# Big Data : Final Report

Hantao ZHAO
Xinyuan YU
Yifan SU

December 10, 2014

## 1 Abstract

Traditional marketing research for restaurants is based on the questionnaires and customers feedback. The output of it is usually limited by the scale of the feedback. To find possible potential opportunities of restaurants of a certain city, Yelp, a restaurants rating website, has provided a dataset that contains restaurants features and ratings. The project uses Pearson correlation and Collaborative Filtering to find the preference of a region. In order to enlarge the scalability of the project, big data technologies, such as Hadoop MapReduce and Amazon Web Services have been utilized during the data mining process. By finding the most favorite features of the popular restaurants, the implementation is able to recommend a certain kind of restaurants which will have better performance in the region. The final result has given a satisfactory increase of restaurant's rating.

## 2 Introduction

### 2.1 Problem Description

Yelp collects an enormous amount of raw data on a particular business - user reviews, ratings, location, categories, features and more. Businesses organize their own listings while users rate the business from 1-5 stars. While businesses are able to see their current ratings and the features, there is no information relating to forecasts about their business - Will their business increase? What features are needed if they want to become more popular in the nearest future? With this knowledge, a business owner may determine that something is useless with their current business model, or that a new restaurant feature they added is a really big hit.

### 2.2 Traditional Techniques and Our Method

In consideration of this, it would be valuable to businesses to find a restaurant type which may have the best business potential and catch the most customers' need from current dataset. User-based and model-based collaborative filtering are the most successful technology for building recommender systems to date and is extensively used in many commercial recommender systems where squared Euclidean distance, cosine distance, and kernel functions are commonly used to measure similarity. However, our problem is not a standard recommend problems.

In this work, we describe our efforts to process Yelp raw information and select best features using our model. We put forward to methods for feature recommendation based on Pearson correlation coefficient and compare features' similarity between cities to find difference which might improve overall customers' rating. Both tell us the features a business needed that matches what the Yelp market expects. Then we test our system on a larger dataset based on Apache Hadoop software library using MapReduce programming models.

# 3 Implementation

## 3.1 Data Modal

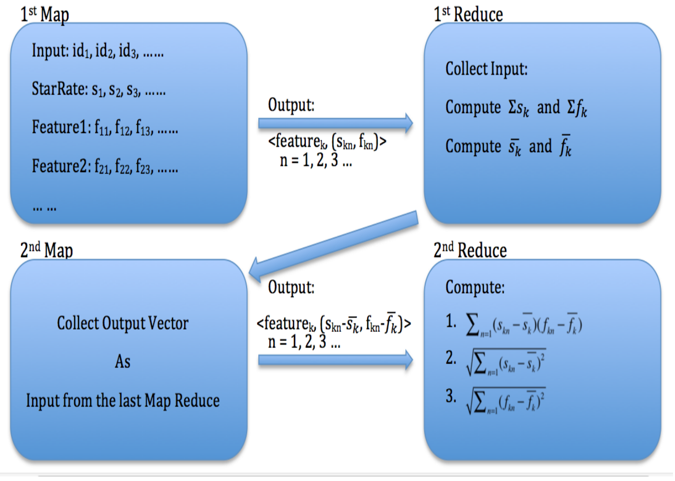
The data model has been applied for this milestone's application has been simplified into matrix to adapt to the map reduce process. The original restaurants information has been stored as a whole data set and it is not convenient to do the corresponding calculation. Thus the data set has been processed and store into the unit of cities. The restaurants' features are thus in the form of the matrix and each line stands for the features of one restaurants.

The data has then been stored in a file of matrix and uploaded to S3 to be analyzed.

We recommend restaurant features based on the Pearson product-moment correlation coefficient (commonly represented by the Greek letter $\rho$), which is a measure of the linear correlation (dependence) between two variables feature X and star rate Y. The formula for $\rho$ is:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{1}$$

MapReduce Model for Pearson Coefficient Calculation is as follows.


1st Map
Input: $id_1, id_2, id_3$, ……
StarRate: $s_1, s_2, s_3$, ……
Feature1: $f_{11}, f_{12}, f_{13}$, ……
Feature2: $f_{21}, f_{22}, f_{23}$, ……
… …
Output:
<$feature_k$, ($s_{kn}$, $f_{kn}$)>
n = 1, 2, 3 …
1st Reduce
Collect Input:
Compute $\Sigma s_k$ and $\Sigma f_k$
Compute $\bar{s_k}$ and $\bar{f_k}$
2nd Map
Collect Output Vector
As
Input from the last Map Reduce
Output:
<$feature_k$, ($s_{kn}$-$\bar{s_k}$, $f_{kn}$-$\bar{f_k}$)>
n = 1, 2, 3 …
2nd Reduce
Compute:
1. $\sum_{n=1}(s_{kn}-\bar{s_k})(f_{kn}-\bar{f_k})$
2. $\sqrt{\sum_{n=1}(s_{kn}-\bar{s_k})^2}$
3. $\sqrt{\sum_{n=1}(f_{kn}-\bar{f_k})^2}$

We partition the job of recommendation between cities into two steps.

