Conglomeration of Instance Filtering's k-Nearest Neighborhood and Collaborative Filtering's Item based Recommendation on Airline Dataset System using Map-Reduce and Mahout

Mrs. D.N.V.S.L.S.Indira #1, Mr. Dr. R. Kiran Kumar #2

1 Research Scholar, Dept. Of Computer Science, Krishna University, Machilipatnam,
2 HOD, Dept. Of Computer Science, Krishna University, Machilipatnam.

Abstract: With the growth of variety in every industry, Customer finds it difficult from N different options available in the market. Its the business responsibility to showcase the best suggested item depending on his/her needs, Ratings of the product, Opinions/Feedbacks of different customers. To make these recommendations very close to the customer bahaviour, we need to process huge existing data to be processed. Hence MapReduce is considered which is emerging as a parallel, distributed paradigm for processing and generating large data sets. This trend combined with the growing need to run Machine Learning (ML) algorithms on massive datasets has led to an increased interest in implementing ML algorithms on MapReduce. Hence using machine learning algorithms in conjunction with big data can bring outright value for any business transformation. This research focuses on a way of analyzing large amount of data to give better recommendations for users by ML algorithms, thereby converting e-commerce site visitors to buyers. We mainly address the challenges of building an efficient and useful recommendation system given a large dataset, and discuss different approaches on identifying like-minded neighbors by making use of similarity, nearest neighborhood, Pearson Correlation higher level ML algorithms. We report probabilities of these kind algorithms with huge amount of data on hadoop clusters

KEYWORDS: E-commerce, Recommender System, Collaborative Filtering, k-Nearest Neighborhood, Map-Reduce, Hadoop, Mahout

1. Introduction

E-commerce has expanded more facilities for consumers, factually stated as 85% current customers prefer online shopping to in-store shopping. Convenient way of buying is the major reason for this transformation. Increased use of internet as a medium of shopping provides an opportunity for building up competitive strategies in E-commerce industry. Major concern of E-commerce business relies on how well a consumer is suggested to buy a product so that a visitor of a site could convert into a consumer.

When an individual wants to make a decision about buying a product or using a service, they have access to huge amount of user ratings of the product which can be analyzed. But analyzing that kind of huge data sets manually or with existing frameworks is too tedious [2]. Hence, a method to find out features and analyzing nearest similarity to the current user to provide a better recommendation with Collaborative Filtering (CF) [1] technique is very much needed.

Collaborative Filtering techniques have achieved prevalent realization in E-commerce nowadays. The remarkable growth of the number of customers and products in recent years poses some key challenges for recommender systems in which high quality recommendations are required and more recommendations per second for millions of customers and products need to be performed. Thus, the improvement of scalability and competence of collaborative filtering (CF) algorithms become increasingly important and difficult. In this paper, we developed and implemented a scaling-up nearest neighbourhood collaborative filtering algorithm on MapReduce, by splitting the three most costly computations in the proposed algorithm into Map-Reduce phases, each of which can be independently executed on different nodes in parallel. We also proposed efficient partition strategies not only to enable the parallel computation in each Map-Reduce phase but also to maximise data locality to minimise the communication cost. Experimental results effectively showed the good performance in scalability and efficiency of the item-based CF algorithm on a Hadoop cluster. Use of mahout for Collaborative Filtering has enhanced the accuracy[4].

2. Existing Work

This section briefly talks about existing methodologies related to collaborative filtering, recommender systems and data mining. Earlier applications of collaborative filtering implemented was based on certain clustered communities with some proper feature, thereby providing recommendation to a large community than to a person. Algorithms like Bayesian networks, clustering, classification have also been applied to recommender systems.

Clustering techniques work on identifying a group with common similarity. Once the group is formed with common feature, predictions for any individual can be made by averaging the opinions of the other users in that cluster. For example, the applications related to online news publishing group their news articles using clustering.

Well known existing algorithms in clustering are Canopy clustering and K-Means clustering. Canopy clustering is a simple and fast technique used by Mahout for clustering purpose. The objects will be treated as points in a plain space. This technique is often used as an initial step in other clustering techniques such as k-means clustering. The key idea of the canopy clustering algorithm is that one can greatly reduce the number of distance computations required for clustering by first cheaply partitioning the data into overlapping subsets, and then only measuring distances among pairs of data points that belong to a common subset[6].

