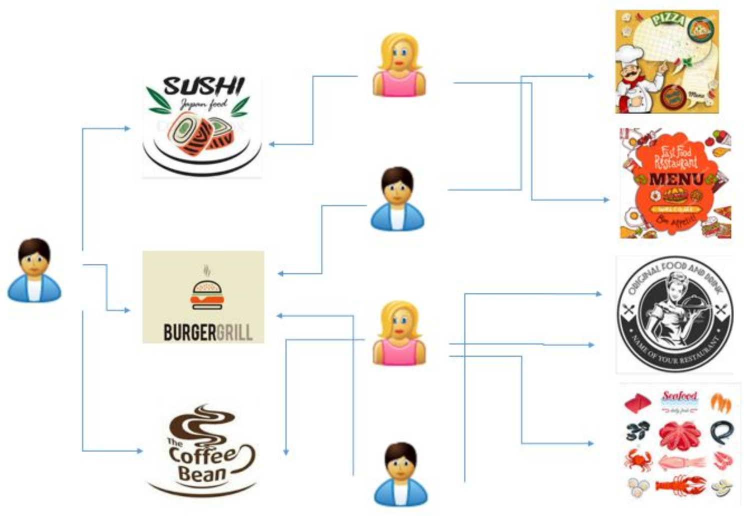
Big Data Project Report : Recommendation System using ALS Collaborative Filtering

Sr. No	Name	Net Id	Role, Contributions
1.	Madhuri Abnave	**MXA146230**	Design, Development, Testing Owned: **Preprocessing**
2.	Ameya Kadam	**ASK140730**	Design, Development, Testing Owned: **Validation**
3.	Ravali Nallamasu	**RXN152730**	Design, Development, Testing Owned: **Analysis**
4.	Kanchan Waikar	**KPW150030**	Design, Development, Testing Owned: **Integration**

Problem Definition:

Typically, users visit restaurants and based on their experience, they rate the same. Each user has specific taste, specific choices and based on the overall experience, a user arrives to the decision of whether or not he or she likes the restaurant. Only if we had the ability to predict the restaurant that user would like to visit, we would be able to provide him/her the accurate information about the restaurant he or she should visit. This would definitely be a good strategy for implementing user focused advertising which is proven to be a much better alternative to mass advertisement approach.

Since population has grown and people love to eat out, Yelp actually has such a comprehensive list of restaurants that we can actually find an approximate match for each user who has identical taste and experience perception as that of another. We use **Yelp review dataset**, Collaborative filtering algorithm coupled with powerful Big Data technology like **Apache spark+HDFS** and try to form a regression line that predicts Restaurant review ratings for a user pretty accurately. We create a restaurant recommendation system that recommends a restaurant to person X based on the ratings provided by the users who share the similar tastes. We achieve this using Scala MLLIb library.



Data Set Used

We use Yelp Dataset of 200K reviews for doing analysis and training our collaborative filtering model. Yelp Dataset used is from following link.

Link : https://www.yelp.com/dataset_challenge

The format of input data we are using is as follows :-

review.json

```
{
   'type': 'review',
   'business_id': (encrypted business id),
   'user_id': (encrypted user id),
   'stars': (star rating, rounded to half-stars),
   'text': (review text),
   'date': (date, formatted like '2012-03-14'),
   'votes': {(vote type): (count)},
}
```

business.json

```
{
   'type': 'business',
   'business_id': (encrypted business id),
   'name': (business name),
   'neighborhoods': [(hood names)],
   'full_address': (localized address),
 ... other attributes
}
```

The Backbone Algorithm

Everybody has got an opinion bias. Everybody has opinion about restaurants they visit and how they find the same. Traditional route of word-of-mouth does not work because the opinion bias of people does not always match because of limited number of people one talks to or gets opinion of. This opinion bias is what needs to be matched. Two people with identical opinion bias are more likely to try restaurants that other one likes. This is where ALS and Matrix Factorization model comes into picture. They discover latent features and data and form the Lower rank Factor Matrices using gradient descent approach.

The collaborative Filtering ALS algorithm is used for predicting Ratings for given user. Let us call the user for which we are doing the prediction of the restaurant, **active user**. The prediction is made by training the model using database of user ratings from a sample that includes ratings of different users for various restaurants.

The database consists of tuple <BusinessId (j), UserId (i), Rating> where rating is corresponding to the rating given by user i on businessId j. If I;i is the set of businessId on which user i has given rating, then we can define the mean rating for user i as :

$$\overline{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

The predicted rating for the active user for businessId j is **p;a,j** which is nothing but the weighted sum of ratings of other users.

