International Journal of Electrical and Computer Engineering (IJECE)
Vol. 5, No. 5, October 2015, pp. 1153~1157
ISSN: 2088-8708

A User- Based Recommendation with a Scalable Machine Learning Tool

Ch.Veena*, B. Vijaya Babu**
* CSE Dept, Bomma Institute of Engineering and Technology, India
** CSE Department, K L University, India

Article Info

Article history:

Received Feb 10, 2015
Revised Jun 26, 2015
Accepted Jul 12, 2015

Keyword:

Decommendations
Distributed
Machine learning
Map reduce
Similarity

ABSTRACT

Recommender Systems have proven to be valuable way for online users to recommend information items like books, videos, songs etc.colloborative filtering methods are used to make all predictions from historical data. In this paper we introduce Apache mahout which is an open source and provides a rich set of components to construct a customized recommender system from a selection of machine learning algorithms. This paper also focuses on addressing the challenges in collaborative filtering like scalability and data sparsity [1]. To deal with scalability problems, we go with a distributed frame work like hadoop. We then present a customized user based recommender system

Corresponding Author:

Ch.Veena,
CSE Dept, Bomma Institute of Engineering and Technology
India
email: cvpchoudary12@gmail.com

1. INTRODUCTION

1.1 Recommender Systems

A recommender system plays a major role in internet technology for data gathering and rating of data. The most popularly and widely used technique is collaborative filtering [1]. Even though there are four types of filtering techniques we have focused on collaborative filtering, because collaborative filtering methods are able to collect and analyze large amount of information on user's behavior, activities or preferences. And also can predict what users would like, based on their similarities to others. To overcome the challenges in collaborative filtering we present the collaborative filtering framework of the Apache Mahout [2]. library for scalable data mining and machine learning. Mahout also provides algorithm implementations to compute recommendations in batch [3]. on a Map Reduce cluster, we put our focus on the functionality it offers for developing single-machine user based recommendation systems [4].

1.2 Colloborative Filtering

Collaborative filtering systems are broadly classified into two types:

User-Based Collaborative Filtering:

User-based collaborative filtering finds the users who share the same rating patterns with the active user (the user whom the prediction is for) [5]. 2 Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user. Collaborative filtering can also be based on implicit observations of normal user behavior which is different from implicit feedback like ratings. These systems observe and matches the user preference, behavior with what all users have done (what music they have listened to, what items they have bought) and use that data to predict the user's behavior in the future, or to predict how a user might like to behave given the chance.

Item based collaborative filtering:

Builds an item-item matrix determining relationships between pairs of items. Estimate the tastes of the current user by examining the matrix and matching that user's data. Not only depending on implicit scoring or rating system which is averaged across all users ignores specific demands of a user, and is particularly poor in tasks where there is large variation in interest.

1.3 Map Reduce Frame Work

Hadoop Map Reduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner[6]. A Map Reduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. The Map Reduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

2. COMPUTATIONAL MODEL

Memory-based user rating data is used in computing similarity between users or items and also for making recommendations. Typical examples of this mechanism are (a) neighborhood based CF and (b) item-based/user-based top-N recommendations.

$$r_{u,i} = \operatorname{aggr}_{u' \in U} r_{u',i}$$

Where 'U' denotes the set of top 'N' users that are most similar to user 'u' who rated item ‘i’. Some examples of the aggregation function include:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$

$$r_{u,i} = k \sum_{u' \in U} \operatorname{simil}(u, u') r_{u',i}$$

$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} \operatorname{simil}(u, u')(r_{u',i} - \bar{r_{u'}})$$

$$k = 1 / \sum_{u' \in U} |\operatorname{simil}(u, u')|$$

Where k is a normalizing factor defined as. And $\bar{r_u}$ is the average rating of user u for all the items rated by that user.

(a) **The neighborhood-based algorithm** calculates the similarity between two users or items, multiple mechanisms such as Pearson correlation and vector cosine based similarity are used for generating a prediction for the user by taking the weighted average of all the ratings. Similarity computation between items or users is an important part of this approach [7].

The Pearson correlation similarity of two users x, y is defined as

$$\operatorname{simil}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Where I_{xy} is the set of items rated by both user x and user y.

The cosine-based approach defines the cosine-similarity between two users x and y as:

$$\mathrm{simil}(x, y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}}$$

(b) **The user based top-N recommendation algorithm** identifies the k most similar users to an active user using similarity based vector model. After the k most similar users are found, their corresponding user-item matrices are aggregated to identify the set of items to be recommended. In this work, we present Mahout's flexible collaborative filtering framework, with a broad range of algorithm implementations and provide an API of Eclipse to implement the item-based, user-based recommendations.

