendations* are computed based on users with identical characteristics.

B. *Item based recommendations* are computed based on similar items.

C. Slope-one: In this recommender, similarity metric is not considered as standard component. It is fast and simple approach for item recommendation.

II. METHODOLOGIES

Negar Hariri et al [2] in their research paper present an approach to locate frequent features across items as well as to find association among those features .They use a novel incremental algorithm to extract features from online item descriptions. They utilize association rule mining and the k-nearest neighbor method to create feature recommendations in the process of domain analysis. However this paper reported efficient implementation for only online product listings.

Fong et al [3] in their research paper proposed a web recommender system which models habits and behaviors of users by a knowledge base to develop web access patterns. They used fuzzy representation in their approach to enhance the performance. Their proposed approach is applicable to distribution of recommendations on portable devices.

Marios et al [4] in their research paper proposed a collaboration based recommender in an Internet of things environment. Their approach relies on user to

978-1-4799-6023-1/15 $31.00 © 2015 IEEE
DOI 10.1109/CICT.2015.129

object space-time interaction patterns. They emphasize only on context aware filtering.

Melville et al [5] in their research paper proposed architecture for combining collaborative filtering and content filtering. Their method helps to overcome information overload by recommendations based on history of a user's preferences. Their approach does not consider unstructured big data.

However none of these methodologies are efficient for a generic recommender system .Our proposed generic recommender system is described below.

III. PROPOSED SYSTEM ARCHITECTURE

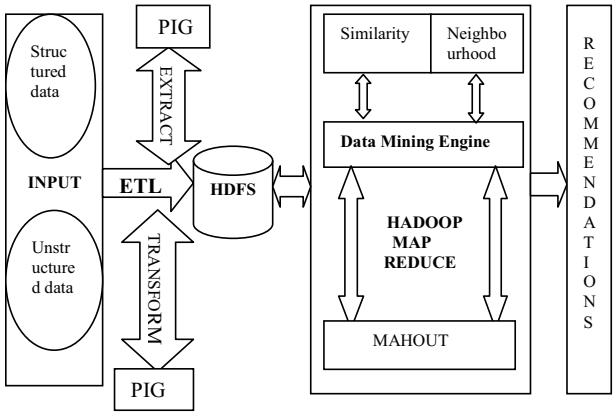

Figure1: Proposed Architecture for Generic Recommender System

A. Generic Recommendation Algorithm

There are several algorithms which are used to compute similarities between two users. The recommendation quality depends upon the algorithm we consider in the implementation framework. The difference in the implementations of recommender depends upon how they analyze the big input data to recognize the similarity between users and items that indicates the relevant preferences for that user.

1) Generic user based recommender

It generates recommendations on the basis of similarity among users. Due to dynamic properties of users, generic user based recommender systems are less scalable. Generic user based recommender works with the following components:

(a) Data model

It encapsulates recommender input data in Mahout and stores the resulting preferences. Data model implementation provides access to data required by recommender algorithms.

(b)User Similarity

There are many implementations to define similarity within Mahout [6].The recommender system framework can use any of the similarity metric such as Pearson correlation, Spearman correlation, log likelihood, Tanimoto coefficient etc. User similarity is one of the important parameters of user-based recommender.

(1) Pearson correlation based similarity

The Pearson correlation is defined as a number between –1 and +1. It computes the tendency of the numbers to move together proportionally. There is a linear relationship among the values in a particular series and another series. When the computed tendency is high, the Pearson correlation is approximately 1.

(2) Defining similarity by Euclidean distance

Euclidean distance similarity provides implementation that depends upon the distance between users. It considers users as points in multi dimension space whose coordinate values are preferences. If the similarity between users is higher, the evaluated value will be smaller.

(3) Defining similarity by cosine measure

It considers users as points in space. Either two users will be close in space or have same direction from the origin if they are similar. The angle between these two lines will be relatively smaller. This angle is used as the basis for cosine similarity metric which means that similarity value can be derived from the cosine of the angle formed between the lines.

(4) Defining similarity by Spearman correlation

It computes a correlation value using relative rank of preference value instead of computing the value using original preference value .It can be considered same as computation of Pearson correlation on the transformed values.

(5)Similarity by Tanimoto coefficient

This implementation of user-similarity completely ignores the preference values. It is also referred as Jaccard coefficient.