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i)$$

where n is the number of users in the collaborative filtering database that have nonzero weights. The weights w(i; a) reflects identicalness between each user i and the active user. This is nothing but similarity between opinions of two different users on the restaurant in question.K is a normalizing factor such that the absolute values of the weights sum to unity.

The correlation between users a and i is where the summations over j are over the businessId for which both users a and i have recorded ratings.

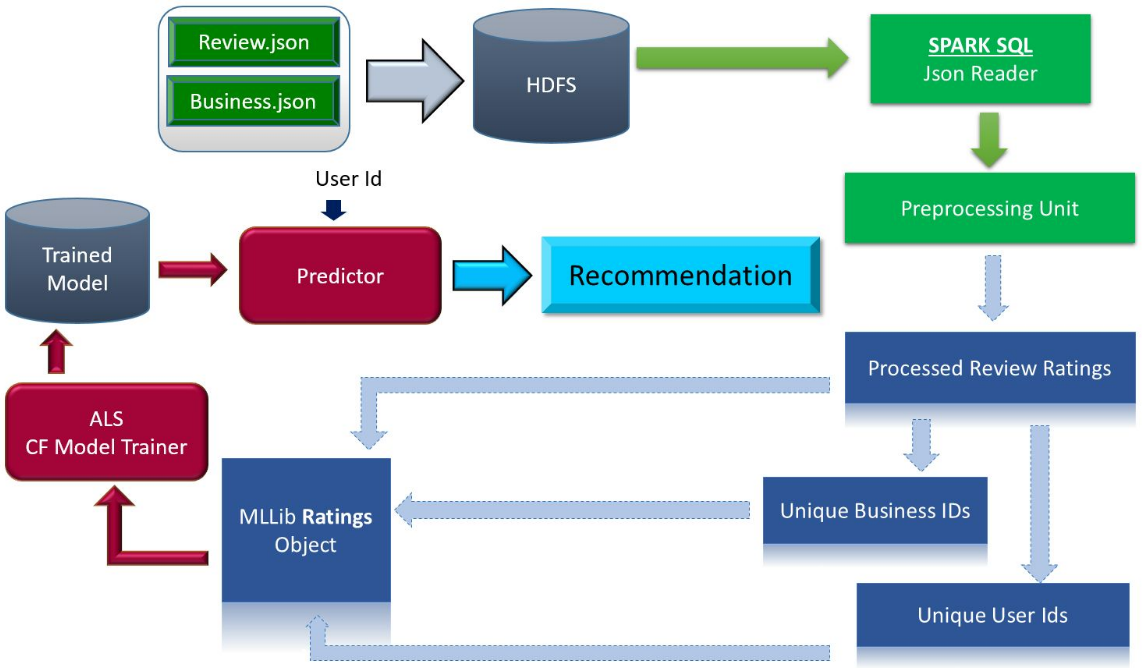
$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}} \qquad (2)$$

Once ratings are predicted for multiple restaurants then we will recommend the user top 5 or 10 restaurants based on the predicted rating. The number of restaurants to be predicted is a variable value and can be changed as per the requirement very easily.

Big Data Strategy: Why Spark? Why HDFS? Why MLLIB?

<u>System Design Diagram</u>

Please find System Design below. Following diagram gives a complete overview of how data flows and which all tiers are used for processing data.



Architecture Tiers:

- Hadoop:

Distributed File System and processing File System. HDFS is used to host the input files on which the processing of data to retrieve the desired results are carried out.
Since HDFS has storage redundancy and can scale infinitely, Choosing HDFS for storing files is a very optimal Design choice.

- Apache Spark:

Low Latency Computing. Spark has been intensively used to evaluate the ratings from the dataset based on the collaborative filtering algorithm. Spark Has Resilient Distributed datasets which can be used for doing parallel computing over multiple nodes in memory itself. This advantage of Spark definitely makes it an optimal choice for executing an algorithm that is memory intensive.

- MLLib:

This is an inbuilt Library of Machine Learning Algorithms. It has well tested implementations for several Machine learning algorithms that it becomes an optimal choice for speedy development of an experiment or even a real life project that heavily relies on machine learning.

- Collaborative Filtering(CF)

Spark - MLLib Implementation of ALS is a collaborative filtering algorithm that uses user information, business information along with rating (selection, purchase information could be also used) and target user history to recommend an item that target user does not have ratings for. Fundamental assumption behind this approach is that other users preference over the restaurants could be used recommending to the user who did not visit before.