3. PROBLEM STATEMENT

Let A be a |U| × |I| matrix holding all known interactions between a set of users U and a set of items I.$\mathbf{a_{u\bullet}}$ represents user U with the history of the item interaction. $\mathbf{a_{u\bullet}}$ the u^{th} row of A. [8].The top-N recommendations for user U correspond to the first N items selected from a ranking r_u of all items ,based on how strongly they were preferred by. This ranking is derived from patterns found in A.

3.1 Sequential Approach for Computing User Cooccurrences

For a pair wise comparison between users, a dot product of columns of A gives the number of items, that the corresponding users have in common .first a search for other users with similar taste is to be conducted [9].

$$\boxed{r_{u = A}{}^{T}\ (A\ \mathbf{a_{u\bullet}})}$$

Algorithm: 1 the standard sequential approach for computing the
Item similarity matrix $S = A^TA$ is shown [10].

```
Foreach user do
Foreach item i interacted by the user u do
Foreach user v also interacted with the same item i do

     $s_{uv} = s_{uv+1}$
```

Counting user coocurrences in map reduce

Let us start our algorithmic frame work to be optimized with Distributed item co occurrence counting .we take a simple model with binary data i.e. (yes-1,no-0).If we want to implement in a distributed framework like mapreduce, and share the work among several nodes, the issues in common are, the algorithm requires random access to both users and items. The complexity of user based approach is quadric in the number of users, because each user is to be compared with other user. Therefore we need to parallize the algorithm1 to run parallel proportional to the number of machines in the cluster of map reduce frame work.

A standard algorithm doesn't suit with this distributed frame work because it needs a random access to the rows and columns of A in the inner loops of the algorithm1. Here comes the advantage of ***mahout*** with its inbuilt scalability of the algorithms to run on a distributed frame work like hadoop [11].

$$\mathbf{S} = \boldsymbol{A^T}\ \mathbf{A} = \sum_{i=1}^{|I|} \sum_{u=1}^{|U|} \sum_{v=1}^{|V|} \boldsymbol{A^T_{i,u} A_{u,v}}$$

We get the outer product formulation of the matrix multiplication. We partition A by columns (the items) and store it in the distributed file system [3].

Mapper: this function reads a single column of A computes the columns outer product and sends it to the reducer.

Reducer: this function simply sums up the individual counts of the mapper and consolidates the similarity **S** per invocation. Here we also make use of this distributed approach to address the data sparsity problem of collaborative filtering.basically, the interaction matrix A is usually very sparse and contains fraction of cells with non-zero elements. This limits the number of user pairs to a small fraction. So the map function which returns the intermediately outer product matrices is formulated in a way that it returns only non-zero entries.

Combiner: all the intermediately results of the mapper are combined and some consolidated data is send on the network to the reducer by the combiner function. It reduces network overhead.

Algorithm 2: counting user co-occurrences

```
Function map ($a_{u\bullet}$):
Foreach u $\epsilon a_{u\bullet}$ do
        c←sparse_vector ()
Foreach v $\epsilon a_{u\bullet}$ with v > u do
        C[v] ←1
        emit (u; c)
Function combine (u,C1,…, Cn)
        c←vector_add (C1… Cn)
        emit (u; c)
Function reduce (u, C1,…,Cn) :
        s←vector_add (C1… Cn)
        emit (u; s).
```

4. RELATED WORK

Measuring distances and similarities:

Machine Learning algorithms like K-Nearest-Neighbor, Clustering use similarity or distance measures between two data points to find the similarities [12].

Distance between categorical data points

We can calculate the ratio to determine how similar two data points are using **simple matching coefficient**: **noOfMatchAttributes / noOfAttributes.** Basically to measure the number of attributes to be changed to match each other, **Hamming distance** is used [13].

Given that most users only see a very small portion of all movies, it doesn't indicate any similarity between the users if both the users have not seen the movie i.e both values are zero. On the other hand, if both user saw the same movie (both value is one), it implies a lot of similarity between the user. Even though matching or not, if category is structured as a **Tree hierarchy**, then the distance of two category can be calculated by measuring the path length of their common parent.

Jacard similarity:

NoOfOnesBoth/noOfOnesIn A +noOfOnesInB – noOfOnesIn A and B)

Similarity between instances containing mixed types of attributes

we can calculate the similarity of each attribute (or group the attributes of the same type)i.e. if a data point contain mixed type of attributes, and then combine them together using some weighted average. **combined_similarity(x, y) = Σ_{over_k} [w_k * δ_k * similarity (x_k, y_k)] / Σ_{over_k} (δ_k) Where Σ_{over_k} (w_k) = 1**

5. METHODOLOGY

A mahout-based collaborative filtering takes users preferences from a small sub set of data and predicts the future from the past preferences [2]. Expressions of preferences are their implicit and explicit ratings of the products. E.g. purchasing a book, reading a news article, rating the product with stars etc [4].