We have chosen Pearson correlation because it does not reflect the count of items over which it is computed and for our purpose that would be useful because the computation would be fast. Pearson correlation similarity provides an extension to the standard computation called weighting. The concept of weighting in Pearson correlation also improves the computed score. This way when the result is based on big data, the Pearson correlation similarity is more reliable.

(c)User Neighborhood

It covers the group of most similar users. It is also considered as one of the standard parameters of user based recommender.

(1) Fixed-size neighborhood

This implementation of neighbourhood locates most similar users by selecting a fixed number of closest neighbours. The distance illustrates similarity: farther means less similar .Recommendations would be based on fewer similar users but would exclude some less-similar users from consideration.

(2) Threshold-based neighborhood

In this neighborhood, we consider threshold value between –1 and 1.In proposed implementation we use Pearson correlation to evaluate the similarity in

which a value of 0.7 or above is referred as high correlation. It is better to select a similarity threshold and take any user which is at least that similar.

(d) Recommender

Our implementation drags these components together for developing recommendations for users. In order to enhance the performance of recommender we have transformed the input file into comma separated file (.csv) using Pig in the ETL step.

2) Generic item based recommender

We know that in case of item based recommender we evaluate similarity between items and develop recommendations. Generic item based recommender [7] uses the data model and item similarity components for evaluating recommendations.

3) Slope-one recommender

Slope-one recommender [8] is a fast runtime approach to evaluate recommendations .The negative point is that it requires plenty of time to pre-compute its internal data structures. In case of slope one recommender [9], similarity metric is not considered as standard component. It is known as 'slope-one' due to following reasons (a) It pre-supposes that there is linear association between values for one item and other, (b) it is required to compute the value for some item A based on the value for item B, considering a linear function like A = m*B + c, (c) It has a simplifying assumption that *m*=1 i.e. 'slope one'. It just remains to find c = A-B, the average difference in value, for each pair of items.

IV. IMPLEMENTATION

A. Experiment1: Implement Generic User Based Recommender

1) Creating the input

The input data is considered as base for recommendation. This input data is recognized as preferences in Mahout. The associations between items and users are defined as preferences. A preference is a combination of item ID, user ID, and user's rating for that item ID. Item ID and user ID in Mahout are integer values. In our implementation, we consider the values of ratings on a scale of 1 to 10, where 1 means user does not like the item, and 10 means favorites.

2) Creating components for recommender

The input file instance, which contains the preference value, is received by the constructor of file data model. File data instance is used to make an instance of user similarity and user neighborhood.

3) Examining the output recommendations

Each recommended item instance encapsulates recommended item-id and the user's rating for the item ID .The rating provided by the user is a float value.

To illustrate we give below the output for one user:

USERID: 102
RECOMMENDED ITEMID 90101.STRENGTH OF THE PREFERENCE: 10.000000
RECOMMENDED ITEMID 90051.STRENGTH OF THE PREFERENCE: 10.000000
RECOMMENDED ITEMID 90071.STRENGTH OF THE PREFERENCE: 9.000000
RECOMMENDED ITEMID 90111.STRENGTH OF THE PREFERENCE: 9.000000
RECOMMENDED ITEMID 90061.STRENGTH OF THE PREFERENCE: 8.000000
BUILD SUCCESSFUL (total time: 5 seconds)

B. Experiment2: Implement Slope-one Recommender

1) Creating the input

The Grouplens dataset [10] is a freely available dataset created by computer science researchers at Minnesota University. Input file consists of million ratings from 6000 users on 4000 items. This data is the association between the users and the item with a number which indicates how much the user likes the item.

2) Creating the instance of data model

To transform the rating.dat file from Grouplens format to the comma separated value file format we create a model class that will handle the format of the new ratings.csv file that we will use .Then we create the instance of file data model using a cycle to retrieve the entire users list contained on the ratings.csv for constructing recommendations for each user.

3) Analyzing the generated recommendations result

To illustrate we give below the output for one user:

User66RecommendedItem[item:2, value:1.0]
User66RecommendedItem[item:3, value:1.0]
User66RecommendedItem[item:4, value:1.0]
User66RecommendedItem[item:5, value:1.0]
User66RecommendedItem[item:6, value:1.0]
User66RecommendedItem[item:7, value:1.0]
User66RecommendedItem[item:2320, value:1.0]
User66RecommendedItem[item:10, value:1.0]
BUILD SUCCESSFUL (total time: 10 seconds)

C. Recommendation Evaluation

To evaluate the generated recommendation obtained from recommender engine [11] we require to create a predictor which will be utilized in predicting the preferences. To evaluate the quality we need to locate the difference between estimated preference value and actual preference value.

