User Recommender System Based on Yelp

Qi Gao
qxg150130@utdallas.edu

Yongxuan Tan
ymt180001@utdallas.edu

Xiaoran Guo
xxg180001@utdallas.edu

Introduction

Nowadays, many people pick a restaurant on Yelp, which is a technology company that provide all kinds of information of business shops, such as photos, reviews, locations and so on.

The basic idea of this project is that we are going to deeply analyze and research on the data that Yelp provide. Yelp has over 35 millions of average monthly mobile app unique users and 69 millions of average monthly mobile web unique visitors. The users are generating tons of data every moment and Yelp has got amount 184 millions reviews since inception. We are going to use those data to extract features from it, and split it into a number of user groups which can be used to build a recommender system. We will also analyze the relationships between users to see if we can help business determine the most influential users on the Yelp platform.

Related Work

Yelp has made their datasets available for a few years prior to the start of our project. They also created their own competitions on Yelp Dataset Challenge as well as Kaggle. We focus our research around previous machine learning courses that used Yelp datasets as source data because of a few reasons. First, one of the goals of our project is to test as many technologies as we learned this semester in this Big Data Management course. Many of the reports written by machine learning students share this same goal. Second, almost all previous analysis was programmed in non-distributed environments using Python. Since the Yelp datasets has grown significantly, we want to test our performance using Apache Spark. Third, these reports not only provided us with a baseline for testing our results, but also included a future work section that allow us to build our report on top of their results. In the end, we selected the following reports:

1. S. Sawant, G. Pai, Yelp Food Recommendation System [2].
2. C. Guthrie, Using Machine Learning to Enhance a Collaborative Filtering Recommendation System for Yelp [3].

Dataset Description

Yelp provides 6 json files, which have millions of lines and include user information, business information, reviews and so on. We sorted out the data and present dataset details as follows [1]:

Business Dataset

Colunm	address	attributes	Business ID	Categories	City	Hours	Is_open
Data Type	String	Struct	String	String	String	String	Long
Column	Latitude	Longitude	Name	Postop_code	Review_count	Stars	state
Data Type	Double	Double	String	string	long	double	string

User Dataset

Column	Average_stars	Compliment_cool	Compliment_cute	Compliment_funny	Compliment_hot	Compliment_list
Datatype	Double	Long	Long	Long	Long	Long
Long	Compliment_more	Compliment_note	Compliment_Photos	Compliment_plain	Compliment_profile	Compliment_writer
Datatype	Long	Long	Long	Long	Long	Long
Column	Cool	Elite	Fans	Friends	Funny	Name
Datatype	Long	String	Long	String	Long	Stgring
Column	Review_count	Useful	User_id	Yelping_since		
Data type	Long	Long	String	String		

Review Dataset

Column	Business ID	Cool	Date	Funny	Review_id	Stars
Data Type	String	Long	String	Long	String	Double
Column	Text	Userful	User ID			
Data Type	String	Long	string			

Datesets Detail

	Business Dataset	Review Dataset	User Dataset
No. Features	14	9	22
No. Instances	174000	5200000	6800000

Environment

This project required the use of Amazon Web Service (AWS). We first used Intellij to create a sbt project with one Scala object for each technology we implemented. After packaging our the Intellij project into a jar file, we uploaded this jar file along with all source data onto Amazon Simple Storage Service (S3). Then, we created an Amazon EMR cluster with the same Spark version we used in our sbt project. Finally, we added steps to our EMR cluster to execute programs that generated output to S3.

Pre-processing Techniques

For the SVD analysis, we are implementing is the Singular Value Decomposition (SVD) library from Spark. The library was mentioned in Sawant and Pai's report [2], where they used the Python version of this library. SVD operates similarly to Principal Component Analysis, another dimensionality reduction tool. Its Spark API states that it is an RDD based library, so we believe it should be considerably faster than most systems. We first extracted user and business identification (ID), and review rating from the review dataset. Next, we replaced those IDs with unique numeric ID using *zipWithIndex* method in RDD. Then, we constructed an IndexedRowMatrix with users as rows and business as columns.