K-means clustering is an important clustering algorithm. The k in k-means clustering algorithm represents the number of clusters the data is to be divided into. For example, the k value specified to this algorithm is selected as 3, the algorithm is going to divide the data into 3 clusters. Along with Canopy and K-means Fuzzy K-means and LDA clustering algorithms are available in Mahout[13][16].

Currently working models just deal with small data, to classify feature. when the data is in terms of petabytes of data thats when a sample of data is considered to generate ml algorithm results.

This work mainly focusses on providing an algorithm that recommends every individual considering historical big data

3. Dataset and Preprocessing

Data consists of flight arrivals and departure details for all commercial flights within the USA from October 1987 to April 2008. Data has more than 120 million records which occupies 1.6 gigabytes when uncompressed.

This data is officially from U.S. Department of Transportations Bureau of Transportation Statistics which tracks summary of different delays. Each file has 29 parameters out of which 11 different delays are stated. Considered Delays in our implementation is Arrival Delay, Departure Delay, Taxi In, Taxi Out, Carrier Delay, Weather Delay, NAS Delay, Security Delay, Late Aircraft Delay. Each of these delays are noted as minutes in the given comma separated file.

Few more parameters that can be considered are origin, Destination, Carrier name of the flight

Data Preprocessing

Data under consideration is a public dataset provided by Department of transformation. Once the data of 1.6GB is downloaded into local filesystem it should be moved to Hadoop Distributed filesystem from local filesystem using 'put' command. This would be a data collection stage, where in data.

Pre-Processing is basically to eliminate bad records and fields in the data-set, once this step is executed out of 29 fields only fields under consideration is given as output i.e., Origin, Destination, Flight Carrier and list of Delays. For this to happen a Map-Reduce job is submitted to the Hadoop cluster. Once the job is submitted on the cluster slave nodes which had task tracker daemon running in it would run the job in parallel by making use of cluster resources there by giving out distributed processing performance

Dataset is collected from

http://stat-computing.org/dataexpo/2009/the-data.html

4. Proposed Algorithms

To build a solution for suggesting a better product to a customer by making efficient use of historical data

1) Provides Recommendation by considering collaborative filtering techniques Pearson correlation and item-item collaborative filtering. Intersected result of above two algorithms is taken as best possible recommendation.
2) k-Nearest Neighbor algorithm is run along with step above to get k nearest neighbors satisfying, algrithms stated above
3) Provides performance when scaled on large/historical datasets using map-reduce processing

The latter part of paper covers machine learning algorithms, which we present in a high level language subsequently compiled and automatically parallelized to execute in Hadoop, an open source implementation of MapReduce. We then describe individual components of Hadoop ecosystem used in order to accomplish this solution, with a clear explanation of datasets used. Finally we report experimental results and publish future scope.

4.1 ML Overview

We now give an overview of methodologies and algorithms used in Machine Learning stack

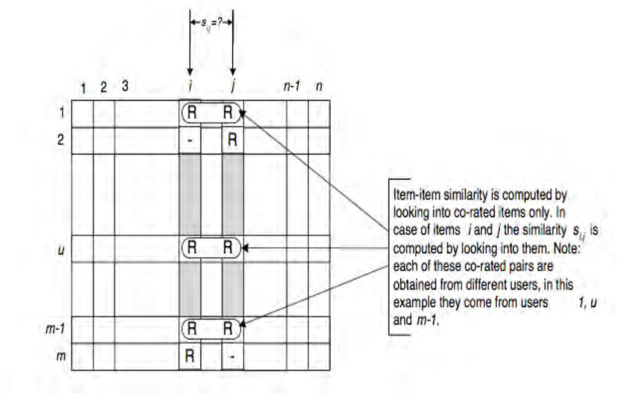
4.1.1 Collaborative Filtering:

Its a technique used by recommender systems. Its a method of making automatic predictions about interests of a user by collecting preference or taste information from many users. The underlying assumption of collaborative filtering approach is that if a person A has the same opinion as a person B on a product, A is more likely to have B's opinion on a different product x than to have a opinion on x of a person chosen randomly.

- Item-Item collaborative filtering:

A form of collaborative filtering based on a similarity between items calculated using people's rating of those items.The basic idea in similarity computation between two items *i* and *j* is to first isolate the users who have rated both of these items and then to apply a similarity computation technique to determine similarity *s(i, j)*[3][8].