Step1. Calculate Pearson Similarity between each pair of cities

The data model is just as before

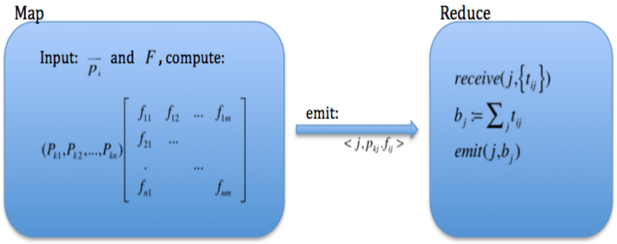
Step2. Recommendation according to similarities

We achieve the recommendation vector by the sum of weighted feature vectors of other cities.

We weight the feature vector using the Pearson Similarity from Step1.

$$City_k * = (P_{k1}, P_{k2}, ..., P_{kn}) \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1m} \\ f_{21} & \cdots & & \\ . & & \cdots & \\ f_{n1} & & & f_{nm} \end{bmatrix}$$

Data Model for recommendation between cities is as follows.

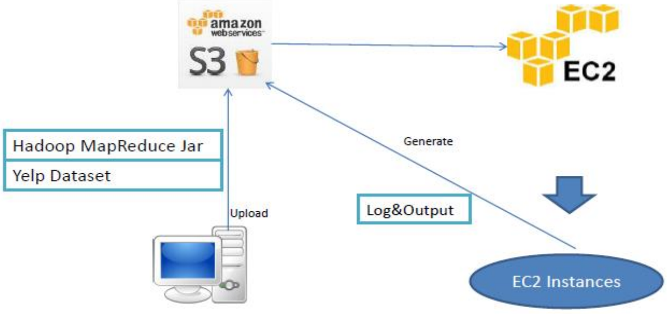

Map
Input: $\overline{P_k}$ and $F$, compute:
$(P_{k1},P_{k2},...,P_{kn}) \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1m} \\ f_{21} & \cdots & & \\ . & & \cdots & \\ f_{n1} & & & f_{nm} \end{bmatrix}$
emit:
$< j, p_{kj} \cdot f_{ij} >$
Reduce
$receive(j,\{t_{ij}\})$
$b_j := \sum_j t_{ij}$
$emit(j,b_j)$

## 3.2 System Architecture

We build our system based on Apache Hadoop software library, which is a framework that allow for the distributed processing of large data sets across clusters of computers using simple programming models (MapReduce for example). Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer.

MapReduce is a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. A MapReduce program comprises a Map() procedure that performs filtering and sorting and a Reduce() procedure that performs a summary operation.

Within the first part of our project, the team applied an algorithm to calculate the suitable restaurants features in the data set of Yelp. From the first part we got the result that in all the cities that we applied our algorithm, our recommended restaurants have a better performance than the average performance of all the other restaurants in the same city. Thus we have the reason to believe that this algorithm can be expanded to a wider implementation. In milestone 3 we choose the scalable way to experiment our algorithm. Based on our data structure, which is in the form of JSON, we used Hadoop plus Amazon Web Services. Hadoop can naturally adapt Java file and to calculate the result we need to upload the Jar file to Amazon Web Services. To simplifier the calculation we have adapt our dataset to a matrix of the restaurants' features, in order to upload the data set to Amazon. The matrix contains the corresponded features' true and false value, and the stars rating of the restaurant. The following figures shows the architecture of the data model.



## 4 Contribution

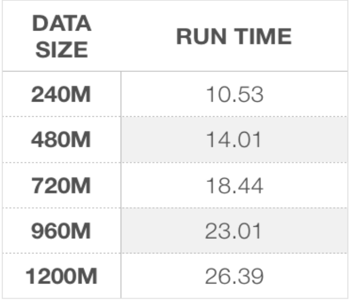
Nowadays, businessmen find most prospective restaurant type generally through market research, artificial statistical methods, which requires a significant investment of human, material and financial resources. How to make full use of the large amount of online business dataset (such as Yelp challenge) remains to be a promising research trend.