Another technique we implemented was the GraphX library avalaible on Spark. GraphX provides traditional graph computation while extends Spark RDD making it an ideal library to process large datasets efficiently. The detail API is available on Apache Spark's GraphX Programming Guide. The user dataset provided us valuable information to construct a property graph. We first extracted unique users to be our vertices. These vertices included user ID, name, and number of fans. The numerical user ID was created by the *monotonically_increasing_id* method. Next, we extracted the friends list from each user. By using split and explode method available to dataframe, we turned this friends list into one friendship per row.

Methods

Following the footstep of Sawant and Pai's work [2], we attempted to replicate their SVD result using Spark. We populated each user's rating of a business by the value from source data. Contrary to Sawant and Pai, we initialized missing ratings to be 0 instead of the average of all ratings for that business [2]. We also arbitrarily chose to keep the top 25 singular values and corresponding singular vectors. After training our SVD, we reconstructed our predicted values in the form of a matrix with the same dimension as our original matrix. Finally, we used 10% of our input data to calculate our Root Mean Square Error (RMSE). The RMSE is calculated as: $\sqrt{(p-a)^2}$, where p is the predicted value and a is the actual value.

In order to construct a property graph, we converted our friendships table into directed edges with the number of fans as value. These edges represent how many fans a user is contributing to this friendship. The idea was that if a user became friends with someone who had a large fan base, the ranking of this user increased. Finally, we used the following native methods to calculate our ranking in table 1: *outDegrees*, *pageRank* with tolerance value 0.1, and *triangleCount*.

In addition, we used collaborative filtering technique to build up our recommender system. And we used Spark library, MLlib, to create the recommender system. Generally speaking, there are three common algorithms, which are Demographic-based Recommendation, Content-based Recommendation, and Collaborative Filtering-based Recommendation. For collaborative filtering, it is divided to three sub-categories, which are user-based, item-based, and model-based. Collaborative filtering of Spark library provides user-based and model-based algorithms. For user-based collaborative filtering, use statistical techniques to find neighbors who have the same preferences as the target users, and then generate recommendations to the target users based on the preferences of the target users' neighbors. The basic principle is to use the similarity of user access behavior to recommend resources that users may be interested in. For item-based, it would be based on the evaluation of the item or information by all users, then find out the similarity between items, and then the user would receive recommendations of similar items are recommended to the user based on the historical preference information of the user.

Spark MLlib library provides collaborative filtering library to implement recommender system. There are several things need to pay attention on. First of all, programmers need to know how to use the parameters, such as maxIter, rank, numBlocks. In addition, we need to decide to use explicit or implicit feedback. The difference between those two is that explicit feedback only use rating as label to train and evaluate, but implicit feedback may use other columns as label like number of clicks. Moreover, we decided to drop those test data that were not trained in training set.

Results and Analysis

The SVD calculation took more than 14 hours to execute using an EMR cluster. Our RMSE was at 3.9, which was higher than what we expected since Sawant and Pai stated their RMSE for the SVD algorithm as at 1.5.

The graph analysis portion of our program took 26 minutes to execute. Our result is presented in Table 1. To our surprise, users with highest number of fans do not mean they also have highest number of friends. This finding suggested that users with the most followers don't necessary become friends with their followers. We also found that users with highest potential to grown fans are not the ones with most friends. Both of this finding made us to believe that there are many closed communities among Yelp users who have the most friends. Due to time required to compute strongly connected components (SCC) on such big data, we could not verify this hypothesis at this moment. Furthermore, we were puzzled by users with highest triangle count but were not those with the most friends. This suggested that there were no SCC among those with most friends.