Fig1. illustrates this process, here the matrix rows represent users and the column represent items. Below section talks about one of the statical methods to compute the similarity.



s_{ij}=?
1 2 3 i j n-1 n
1
2
u
m-1
m
R
-
Item-item similarity is computed by looking into co-rated items only. In case of items i and j the similarity s_{ij} is computed by looking into them. Note: each of these co-rated pairs are obtained from different users, in this example they come from users 1, u and m-1.

Fig 1: Isolation of the co-rated items and similarity computation

Item-based algorithms are two steps algorithms; in the first step, the algorithms scan the past informations of the users; the ratings they gave to items are collected during this step. From these ratings, similarities between items are built and inserted into an item-to-item matrix M. The element xi;j of the matrix M represents the similarity between the item in row i and the item in column j. Afterward, in the final step, the algorithms selects items that are most similar to the particular item a user is rating[9].

There are a number of different ways to compute the similarity between items. These are cosine vector similarity, Pearson correlation, Spearman correlation, entropy-based uncertainty measure, mean-squared difference and adjusted-cosine similarity [3][7]. Some authors suggest that Pearson correlation performs better than cosine vector similarity, while some other authors suggest that Pearson’s correlation performs better than Spearman’s correlation, entropy-based uncertainty and meansquared difference for collaborative filtering[7]. Hence here we are taking Pearson correlation based method to compute similarity. Item-to-Item collaborative filtering — focus on finding similar items, not similar customers. For each of the user’s purchased and rated items, the algorithm attempts to find similar items. It then aggregates the similar items and recommends them[5].

- Pearson Correlation similarity

In this case, similarity between two items *i* and *j* is measured by computing the *Pearson-r* correlation *corr* $_{i,\,j}$. To make the correlation computation accurate we must first isolate cases where the users rated both *i* and *j* as shown in Fig 2. Let the set of users who both rated *i* and *j* are denoted by *U* then the correlation similarity is given by

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}}.$$

Fig 2 : Formulau for Similarty Measure.

Here *Ru*,*i* denotes the rating of user *u* on item *i*, $\bar{R_i}$ is the average rating of the *i*-th item.

Its is a measure of linear correlation between two variables i and j, giving a value of -1 and +1 inclusive, where 1 is total positive correlation , 0 is no correlation and -1 is negative correlation[3][10].

4.1.2 K-Nearest neighbourhood:

Its one of the instance based learning technique which is used for classification of data in a given dataset. Its an approximation algorithm which is used to provide a solution for travelling salesman problem.Its key idea is to store all the training data and give a best match for current user vs. the user present in its training data[17].

Advantages of using this algorithm is training is way too faster, and you never loose information. On the other side using this algorithm at query time is very slow. moreover a huge data should be stored in order to find a better nearest neighbour. To address these disadvantages and provide a better solution , we opt Hadoop ecosystem for storage and processing. The next section would brief about components we mainly use in Hadoop for data processing as well as data storage.

4.2 Hadoop Ecosystem:

The Hadoop environment supports for big data processing up to terabytes to petabytes[11]. Hadoop is a free, Java based programming framework that supports the processing of large datasets in a distributed computing environment. Its key components in architecture can be broadly divided into storage and processing unit

4.2.2 HDFS - Hadoop Distributed File System

Its a distributed file system designed to run on a commodity hardware.HDFS provides a high throughput access to application data and is suitable for applications that have large datasets. All the trained datasets are stored in HDFS in form of 64MB blocks split across the cluster of 10 nodes. a cluster is a combination of name-node(Master of cluster) and data-node(Slaves of cluster)[12]. We maintain a replication factor 3 to prevent dataloss in the cluster. Data can be pulled into HDFS from

1.An external RDBMS system using sqoop

2.As a text file from the flat file available on any local disk

4.2.3 MapReduce - A processing model

Its a programming model and an associated implementation for processing and generating large data sets with a parallel, distribution algorithm in a cluster[15]. In this paper we would be using map reduce algorithm at multiple places namely for

1). Data retrieval from external storage systems or RDBMS like MySQL.

2). Data Pre-processing which includes data cleaning i.e., removal of bad records

3). Data set preparation by aggregating two or more datasets.

4.3 Mahout

Its a library for scalable machine-learning algorithms implemented on top of Apache Hadoop and MapReduce paradigm. Mahout is a production-level, open-source, system and consists of a wide range of applications that are useful for a recommender system developer: collaborative filtering algorithms, data clustering, and data classification[14]. Aim of mahout is to make use of big datasets to find meaning full pattern and turn into big information. In our study we would explore collaborative filtering technique using Pearson correlation statistical algorithm and Instance based learning technique using K-Nearest neighbourhood algorithm[18].