Creating a user-based recommender API

In this approach we compute recommendation for particular users; we look for other users with a similar taste and pick the recommendations from their items. Mahout uses any text file derived from user/item matrix. Columns and rows are identified by userID, itemID and *value* denotes the strength of the interaction (e.g. the rating given to a movie).

Loading the data from texfile into the mahout interface

DataModel M = new FileDataModel (new File ("/home/user/desktop/music.csv"));

So to compute the correlation coefficient between their interactions we use Pearson correlation coefficient.

User Similarity siml = new PearsonCorrelationSimilarity (M);

Top-N recommendations and rating prediction are listed using

List<RecommendedItem>topItems =recommender. recommend (userID, 10);
float preference = recommender.estimatePreference (userID,itemID);

Evaluation

Provide a statement that what is expected, as stated in the "Introduction" chapter can ultimately result in "Results and Discussion" chapter, so there is compatibility. Moreover, it can also be added the prospect of the development of research results and application prospects of further studies into the next (based on result and discussion).

6. CONCLUSION AND FUTURE WORKS

In this paper we have presented the importance and the challenges of collaborative filtering. We rephrased the statistical methods, data mining and machine algorithms used to build a recommender system. We demonstrated how mahout is more versatile to perform user based or item based recommendations. We showed how a scalable similarity-based recommender system on a map reduce frame work overcomes the scalability challenge. We have also presented the mahout evaluator tool, and also how it measures the quality of prediction. Most of the algorithms of mahout are already implemented in a way that they are scalable across hadoop.but still there are few challenges like sparsity, cold-start, even though mahout implements few algorithms, some of the algorithms cannot be parallelized over hadoop like stochastic gradient descent and support vector machine. In our future work we focus on the above issues and intend how our algorithms solve the challenges and scalable on large parallel and distributed networks.

REFERENCES

[1] "A survey on recommender systems based on collaborative filtering techniques", *International journal of innovation in engineering and technology*, volume 2, issue 2, April 2013.
[2] *Collaborative Filtering with Apache Mahout*, Sebastian Schelter, Technische Universität Berlin, Germany, ssc@apache.org, Sean Owen, Myrrix Ltd, srowen@apache.org
[3] *Scalable Similarity-Based Neighborhood Methods with Map Reduce*, Sebastian Schelter Christoph Boden Volker Markl, Technische Universität Berlin, Germany, firstname.lastname@tu-berlin.de
[4] *Distributed Itembased Collaborative Filtering with Apache Mahout*, Sebastian Schelter, ssc@apache.org, twitter.com/sscdotopen, 7. October 2010.
[5] "*Collaborative filtering*", Wikipedia, the free encyclopedia
[6] "apache.org"
[7] *Item-based collaborative _ltering recommendation algorithms*. WWW. pp. 285-295, 2001.
[8] Y. Koren, R. Bell, and C. Volinsky. *Matrix Factorization Techniques for Recommender Systems*. Computer, 42:30–37, 2009.
[9] J. Jiang, J. Lu, G. Zhang, and G. Long. *Scaling-up item-based collaborative _ltering recommendation algorithm based on hadoop*. SERVICES, pp. 490-497, 2011.
[10] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative _ltering. Internet Computing, IEEE, 7(1): 76-80, 2003.
[11] "Inverted indexing in Big Data using HADOOP", International journal of advanced computer science and applications, volume 4, issue 11, 2013.
[12] "*Pragmatic Programming Techniques*", rickyro, August 8, 2012
[13] "Incremental learning for dynamic collaborative filtering", International journal of advanced computer science and applications", volume 6, issue 6, June 2011.

BIOGRAPHIES OF AUTHORS

Mrs. Ch. Veena, B.Tech, M.Tech, (PhD), is working as Asst.Prof in CSE department of Bomma Institute of Technology and Science, KHAMMAM. She is Pursuing PH.D in CSE department of K L University, Vaddeswaram, Guntur (Dt) Andhra Pradesh, INDIA.

Dr B. Vijaya Babu is presently working as Professor in CSE department of K L University, Vaddeswaram, and Guntur (D.t) Andhra Pradesh, INDIA. He has obtained B.Tech., (ECE) degree from JNTU College of Engineering, KAKINADA, M.Tech., (CSE) degree from JNTU College of Engineering, Anantapur and PhD degree from Andhra University, Visakhapatnam. He has published several research papers in various International journals and attended International Conferences conducted in India .