CMPT 732 – BIG DATA PROGRAMMING I

Project – Recommendation Engine and Visualization with Yelp data

Contents

1. Introduction

Yelp organizes a data challenge with its datasets, which contain information on Yelp's restaurants, users, ratings, and check-in(s) [1]. Such information holds a potential to help Yelp engage with its users in a more personalized level. Imagine a scenario when a customer has just arrived at a new city or simply plans to eat out, Yelp can list out personalized recommended restaurants and let the customer know how busy those restaurants are in each hour.

With 2.5GBs of data containing information of 600K+ users, 70K+ restaurants and 2M+ ratings across 10 selected states, we would face challenges in processing, analyzing and visualizing such large-volume of data, especially when the recommender system needs to scale up to handle all locations that Yelp operates in. In this project, we first implement an end-to-end Spark solution to perform data processing and analysis to determine the best 20 restaurants in each state for each user. Then we develop an interactive web-based solution for a user to select a specific state and view the recommended restaurants, together with their details such as address, ratings, estimated number of customers at different time.

2. Methodology

Our end-to-end solution is illustrated in Figure 2.1:

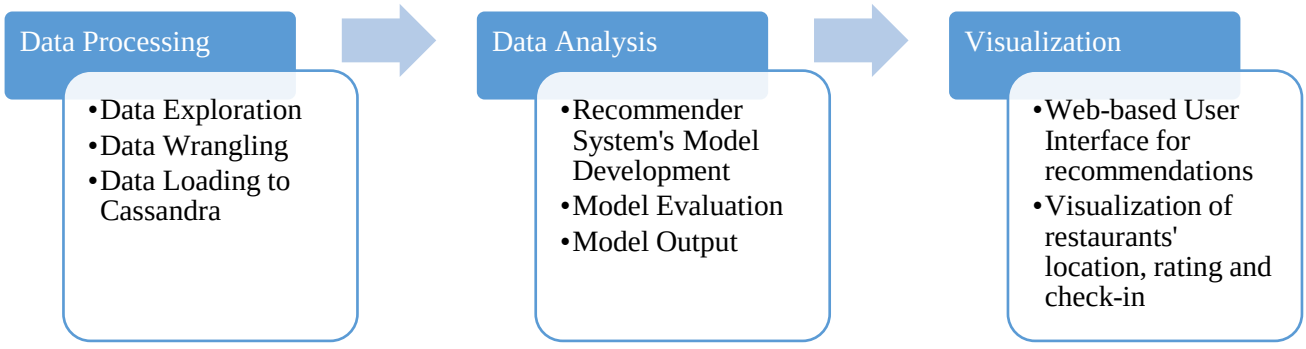


Figure 2.1 – End-to-End Solution for recommender system

2.1. Data Processing

The 2.5GBs source datasets are provided in JSON format. The below Yelp datasets are relevant to the projects:

- Users: user-specific information such as user ID, first name.
- Businesses: restaurant-specific information such as restaurant ID, restaurant name, address, state, latitude, longitude, review counts, overall rating, number of users checking in each hour on each weekday.
- Review: rating that users give to restaurants

Yelp provided a sample code to convert their JSON datasets to tabular format [2]. We modify the code to handle data cleaning & transformation and develop a Spark job to load the data into four **Cassandra** tables:

- Yelp_user: contains information from Users JSON dataset
- Yelp_review: contains information from Review JSON dataset
- Yelp_business: contains part of information from Businesses JSON dataset, excluding check-in data

- Yelp_business_checkin: contains normalized data for businesses' check-in information

In the project, the Spark job needs to be initiated through command line. In practice, such job can be scheduled to periodically load JSON source data into Cassandra tables. We choose Cassandra for our data storage since its parallelism can handle large volumes and ingest more data in less time.

2.2. Data Analysis

Spark MLlib is a scalable machine learning library consisting of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, as well as underlying optimization primitives. We used MLLib – Collaborative Filtering to predict user ratings for restaurants. Collaborative filtering is commonly used for the recommender systems. These techniques aim to fill in the missing entries of a user-item association matrix. To predict these missing entries, we have used the alternating least squares (ALS) algorithm of MLLib.