Our system provide a feasible feature selection method that not only extract efficient features based on the overall Pearson correlation (tested to be more precise in our case than other baseline similarity function) between feature and star rate vector, but also recommend between different cities. We consider the former as vertical comparison, the latter as horizontal comparison. On the other hand, not just limited to small datasets, we expand our system to a larger data set using MapReduce framework to accelerate computation speed, which achieves a much better extensibility.

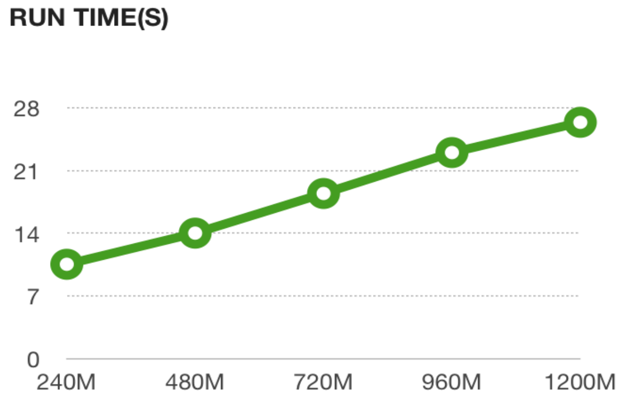
## 5 Performance Measurements and Results

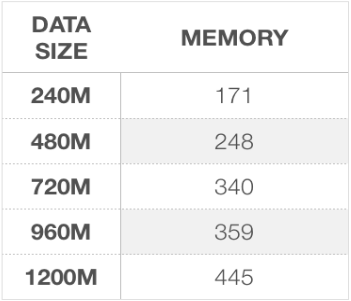
We select five datasets with different data size of 240M, 480M, 720M, 960M and 1200M. The experiment environment is Hadoop which we installed on local desktop with 8G memory and processor of 2GHz Intel Core i7.

We compared run time and used memory according to different dataset. The result is showed in the following graph.

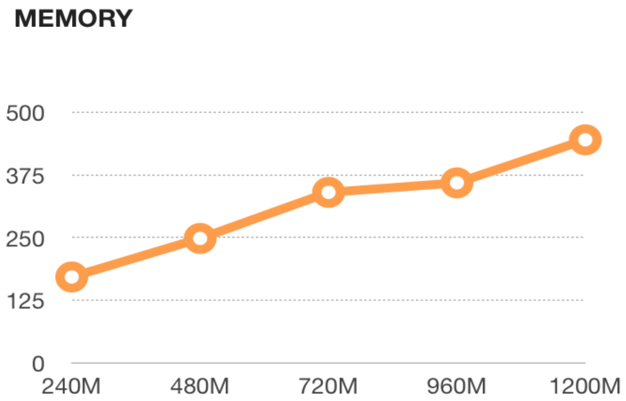


| DATA SIZE | RUN TIME |
|---|---|
| 240M | 10.53 |
| 480M | 14.01 |
| 720M | 18.44 |
| 960M | 23.01 |
| 1200M | 26.39 |



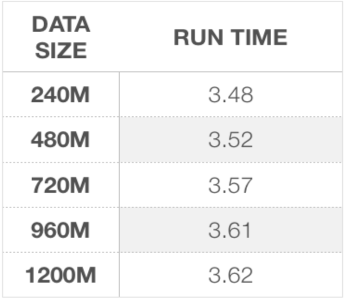


| DATA SIZE | MEMORY |
|---|---|
| 240M | 171 |
| 480M | 248 |
| 720M | 340 |
| 960M | 359 |
| 1200M | 445 |

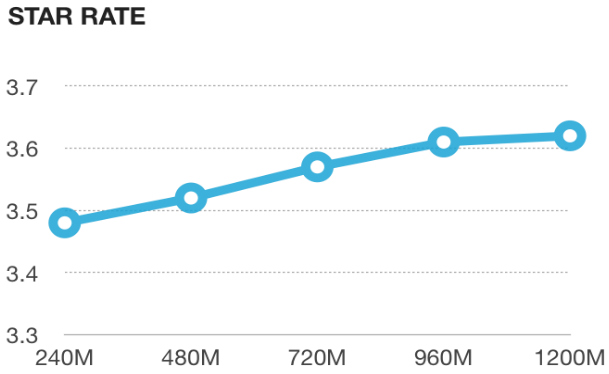


We can figure out that the run time and memory used are both linear function of data size.