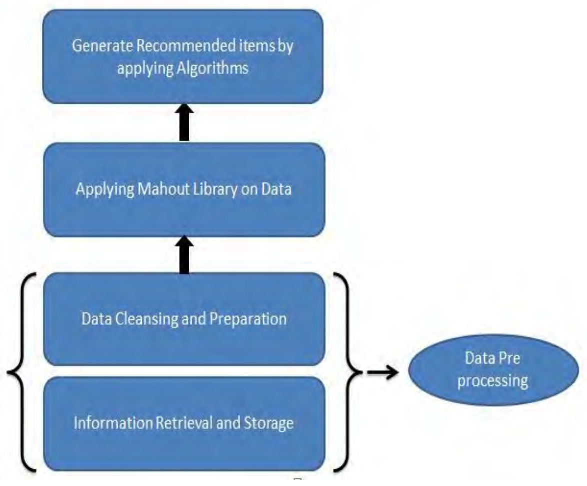

Generate Recommended items by applying Algorithms
Applying Mahout Library on Data
Data Cleansing and Preparation
Information Retrieval and Storage
Data Pre processing

Fig 3: Flow of algorithms

5. Architecture

The architecture proposed is three layered design, namely

- Data Pre-Processing using Map-Reduce algorithm
- Data Transformation using Map-Reduce algorithm
- Recommendation
    - by collaborative filtering algorithm using Mahout
    - by Instance learning algorithm using Mahout.

Current Hadoop cluster has a Master node and 5 slave nodes making it a 6 node cluster altogether, All the above steps stated would be running on the cluster for a better performance

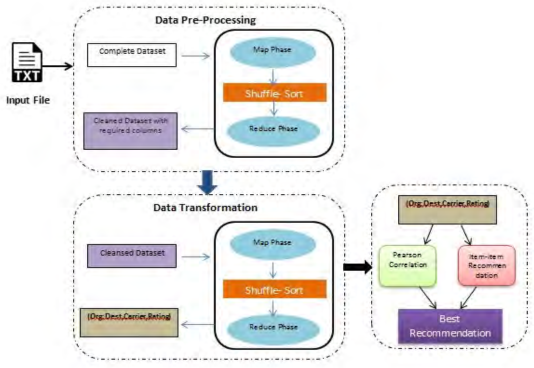


Fig 4: Proposed Architecture

5.1 Data Pre-Processing

5.1.1 Implementation of Map-Reduce Algorithm

- Map Phase:

Input of the Map Phase would be complete input file which had 29 named columns. FileInput format considered is Text Input FileFormat, which sends input (key, Value) pairs as (Line offset, content of the line record).Out of complete records only few columns required for our recommendation will be considered like origin, Destination on list of delays.

- Reduce Phase:

Data from Map phase would be passed into Reduce phase after shuffling and sorting. Reduce task output would be written to output file through context object which is part of Map-Reduce API. This output file would be used for next level of processing i.e., Data Transformation.

5.2 Data Transformation

Above layer gives an output sales which has columns that are required for providing best recommended carrier from a given origin to destination considering airline delays.

Transformation layer generates rating of the flight carriers for a given origin and destination combination.

5.2.1 Implementation of Map-Reduce Algorithm

- Map Phase:

Input of Map phase would be complete output file of the pre-processed data, which takes key as all the required columns.These columns would be grouped by a combination of origin and destination for a given carrier name.

- Reduce Phase:

Shuffle and sort phase would give output as list of delays for every unique combination of origin , Destination and Carrier.

These Delays would be considered to generate service rating of the carriers from origin to destination.

$$\bigvee_{originDestination} \sum_{i=1987}^{2008} \sum_{j=1}^{Number\ of\ trips} \frac{Sum\ of\ all\ Delays}{No.of\ trips}$$

Fig 5: Formula for Service Rating

Calculation of Service Rating:

Delay threshold is calculated as maximum delay that a flight holds from a given origin to destination. Flights whose Average delay is falling from

- [0% -10%] of threshold is rated as 5
- [11% - 20%] of threshold is rated as 4
- [21% - 40%] of threshold is rated as 3
- [41% - 70%] of threshold is rated as 2
- [71% - 100%] of threshold is rated as 1