The dataset was split into 6:2:2 ratio for Training, Validation, and Test. The training and validation datasets were used to evaluate and select the best model with minimum mean squared error. Once the best parameters were obtained, the model was trained over the entire dataset of user provided ratings for restaurants. Then, the ratings of the restaurants for selected users were predicted and saved in Cassandra Database. Based on the predicted ratings, top 20 restaurants per state for the selected users were filtered out and the respective check-in information for the restaurants was obtained. The restaurant recommendations for the users along with the check-in information was saved in Cassandra Database. We chose Cassandra to store the recommendation outputs to make use of its capability to handle large amount of data, parallel retrieval for faster access and replication level to ensure data availability for web service to pick up.

2.3. User Interface and Visualization

ExpressJs in **Nodejs** is used for implementing webserver which acts as a middleware between the user interface and Cassandra database. Nodejs uses javascript and hence it will be easy to interact with user interface written in javascript. Also, a library is available in nodejs to interact with Cassandra database which is simple to use. Nodejs use non-blocking, event-driven i/o to remain lightweight and efficient in the face of data intensive applications and hence it will be fast when dealing with large data.

SFU Gateway is used as a webserver [3] and it is connected to Cassandra database in the cluster. When a user logs in with the userid and password, webserver the webserver will push the login information page to the user and perform input validation by querying yelp_user table (which stores the userid and password of each user). If the input provided by the user is not correct, an appropriate error message is sent back to the user. If the input is correct, then the webserver queries recommendations and recommendations_checkin tables to retrieve top 20 recommended restaurants and its check-in information for that user in each states. Webserver then renders userInfo.ejs, which contains the retrieved data in javascript's JSON formatted variables and Javascript functions for visualization. A challenge with such visualization on a web interface is to ensure the site is responsive and

scalable even with large amount of data. We opt for **Google Map API** to handle spatial data visualization and **D3.js** to create different charts since these two technologies can scale well with large volumes of data

3. Problems & Challenges

3.1. Data Processing

Our main problems in data processing include data quality, data format and data volume.

- Data Quality: While performing initial data exploration, we realize that even though the Yelp source datasets are supposed to contain only restaurants in 11 selected states, they include restaurants from other or unknown states. We exclude these restaurants while loading in the data. Besides, check-in data (number of customers at different time) lacks information on local time zone, so we need to adjust the check-in time to match local time zone.
- Data Format: Some text data such as address contains special characters, which need to be removed before we can load in Cassandra. Besides, since the check-in information in Business JSON dataset is not normalized, we need to normalize it before loading in Cassandra. Such data transformation not only eliminates unnecessary duplication but also facilitates our data extraction and visualization later.
- Data Volume: At first, we tried loading the data in Cassandra using normal Python application, but it takes too much time. Hence, we develop a Spark application to make use of its parallelism capability for data loading.

3.2. Data Analysis

The ALS.train function expects the input RDD to contain user_id(s) and product_id(s) in integer format, but in the Yelp dataset the user_id(s) and restaurant_id(s) were alphanumeric. We had to use dictionary to map these alphanumeric values to integer values and then convert them back to alphanumeric values after predicting the ratings of the restaurants for the users.

3.3. User Interface and Visualization

To display to recommendation result to users, we developed a web-based solution to retrieve data from two Cassandra tables and send the data to user interface, i.e. HTML response in this case. Each query will be executed using an API and its result is captured in its callback. From each callback, one more callback function is called to send the data to user interface. As Nodejs is asynchronous, as soon as it receives data from any one of the tables, it will send it to user interface. In this case, user interface will receive data from only one of these tables. To overcome this problem, we introduced a lock mechanism. A variable lock is declared and initialized to 2. This lock will be decremented by 1 each time a query is executed and when it is equal to 0, then the callback to send the data to user interface is executed.

On web user interface, a user can interactively select a state that he or she is interested in and visualize recommended restaurants' location, ratings, estimated number of customers at different time. D3.js time-series line chart requires the data to be sorted by time. Since Cassandra's support for sorting is limited, we had to sort the data using javascript so that D3.js can display the line chart properly.