	Most Fans (#)	**Most Friends (#)**	**Most Potential to Grown Fans (PageRank)**	**Largest Communities (TriangleCount)**
1.	Mike (9538)	Nate (4919)	Sarah (599)	Michael (583614)
2.	Katie (2964)	Arla (4603)	Todd (339)	Yasaman (402824)
3.	Cherylynn (2434)	Jay (4597)	Nikki (324)	M (386124)
4.	Ruggy (2383)	Katelyn (4437)	Stacey (303)	Ale (377310)
5.	Daniel (2132)	Robin (4222)	Andrea (281)	Richard (352599)
6.	Candice (2123)	Travis (4211)	Chrissy (263)	Roland (336494)
7.	Peter (2086)	Katherine (4134)	Dallas (260)	Ronald (331377)
8.	Jessica (1964)	Travis (4067)	Ash (245)	Ray (323748)
9.	Jeremy (1944)	Rodney (3943)	Jeff (243)	Lindsay (317564)
10.	Brittany (1878)	Lindsay (3923)	Lauren (237)	Marilyn (312140)

Table 1. Top 10 ranking of users using GraphX

The dataset from review json file is divided to training and testing dataset for their own purposes. After training the dataset by collaborative filtering API from Spark library, we transformed model on test dataset and evaluated on it. After several training and running, the best result of evaluation is that root-mean-square error is equal to 3.68. In addition, after user input business ID, our program will produce recommendation table as follows:

bussiness_id_index	recommendations
8085	[[45796, 4.997649...]
17148	[[646, 4.996711],...]
4025	[[5976, 4.997857]...]
32098	[[646, 4.996711],...]
2578	[[1961, 4.711012]...]
4283	[[10053, 4.997807...]

This table produces a list of recommendations of user who may like those specific businesses that user input. And the other way is that user input its user ID and finally get relative business recommendations from our program.

Conclusion

The primary reason our SVD algorithm performed worse than our compared report was that they used the average rating of a business to populate user ratings for all businesses with no ratings. Even though our approach eliminated any bias assumption that users would have rated businesses according to the average, another problem arise. Our matrix became too sparse. Guthrie also encounter the same problem in his report [3]. The reason was many users had only few reviews total, thus most of the 174 thousand businesses had a 0 rating. This was furthered evidence by the fact that the total number of businesses went from over 6 thousand in 2013 to 174 thousand today. The second reason is that we can have tested more options to select top singular values.

We had hoped to find the most influential users on Yelp using graph applications. This finding would help Yelp businesses to narrow their promotion to those users therefore receiving highest return on investment. However, our results currently do not provide enough confidences to suggest users with the most fans were the most influential people.

In the future, we can improve our SVD algorithm by first trying a wide range of options for the number of top singular values. Second, we could eliminate users with limit reviews since our dataset is very large. We can find the strongly connected components of our graph to further explain our findings so we can make a stronger case to businesses.

Parameter tuning is very important for collaborative filtering of Spark library, because delicate difference might produce huge difference on results. In order to improve accuracy, parameter tuning

methods are supposed to be used. Moreover, K fold cross validation is a good method to enhance accuracy. Therefore, whatever library and algorithm you use, evaluation method and parameter tuning methods are necessary to use for enhancing model performance.

Contribution

Qi: utilized collaborative filtering API from Spark library to implement a recommender system.

Yongxuan: implemented SVD algorithm and analyzed RMSE result; implemented Property Graph and interpreted ranking results.

Xiaoran: read DataFrames from json files to input datasets, and filter rows of Dataframe according to some arguments like Yelp, such as category, city and state.

Reference

1. Yelp Dataset Documentation. https://www.yelp.com/dataset/documentation/main
2. S. Sawant, G. Pai, Yelp Food Recommendation System, Stanford University http://cs229.stanford.edu/proj2013/SawantPai-YelpFoodRecommendationSystem.pdf
3. C. Guthrie, Using Machine Learning to Enhance a Collaborative Filtering Recommendation System for Yelp, Stanford University http://cs229.stanford.edu/proj2013/Guthrie-YelpRecommendationSystem.pdf
4. "Collaborative Filtering." *Collaborative Filtering - Spark 2.4.3 Documentation*, spark.apache.org/docs/latest/ml-collaborative-filtering.html.
5. Apache. "Apache/Spark." *GitHub*, github.com/apache/spark/blob/master/examples/src/main/scala/org/apache/spark/examples/ml/ALSExample.scala.