Algorithm

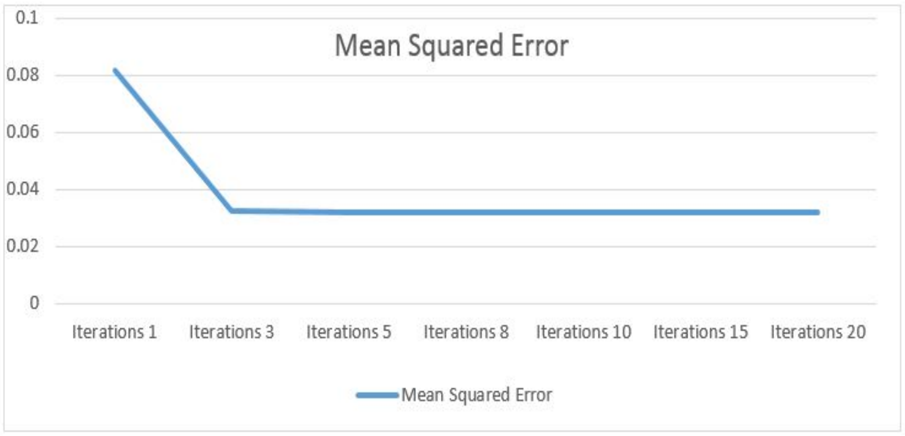
1. Create SQL Context for loading Json
2. Read Json from hdfs using sql context json reader
3. Create tuple of UserId, BusinessId and Ratings
4. Remove null business ids and user ids and select distinct entries into two separate lists

5. Create a org.apache.spark.mllib.recommendation.Rating instance using index of user id as well as business id, Since ALS accepts only integers
6. Train model using ALS, give numIterations as 10 and rank as 10 too.
7. Do prediction on entire trained data
8. Merge original dataset with predicted values - this gives variable contains <user_id,Business_id,<Actual,predicted>>
9. Calculate MSE for model
10. Select the active User and call recommend Products on model
11. Analyze the restaurant recommendations.

Our algorithm converts the restaurant review data into ALS-MLLIb specific Ratings data and uses the predictions generated by the model in order to recommend the restaurant.
Helper functions used: Define a helper function converts Any format of data to String

Analysis of Results:

The analysis of the algorithm is mainly based on the the root mean squared error and the predictions are done on the data is compared with the actual business reviews the user has given. Root mean square plays a very important role when it comes to testing the regression line. It depicts the variance between the predicted value and actual value. In our case, the difference is between the predicted rating Vs actual rating. The following diagram represents the mean squared error for a particular user by varying the number of iterations.
The trend in the following graph shows the system shows accurate predictions as the iterations increase.



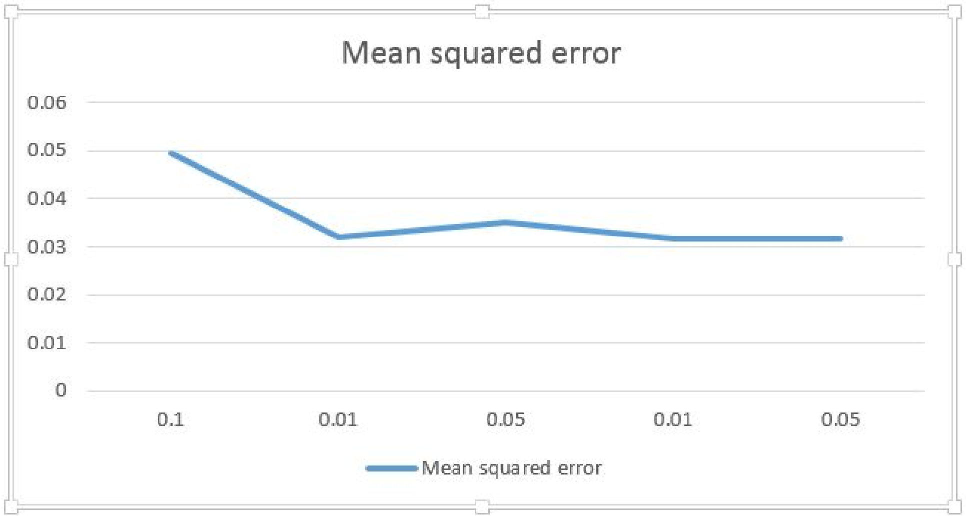
MSE vs. Number of Iterations

The following diagram shows the changes in the accuracy of the algorithm if the parameter lambda is varied.

The parameter lambda specifies the regularization parameter in the algorithm. It controls the tradeoff between our two goals.