We use the average star rating as the evaluation of our system. We calculate the average star rating of the recommendation restaurants in experiment of datasets with different size. The result is described in the following graph.



| DATA SIZE | RUN TIME |
|---|---|
| 240M | 3.48 |
| 480M | 3.52 |
| 720M | 3.57 |
| 960M | 3.61 |
| 1200M | 3.62 |



We can see that the average star rate increases linearly from 240M to 960M, and the increasing rate becomes smooth after 960M.

# 6 Conclusion

## 6.1 Trade-off analysis

The tradeoff from this project mainly focuses on the data sets size, which is the fact that if we reduce the data set then the average performance of the restaurants would decrease too. We have made the following diagram to see the change of the result when we apply different dataset.

To compute the result we used the Hadoop MapReduce to add another feature to our calculation. We have successfully improved the result to the final 3.62 yet the calculation time and memory uses have almost doubled. We believe that if in the future we are facing a big data problem we will still choose to apply the MapReduce algorithm in order to improve our performance. So the tradeoff is apparently worth it and quality of the result will increase if we invest more computational effort and data set.

From on Milestone 2 on we have introduced the usefulness feature of the review. As can be seen previously the increase of the dataset has led to a better performance. The connection between the increase of the dataset size and the increase of the performance is almost linear. It is also stabilized when 80% of the data set is used. Thus we can conclude that the size of the dataset has been wisely chosen.

## 6.2 Difficulties

As to the difficulties we have encountered, we have experienced the following issues during our implementation:

1. Data source. When we try to retrieve more information from the Yelp database we find out that the API they have provided to the developers has limitation for each query. That means we can only request limited number of restaurants each time so that it is difficult to organize a huge database. Fortunately we have found a dataset of Yelp that has been provided for a dataset for a competition and it contains the necessary input for our calculation.

2. Evaluation. Based on the major goal of this application we set the result of the calculation should be an ideal restaurants which will achieve a great star rating in the experimented city. Yet since we cannot really open a restaurant and test its performance on Yelp, it is hard to find a way to evaluate our result. The limitation is that we can only evaluate our result based on the existing restaurants, yet there is usually not perfect matching of the suggested restaurant since multiple features exist.

3. Amazon Web Services. In milestone 3 we begin to implement the application on the Amazon Web Services. The first biggest issue is to adapt to the way to use the cloud calculation. Although there are tutorial and instructions a lot of efforts and time has been put on it. Second of all since our previous program doesn't take long to run, it is annoying to wait for the long time launching of the Amazon cluster. Within the milestone 3 our implementing time is thus limited by the AWS launching time. Last but not least we accidentally used the Java API of the AWS and it turns out that we have launched some expensive services. We are still in touch with AWS customer services to deal with this problem. Since this part concerns real money we should have been more careful about it.

### 6.3 Potential improvements

There are plenty of things that we have learned. Firstly the wide use of the ideal big data. When we tried to find a suitable subject for the project we have found out that almost all the data mining or it companies have more or less introduced the idea of big data. The huge amount of already existing projects have led us to dig more and learn more about how big data has influenced people's daily life. Also the idea of cloud calculation has been known for long time but we have never had the chance to use it by ourselves. AWS is definitely a great platform to learn and implement data mining projects. In the future when issues of calculation have occurred, we would for sure consider Amazon Web Services. Yet since this time we have only processed small amount of data and we have already generate quite a lot of bill, the cost of cloud calculation has also been a factor during the procedure of development. It is good to learn the advance technology and its cost beneath. What we would do differently would be the way we started the milestone 3. We could start by using the local Hadoop and run the example locally. But instead we began by using AWS and it cost us a lot of time and efforts. It is logical to do some prototype locally and then apply more advanced technology. If we have done that we would have time to polish our milestone 3 result and maybe we wouldn't have had the bill issue.

The algorithm of our program is something we would be differently. In this part we have consider a city as a whole object yet the geographical information of the city has not yet been introduced. Different zones of the same city may have different needs and the locations of the restaurants are also very vital, especially if we want to introduce whole new restaurants.

### 6.4 Continuation of the project

The possible extension of this project is beyond the measure. First of all we would definitely expand the dataset. With data mining technologies we could collect more restaurants information within one city and thus improve the precision of the result.

Another continuation that would be surely taken into consideration is the geographical information. So far we calculate the city as one unit and in reality the city is usually divided by zones. Naturally the location of the restaurant is another vital feature that could be included. Yelp has provided the restaurants address and names. With the API of Google Map we believe that we could find a way to recommend a perfect location for the new restaurants.