We summarize below in Table1 the features of our implementations along with the arguments and characteristics:

Recommender	*Argument*	*Characteristics*
Generic user based recommender	User similarity value, size, user neighborhood	Depends upon the count of users in the input data
Generic item based recommender	Item similarity	depends upon the number of items in the input file

Slope-one recommender	Diff storage Strategy	It is fast and simple strategy of recommendation.

TABLE 1: Recommender systems in Mahout with their argument and characteristics

1) Experiment 3: Evaluating Recommendations by Average Absolute Difference

If we consider big data file as input for the evaluation of quality of recommender then it will take considerable amount of time. For quick evaluation, it will be convenient to reduce the input data. But if we consider very less data then it weakens the accuracy of the results. In our implementation we gave two parameter values as 0.95 and 0.05 which means only 5 percent of all the data is used for evaluation. The parameter 0.95 indicates to build a model with 95 percent of the data, and the remaining 5 percent will be used as test data in the process of evaluation of recommender system.

2) Analyzing the generic recommender evaluation

The instance of file data model for the input file is created. The evaluation is performed using 0.95 of the total data. The file information is read by the file data model instance. In our implementation we get result evaluated as 0.769430722509117. The recommender evaluation score is 0.769431. BUILD SUCCESSFUL (total time: 7 seconds).

3) Experiment4: Evaluating Generic Recommender by Precision and Recall

Mahout provides components that enable us to evaluate the quality of the estimated preference values of the recommendations generated by a recommender.

Precision is referred as the fraction of recommendations that are relevant to user, and recall is referred as the fraction of relevant recommendations obtained from all other generated recommendations. In our implementation we get the following result:

Precision/recall/fall-out: 0.16666666666666669 / 0.33333333333333337 / 0.06461538461538463
The recommender precision is 0.100000.
The recommender recall is 0.200000
BUILD SUCCESSFUL (total time: 5 seconds)

V. GENERATING RECOMMENDATIONS IN DISTRIBUTED MODE

Hadoop[12] is a software for enabling a computation across clusters of thousands of nodes. It is simpler to set up a Hadoop cluster on our local machine for learning. The cluster implementation on our local machine is same as creating a proper cluster. In case of distributive mode the idea of file data model and user neighbourhood does not exist because data is distributed across multiple clusters of nodes. If we perform implementation of recommender using Mahout in distributed mode there are collections of mapper and reducer processes each with some intermediate result. The implementation process starts with the computation of co-occurrence matrix and user vectors. Apache Mahout is used to incorporate its extensible collective intelligence library that enables us to implement recommendation algorithms. It provides a collaborative framework to generate recommendations. It requires both JDK and Maven. It enables to start exploring machine learning algorithms easily. The two main strengths of Mahout are fast prototyping and evaluation. Mahout implementation framework is based on Java. Therefore the implemented code can be executed on any platform that uses Java virtual machine. Mahout provides a job to enable item based recommenders in distributed mode .It is required to input onto Hadoop distributed file system i.e. HDFS [13] to make it available to Hadoop because if the data is readily available on the local file-system, it needs to be copied again into HDFS. Because Hadoop is a software [14] which runs across many nodes, so any data it uses must be available not on single machine but on multiple machines in the cluster. HDFS is an entity that can make data available to many nodes. Our implementation is outlined below:

A. Creating Input Directory in HadoopDistributedFileSystem(HDFS)

An input directory is created in Hadoop distributed file system (HDFS) then we copy the input file from local file system to HDFS using the command:

./hadoop fs –copyFromLocal /mahout_hadoop/data.csv

B. Creating Output Directory in HDFS

An output directory is created in Hadoop distributed file system (HDFS) using the command:

./hadoop fs –mkdir /mahout_hadoop_output

C. Mahout Recommendations implementation using Hadoop Map-Reduce pipeline

The library that wraps the mappers and reducer components together is apache/mahout/cf/taste. Hadoop/item/recommenderjob. It is available within the Mahout source distribution. It constructs and implores the pipeline of map-reduce jobs .To run recommender in distributed mode using Hadoop we are required to locate all source code into a JAR file. This can be performed by using Maven package from the main directory folder of the Mahout distribution. The command shows the path of the main directory of Mahout distribution which is outlined below:

./hadoop jar /home/project/Downloads/mahout-distribution/mahout-corejob.jar

VI. RESULTS AND DISCUSSION

Results of our implemented experiments show that Mahout makes it simple to explore machine learning algorithms and data mining concepts, and we can work with existing data models and test different components to generate different kinds of recommendations. If we use Mahout alone it provides implementation framework for non-distributed collaborative filtering. Our experiments which run in series using Mahout and Hadoop together,

enable us to start working with recommender and begin evaluating their accuracy. Considering preferences of 10 users , the average difference between actual and estimated preference values, is 0.98 which is high .However when we try a neighborhood of 400 users, the result drops to 0.75, which is of course better.