4. Results

4.1. Model Evaluation

The recommendation model was evaluated by measuring the Mean Squared Error of rating prediction over combination of various parameters which are as follows:

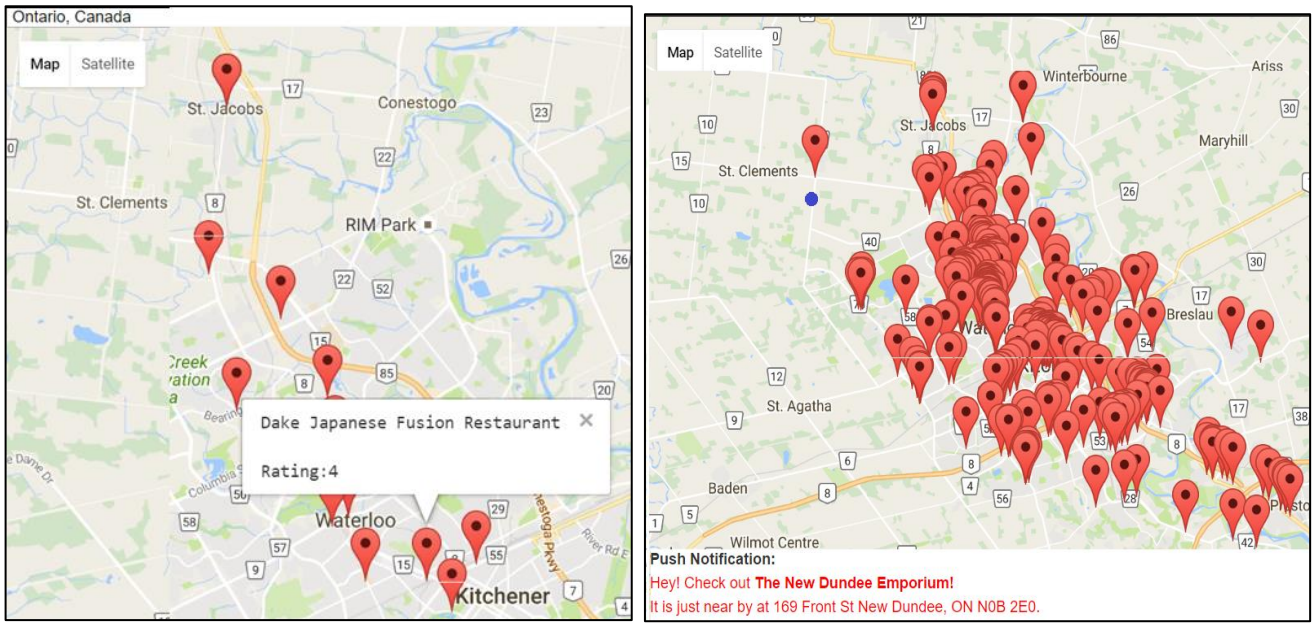
Parameter	List of values
Iterations	10, 15, 20
Regularization	1.0, 0.1, 0.01
Rank	12, 16, 20

The best model was achieved with the below parameters:

Iteration	Regularization	Rank	Train Error	Test Error
20	20	0.1	1.5568	1.5569

4.2. Visualization

After logging in, an user can use top right selection box to choose a specific state that he/she is interested in. The best 20 recommended restaurants will be visually displayed on Google Map as illustrated in Figure 4.2.1 (a).



(a) (b)

Figure 4.2.1 – (a) Spatial Visualization of Recommendation -(b) Simulation of location -based push notification

Figure 4.2.1 (b) shows a simulation in which a customer (representing by the blue dot) is nearby one of his/her potentially favorite restaurant and Yelp can send a push notification to let the customer know about it.

Ratings of these restaurants are also displayed in a chart (Figure 4.2.2) so that users can easily compare ratings among them.