- The first is the fit the training set well.
- The second would be to keep the parameters small

As we can see in the diagram below the algorithm converges as the lambda decreases.



Lambda vs MSE

Detailed Analysis & Validation

Let us take an example in order to see how well our code predicts. When we train a CF model for small size data(1mb file containing approximately 1k reviews), we get the MSE of ~0.032, which is very accurate as it depicts that regression line formed by ALS collaborative filtering algorithm is very accurate. We also can use this data in order to validate whether predictions made by ALS is accurate or not.

Please find stats for a user given below.

Input: User 248

User 248's ratings

Sr. No.	Business Id	Rating	Address
1.	45	2.0	200 E Main St Carnegie Carnegie, PA 15106
2.	41	4.0	171 E Bridge St Homestead Homestead, PA 15120
3.	31	4.0	166 E Bridge St Homestead Homestead, PA 15120
4.	40	4.0	1 Ravine St Dravosburg, PA 15034
5.	71	4.0	205 East Waterfront Drive Homestead Homestead, PA 15120

We can clearly see that User 248 likes visiting restaurants in Pennsylvania around 15120 Zip Code. When we run predictions for this user, we get following in our output.

Sr. No.	Business Id	Rating	Address
1.	12	3.5	2100 Washington Pike Carnegie, PA 15106
2.	70	4.5	300 Beechwood Ave Carnegie Carnegie, PA 15106
3.	10	3	1000 Sandcastle Dr Hays Homestead, PA 15120

4.	58	4.5	1073 Washington Ave Carnegie, PA 15106
5.	21	3.5	1011 Washington Ave Carnegie, PA 15106
6.	1	3.5	520 North Bell Avenue Carnegie Carnegie, PA 15106
7.	23	4.5	101 E 7th Ave Homestead Homestead, PA 15120
8.	44	2.5	214 E Main St Carnegie Carnegie, PA 15106

We clearly see that all the businesses that algorithm has returned as top 8 predictions belong to the same zip code. We also notice that It also returns restaurants whose rating is around average rating given by the user 248. The above data provided is based on 1k reviews, when we increase the size of the data to 200k, we get more accurate predictions based on user's section since the matrix is calculated for each of the user.

When we run the algorithm on actual Review data of size 200 mb containing 200k reviews for several businesses by several users, we get MSE of 0.17, which is still not bad. This means that even on big data, ALS works pretty well and assuming number of computations that are required for generating Matrix Factorization model for such huge dataset, Only framework like HDFS+Spark+Scala is an optimal choice. Since the algorithm requires data to be present in memory, achieving the same using MapReduce would be really complicated and time consuming too. Spark has proven performance over frameworks like MapReduce since data is present in memory which is much faster than hard disk access.

Statistics for Big Data File

time taken for 200 mb sample file

start time=> 10:53:14

end time =>10:55:40

total time taken : 2 min 26 seconds

Note: The above performance was observed when we started spark-shell with 4 nodes.

Conclusion:

Our hypothesis of the project was to predict the ratings of a particular user and recommend him the top restaurants on the basis of the ratings provided by people of similar tastes. The results we obtained support our hypothesis. The traditional methods like RDBMS could not help store and analyse the data efficiently to its full value. The choice of distributed technologies like Hadoop and Scala which provide a relatively low cost and extremely scalable platform for recommendations. The technologies helped in designing a distributed memory which is fault tolerant and efficient working with huge amounts of data from YELP. Since, Spark with MLLib offers a great library of established Machine Learning algorithms reducing development efforts, we implemented the ALS algorithm in training the dataset and Matrix Factorization methods to predict the results.
Future enhancements could include to provide information about products or explanations for the recommendations which is not included in this project.

References:

- http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf - *Paper on "Collaborative Filtering Recommender Systems" by Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan.*
- http://www.slideshare.net/CasertaConcepts/analytics-week-recommendations-on-spark - Information on Building a Recommendation System using Spark
- http://cs229.stanford.edu/proj2013/SawantPai-YelpFoodRecommendationSystem.pdf - Paper by Sumedh Sawant and Gina Pai *"Food Recommender System".*
- http://spark.apache.org/ "Apache Spark" website.
- Y.F. Hu, Y. Koren, and C. Volinsky, *"Collaborative Filtering for Implicit Feedback Datasets,"* Proc. IEEE Int'l Conf. Data Mining (ICDM 08), IEEE CS Press, 2008, pp. 263-272.
- http://spark.apache.org/docs/latest/mllib-collaborative-filtering.html