To consider million preferences we require to implement recommender algorithm using distributed computing approach from Mahout based on map reduce paradigm and Apache Hadoop. Our implementation framework can handle a data set of million ratings on single node. It is also efficient in computing the recommendations in real time. Our implementation, utilizes open source platform of Apache Mahout with Hadoop. This implementation is platform independent and performs distributed map reduce computation [15][16].Our implementation framework indicates that use of Apache Hadoop with Mahout is most suitable for implementation of large scale and distributed generic recommender systems.

VII. CONCLUSION

Relevance of a particular item to meet user need can be estimated by a recommender system using data mining algorithms. Though several methodologies for the implementation of recommender have been reported but none of these are efficient to compute generic recommendations using unstructured big data.

Our generic recommender implementation deals with unstructured big data input.

We implemented a generic recommender system following user based and slope-one item based approaches using Mahout.

We evaluated the generated recommendation by Average Absolute Difference. We also compute precision and recall to evaluate the efficiency of implemented generic recommender system.

We have proposed an approach for generating recommendations in distributed mode using Apache Hadoop.

Based on hands-on exploration, experimentation, and evaluation on real big input data it is found feasible to develop an efficient generic recommender system using open source software like Apache Hadoop and Mahout.

VIII. REFERENCES

[1]Xingyuan Li et al,"Collaborative filtering recommendation algorithm based on cluster", International Conference on Computer Science and Network Technology (ICCSNT), Dec 2011, page(s):2682-2685.

[2] NegarHariri et al, "Supporting Domain Analysis through Mining and Recommending features from Online Product Listings", IEEE Trans. on Software Engineering, Vol.39, No.12, Dec 2013, pp.73-82.

[3]Fong et al, "Web Content Recommender System based on Consumer Behavior Modeling", IEEE Trans. on Consumer Electronics, Vol.57, No.2, May 2011.

[4] MarioMunoz et al, "A Collaborative Recommender System Based on Space-Time similarities" IEEE CS, Vol.10, No.6, 2010.

[5] Melville et al, "Content-Boosted Collaborative filtering for improved recommendations", Proceedings of the Eighteenth National Conference on Artificial Intelligence, July 2002, pp.187-192.

[6] "Apache Mahout", http://mahout.apache.org/.

[7] Guohui et al, "Applying User Interest on Item Based Recommender System", 5th IEEE International Joint Conference on Computational Sciences and Optimization, 2012, page(s):635-638.

[8] Lemire, D. and A. Maclachlan, "Slope One Predictors for Online Rating-Based Collaborative Filtering", Proceedings of SIAM Data Mining (SDM'05), 2005.

[9] Ziqing Zhang et al, "Applying User Favorite Item Based Similarity into Slope-One Scheme for Collaborative filtering", World Congress on Computing and Communication Technologies (WCCCT),2014, page(s):5-7.

[10] "GroupLens data set", http://www.grouplens.org/.

[11] Patil et al, "Comparing Performance of Collaborative Filtering Algorithm", International Conference on Communication, Information and Computing Technology,Oct 2012,page(s):1-6.

[12] "Apache Hadoop", http://hadoop.apache.org/.

[13]KalaKarun,A.Chitharanja,"A Review on Hadoop HDFS infrastructure extensions" , IEEE Conference of Information &Communication Technologies (ICT), 2013,Page(s):132-137 .

[14] Yang Lai et al, "An Efficient Data Mining Framework on Hadoop using Java Persistence API" , 10th International IEEE Conference on Computer and Information Technology (CIT), 2010 , page(s):203 – 209.

[15]Bhandarkar. et al, "MapReduce programming with Apache Hadoop", IEEE International Conference on Parallel & Distributed Processing Symposium (IPDPS)" ,2010.

[16]Nicolae et al, "BlobSeer: Bringing High Throughput under Heavy Concurrency to Hadoop Map-Reduce Applications", 24th IEEE International Conference on Parallel and Distributed Processing Symposium Atlanta, IPDPS,2010.