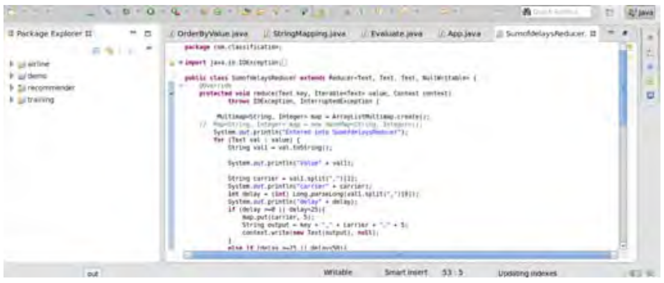
Where Average delay is calculated as sum of all delays from an origin to destination divided by number of trips between the same origin and destination combination.

Output of this phase would be (Origin:Destination,Carrier Name, Rating). This file would be used as the trained data to providing recommendations for a given combination of origin and destination airports.

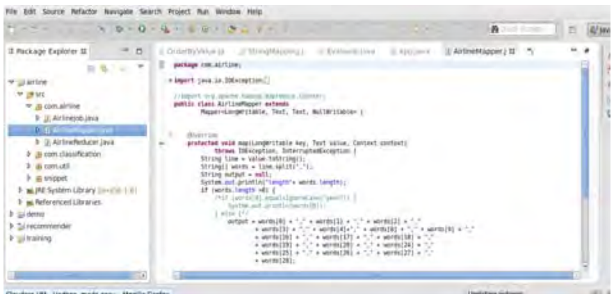
5.3 Recommendation Algorithm

Pearson Correlation similarity and Item-Item similarity recommendation algorithms would be implemented on the output of above layer. Intersected results of both the algorithms would be considered as best recommendation, rest of the similarities will be showcased as better recommendations

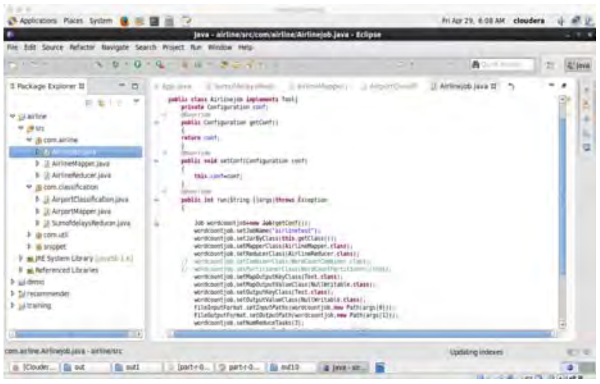
6. Screen Shots



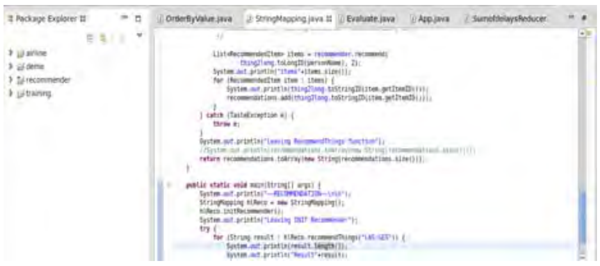
```
package com.classification;

import java.io.IOException;

public class SumofdelaysReducer extends Reducer<Text, Text, Text, NullWritable> {
    @Override
    protected void reduce(Text key, Iterable<Text> value, Context context)
            throws IOException, InterruptedException {

        Multimap<String, Integer> map = ArrayListMultimap.create();
    //  Map<String, Integer> map = new HashMap<String, Integer>();
        System.out.println("Entered into SumofdelaysReducer");
        for (Text val : value) {
            String val1 = val.toString();

            System.out.println("Value" + val1);

            String carrier = val1.split(",")[1];
            System.out.println("carrier" + carrier);
            int delay = (int) Long.parseLong(val1.split(",")[0]);
            System.out.println("delay" + delay);
            if (delay ==0 || delay<25){
                map.put(carrier, 5);
                String output = key + "," + carrier + "," + 5;
                context.write(new Text(output), null);
            }
            else if (delay >=25 || delay<50){
```