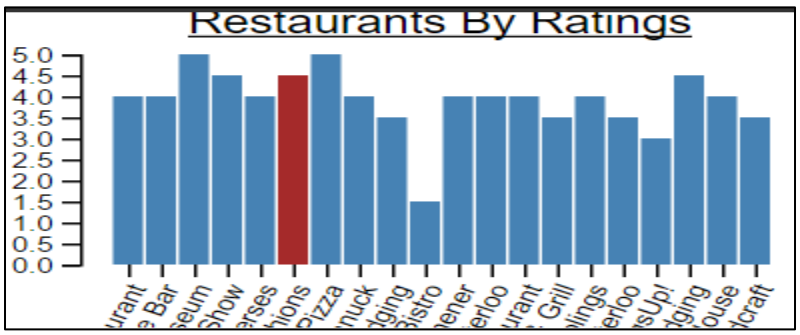


Figure 4.2.2 – Bar Chart for rating comparison

Users can select a restaurant to view how busy the restaurant is at different time in each week day in the past.

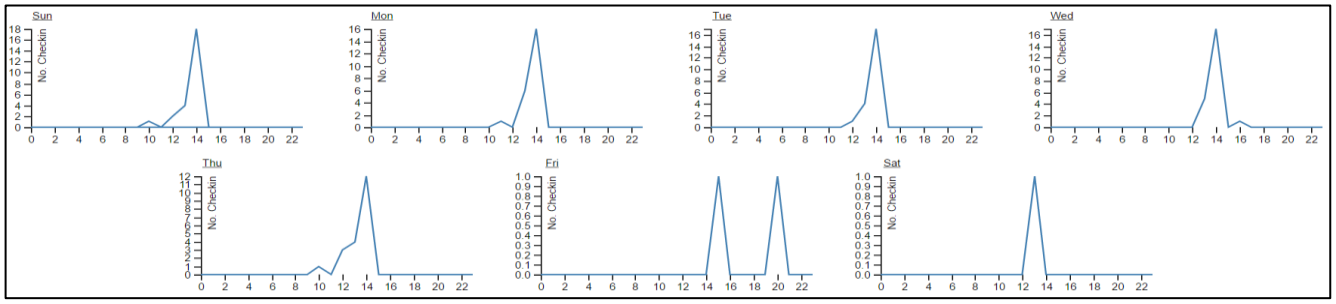


Figure 4.2.3 – Line Charts showing average number of check-in(s) at different time

4.3. Summary

We have described our approach and solution for Yelp to further engage its existing customers. With this recommender system solution, Yelp would be able to add value to customers through personalized and location-awareness restaurant recommendations. Furthermore, Yelp can even leverage on location analytics and send a push notification to a customer when he/she is near by one of the top recommendations.

Through the project, we can see that two crucial components in big data solution are data format and parallelism. Even through we analyze and design the data structure before proceeding to implementation, we still face challenges in handling data format such as Spark MLlib requiring integer instead of alphanumeric format and D3.js requiring time-series data source to be sorted in advanced. In fact, we spend 70-80% of our implementation time to handle challenges in data. In addition, we learn that setting the right level of parallelism from Cassandra and Spark can significantly speed up big data tasks. Hence, it is important that during planning phase for big data project, we align on end-to-end solution, architecture, and data so that rework and wait-time can be minimized during implementation.

REFERENCES

1. *Yelp Data Challange* [link]
2. *Yelp sample code to convert JSON datasets to tabular format* [link]
3. *Project Interactive Demo* [link]
4. *Project Github [link]*

APPENDIX – Project Summary

Category	Points
Getting the data: downloading datasets from Yelp	1
ETL: Spark job to clean the datasets and perform Extract-Transform-Load into Cassandra tables	2
Algorithmic work: MLLib ALS to train the model over provided users' ratings, predict ratings of the restaurants for users and recommend top 20 restaurants per state for the users.	3
Bigness/parallelization: 2.5GBs of data with 600K+ users, 70K+ restaurants and 2M+ ratings	2
UI: Web Server retrieves user information from Cassandra DB and sends it to UI.	5
Visualization: Spatial Visualization of restaurant locations and Charts Visualization for ratings and check-in information	4
New Technologies: Cassandra, Spark MLlib, NodeJS, Google API, D3.JS	3
TOTAL	20