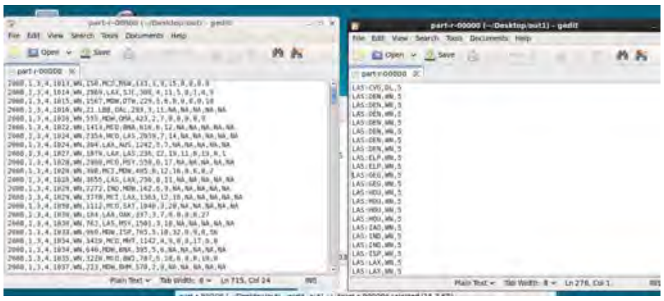
```
package com.airline;

import java.io.IOException;

//import org.apache.hadoop.mapreduce.Counter;
public class AirlineMapper extends
        Mapper<LongWritable, Text, Text, NullWritable> {

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
        String line = value.toString();
        String[] words = line.split(",");
        String output = null;
        System.out.println("length"+ words.length);
        if (words.length >0) {
            /*if (words[0].equalsIgnoreCase("year")) {
                System.out.println(words[0]);
            } else {*/
                output = words[0] + "," + words[1] + "," + words[2] + ","
                        + words[3] + "," + words[4]+"," + words[8] + "," + words[9] + ","
                        + words[16] + "," + words[17] + "," + words[18] + ","
                        + words[19] + "," + words[20] + "," + words[24] + ","
                        + words[25] + "," + words[26] + "," + words[27] + ","
                        + words[28];
```



```
public class Airlinejob implements Tool{
    private Configuration conf;
    @Override
    public Configuration getConf()
    {
    return conf;
    }
    @Override
    public void setConf(Configuration conf)
    {
        this.conf=conf;
    }
    @Override
    public int run(String []args)throws Exception
    {

        Job wordcountjob=new Job(getConf());
        wordcountjob.setJobName("airlinetest");
        wordcountjob.setJarByClass(this.getClass());
        wordcountjob.setMapperClass(AirlineMapper.class);
        wordcountjob.setReducerClass(AirlineReducer.class);
    //  wordcountjob.setCombinerClass(WordCountCombiner.class);
    //  wordcountjob.setPartitionerClass(WordCountPartitioner.class);
        wordcountjob.setMapOutputKeyClass(Text.class);
        wordcountjob.setMapOutputValueClass(NullWritable.class);
        wordcountjob.setOutputKeyClass(Text.class);
        wordcountjob.setOutputValueClass(NullWritable.class);
        FileInputFormat.setInputPaths(wordcountjob,new Path(args[0]));
        FileOutputFormat.setOutputPath(wordcountjob,new Path(args[1]));
        wordcountjob.setNumReduceTasks(3);
```



```
            List<RecommendedItem> items = recommender.recommend(
                    thing2long.toLongID(personName), 2);
            System.out.println("items"+items.size());
            for (RecommendedItem item : items) {
                System.out.println(thing2long.toStringID(item.getItemID()));
                recommendations.add(thing2long.toStringID(item.getItemID()));
            }
        } catch (TasteException e) {
            throw e;
        }
        System.out.println("Leaving RecommendThings function");
        //System.out.println(recommendations.toArray(new String[recommendations.size()]));
        return recommendations.toArray(new String[recommendations.size()]);
    }

    public static void main(String[] args) {
        System.out.println("--RECOMMENDATION--\n\n");
        StringMapping hlReco = new StringMapping();
        hlReco.initRecommender();
        System.out.println("Leaving INIT Recommender");
        try {
            for (String result : hlReco.recommendThings("LAS:GEG")) {
                System.out.println(result.length());
                System.out.println("Result"+result);
```


part-r-00000 (~/Desktop/out) - gedit
File Edit View Search Tools Documents Help
Open Save
part-r-00000
Plain Text Tab Width: 8 Ln 715, Col 24 INS
part-r-00000 (~/Desktop/out1) - gedit
File Edit View Search Tools Documents Help
Open Save
part-r-00000
Plain Text Tab Width: 8 Ln 276, Col 1 INS

```
2008,1,3,4,1013,WN,158,MCI,MSW,133,1,9,15,0,0,0,0
2008,1,3,4,1014,WN,2909,LAX,SJC,308,4,11,5,0,1,4,9
2008,1,3,4,1015,WN,1567,MDW,DTW,229,1,6,0,0,0,0,18
2008,1,3,4,1016,WN,21,LBB,DAL,293,3,11,NA,NA,NA,NA,NA
2008,1,3,4,1020,WN,555,MDW,OMA,423,2,7,0,0,0,0,7
2008,1,3,4,1022,WN,1414,MCO,BNA,616,6,12,NA,NA,NA,NA,NA
2008,1,3,4,1024,WN,2354,MCO,LAS,2039,7,14,NA,NA,NA,NA,NA
2008,1,3,4,1024,WN,204,LAX,AUS,1242,5,7,NA,NA,NA,NA,NA
2008,1,3,4,1027,WN,1874,LAX,LAS,236,22,19,11,0,15,0,1
2008,1,3,4,1028,WN,2800,MCO,MSY,550,6,17,NA,NA,NA,NA,NA
2008,1,3,4,1029,WN,360,MCI,MDW,405,8,12,16,0,6,0,7
2008,1,3,4,1029,WN,3655,LAS,LAX,236,8,11,NA,NA,NA,NA,NA
2008,1,3,4,1029,WN,2272,IND,MDW,162,6,9,NA,NA,NA,NA,NA
2008,1,3,4,1029,WN,3770,MCI,LAX,1363,12,18,NA,NA,NA,NA,NA
2008,1,3,4,1030,WN,1112,MCO,SAT,1040,3,20,NA,NA,NA,NA,NA
2008,1,3,4,1030,WN,184,LAX,OAK,337,3,7,0,0,0,0,27
2008,1,3,4,1030,WN,762,LAS,MSY,1501,3,10,NA,NA,NA,NA,NA
2008,1,3,4,1033,WN,960,MDW,ISP,765,3,10,32,0,0,0,56
2008,1,3,4,1034,WN,3419,MCO,MHT,1142,4,9,0,0,17,0,0
2008,1,3,4,1034,WN,646,MDW,BNA,395,5,6,NA,NA,NA,NA,NA
2008,1,3,4,1035,WN,3220,MCO,BWI,787,5,10,0,0,0,19,0
2008,1,3,4,1037,WN,233,MDW,BHM,570,7,9,NA,NA,NA,NA,NA
```

```
LAS:CYS,DL,5
LAS:DEN,WN,5
LAS:DEN,WN,5
LAS:DEN,WN,5
LAS:DEN,WN,5
LAS:DEN,WN,5
LAS:DEN,WN,5
LAS:DEN,WN,5
LAS:ELP,WN,5
LAS:ELP,WN,5
LAS:GEG,WN,5
LAS:GEG,WN,5
LAS:HOU,WN,5
LAS:HOU,WN,5
LAS:HOU,WN,5
LAS:HOU,WN,5
LAS:IAD,WN,5
LAS:IND,WN,5
LAS:IND,WN,5
LAS:ISP,WN,5
LAS:LAX,WN,5
LAS:LAX,WN,5
```

7. Experimental Results

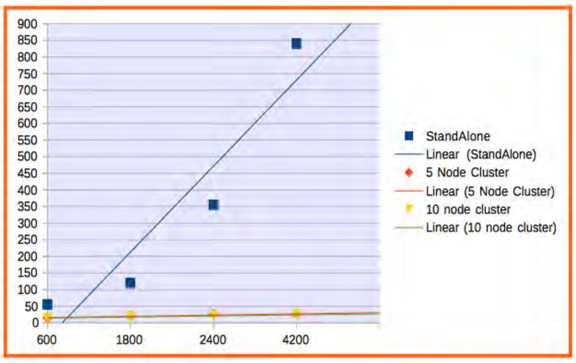


Chart 1 :Line chart describes the performance of exixting architecture which is named as standalone machine vs. 5 node cluster(Hadoop) vs. 10 node cluster(Hadoop)

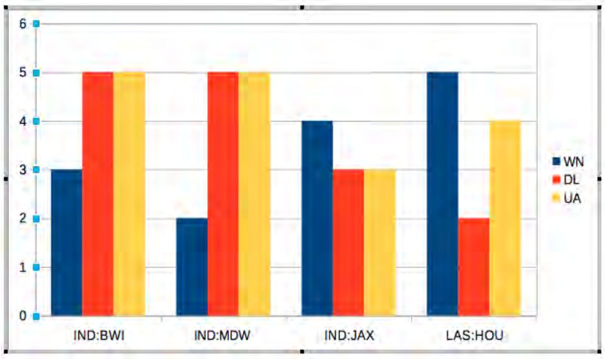


Chart 2: Bar chart represents the proximity of carrier UA for different origin :Destination combinations. Best recommendation for UA replacement as per algorithm is DL

Table 1: Experimental Results: Better Recommendations

Origin: Destination	**Recommendations (Pearson Correllation)**	**Recommendations (Slope One)**	**Better Recommendation**
IND:BWI	**WN,UA**	**WN,DL**	**WN**
IND:MDW	**WN,DL**	**WN,DL**	**WN,DL**
IND:JAX	**DL,UA**	**DL,WN**	**DL**

8. Conclusion and Future Work

The feature proposed above not only talks about better recommendation model but also performance in terms of how fast a processing engine can give accurate output . As its well known concept that when the trained data is huge, the probability achieved in recommending best product/item would be more. In order to process this we used Hadoop Architecture in our paper. Going beyond this the same recommendation can be achieved using Spark which is termed as 100x faster than Hadoop Map-Reduce in memory. Spark has it own key component called MLlib which provides machine learning libraries and can be a best replacement to Apache Mahout

9.REFERENCES

[1] Samundeeswary K, Vallidevi Krishnamurthy, “Design of A Recommendation System Using Hybrid Collaborative Filtering”, I J C T A, 8(5), 2015, pp. 2043-2049, © International Science Press.

[2] D. Mali, M. Abhyankar, P. Bhavarthi, K. Gaidhar, M.Bangare, “Sentiment Analysis of Product Reviews for Ecommerce Recommendation”, Proceedings of 44th IRF International Conference, 29th November 2015, Pune, India, ISBN: 978-93-85832-59-8.

[3] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, “Item-based Collaborative Filtering Recommendation Algorithms”, Appears in WWW10, May 1-5, 2001, Hong Kong.

[4] Swati Pandey, T. Senthil Kumar, “Costomization of Recommendation System Using Collaborative Filtering Algorithm on Cloud using Mahout”, IJRET: International Journal of Research in Engineering and Technology, eISSN: 2319-1163 | pISSN: 2321-7308.

[5] Greg Linden, Brent Smith, and Jeremy York, “Amazon.com Recommendations Item-to-Item Collaborative Filtering” , JANUARY - FEBRUARY 2003 Published by the IEEE Computer Society, 1089-78, 01/03/2003.

[6] Andrew McCallum, Kamal Nigam, Lyle H. Ungar, “Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching”, SIGKDD ’02 Edmonton, Alberta, Canada Copyright 2002 ACM 1-58113-567-02/0007.

[7] Manos Papagelis, Dimitris Plexousakis, “Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents” ,Engineering Applications of Artificial Intelligence 18,(2005) 781–789.

[8] Prem Melville and Vikas Sindhwani, “Recommender Systems”, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598.

[9] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, William Nzoukou, “A Survey Paper on Recommender Systems” , arXiv:1006.5278v4 [cs.IR] 24 Dec 2010.

[10] Zibin Zheng, Hao Ma, Michael R. Lyu, and Irwin King, “WSRec: A Collaborative Filtering Based Web Service Recommender System”, 2009 IEEE International Conference on Web Services.

[11] Hemlata Kardas, Sulochana Sonkamble, “ Review on Analysis of User Behavior Based on Prediction Algorithm”, International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064, Volume 4 Issue 1, January 2015.

[12] http://en.wikipedia.org/wiki/ Apache_Hadoop.

[13] Arantxa Duque Barrachina and Aisling O’Driscoll, “A big data methodology for categorising technical support requests using Hadoop and Mahout” , Journal of Big Data 2014.

[14] Carlos E. Seminario and David C. Wilson, “Case Study Evaluation of Mahout as a Recommender Platform”, Workshop on Recommendation Utility Evaluation: Beyond RMSE (RUE 2012), held in conjunction with ACM RecSys 2012. September 9, 2012, Dublin, Ireland.

[15] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, ACM, 51(1):107-113, 2008.

[16] Prajesh P. Anchalia, Anjan K. Koundinya, Srinath N. K., "MapReduce Design of K-Means Clustering Algorithm," icisa, pp.1-5, 2013 International Conference on Information Science and Applications (ICISA), 2013.

[17] Prajesh P Anchalia and Kaushik Roy, “The k-Nearest Neighbor Algorithm Using MapReduce Paradigm”, 2014 IEEE Fifth International Conference on Intelligent Systems, Modelling and Simulation, 2166-0662/14.

[18] Sebastian Schelter and Sean Owen, “Collaborative Filtering with Apache Mahout” , RecSysChallenge’12, September 13, 2012, Dublin, Ireland.