Location-based Collaborative Filtering and Frequent Itemset A Restaurant Recommendation System for Yelp

Thanh Tung Nguyen (ID: 40042891)

Huy Nguyen (ID: 40023289)

Date: April 17, 2019

Abstract

Before going out for a meal, Yelp has been one of the most popular choices for customers to check for restaurants quality. To help users make better choices, we use techniques and principles of recommendation systems to create an application which makes predictions based on the user similarities. Using Yelp's dataset, we develop an enhanced collaborative filtering using location as a key criterion for generating recommendations and then extract collaborative and content-based features to identify customer and restaurant profiles. Besides, we also provide frequent itemset references to users subsequently based on their chosen restaurants. We would evaluate our algorithm using Root metrics Mean Squared Error and Mean Absolute Error, we then evaluate and compare the algorithms. Due to limitation of time and resources, our scope of work will be narrowed to businesses within Canada.

Keywords

collaborative filtering, frequent itemset, association rules, recommendation, location-based, location, Yelp dataset.

I. Introduction

As a very popular platform for choosing restaurants, Yelp contains an immense database of reviews, ratings, and other information provided by the community about businesses. Yelp's restaurant ratings are being becoming a key performance indicator of to access whether a restaurant is successful and popular. However, reading all the reviews and ratings of a single business and compare them with others could take so much time from users. Therefore, we decided to build a recommendation system based on restaurant ratings which could greatly benefit Yelp users and food lovers by removing all the challenges processing a huge amount of data to make an informed choice.

Recommendation systems have been widely implemented in e-commerce websites to recommend and provide customers with suggestive information helping them decide which products to buy. For example, media-services providers, such as Netflix, iTunes and Spotify, utilize recommendation systems to suggest songs, videos, movies to users based on their previous choices and taste. Given this general theme, techniques and principles of recommendation systems as well as taking users' locations into consideration, we shall create a simple restaurant recommendation app to help Yelp users make quick and better choices

Related Work

Thanks to the vast amount of information contained in the Yelp dataset, numerous past projects and researches and tried to use it to give food and restaurant recommendations to users. For example, Sumedh Sawant and Gina Pai [1] used the network-based-inference collaborative filtering algorithm to develop a recommendation system on the Yelp Dataset Challenge dataset. Sawant also had another Collaborative Filtering recommendation system project on Yelp dataset using Weighted BiPartite Graph Projection [2]. In addition, with regards to using location for personalized POI recommendations in mobile environments,

we found a research paper of done by Tzvetan Horozov, Nitya Narasimhan, Venu Vasudevan [3] proposing GeoWhiz app - a real-world deployment of our restaurant recommender system for location-based points of interest (POI). In this project, we apply collaborative filtering and frequent itemset recommendation methods on restaurants ratings and aims to work on a location-based analysis instead of a nationwide user-based analysis in order to provide suggestions to Yelp restaurants.

II. Materials and Methods

a. Materials

Our primary dataset is the Yelp's businesses retrieved from www.kaggle.com/yelp-dataset/yelp-dataset that contains actual business, user, and users' review data from North America region. From the original dataset, we extracted business information located in Canada as our predefined scope of work. The Canadian businesses were found using postal codes. The Canadian postal code listing is obtained from Google Fusion Tables at https://fusiontables.google.com/. Some basic characteristics of the dataset are summarized as below:

Original Dataset	**Number of entries**
- Number of businesses	192,609
- Number of reviews	6,685,900
- Number of users	1,637,138
Canada	
- Number of Canadian businesses	50,644
- Number of Canadian reviews.	1,063,142
Canadian Postal Codes	
- Number of postal codes	889,320

For this project, we combined the review, business and user datasets and picked diverse feature set for developing the machine learning model. The structure of the different datasets was available in individual json files. We co-related the data in the files, joined, denormalized them and extracted the desired features.

b. Methodology

The theory and formal definitions of our methods for this project are presented in this section before we present our app algorithm in the next section.

Collaborative Filtering

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating).

For our project, we want to use matrix factorization [4], a more sophisticated machine learning technique used in recommender system, in collaborative filtering. We will discuss what is matrix factorization and how is it implemented in Spark.

In collaborative filtering, matrix factorization is the leading-edge solution for sparse data problem. Matrix factorization is a factorization of a matrix into a product of matrices. In the case of collaborative filtering, matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. One matrix can be seen as the user matrix where rows represent users and columns are latent factors. The other matrix is the item matrix where rows are latent

factors and columns represent items. In the sparse user-item interaction matrix, the predicted rating user u will give item i is computed as:

$$\tilde{r}_{ui} = \sum_{f=0}^{nfactors} H_{u,f} W_{f,i}$$

(where H is user matrix, W is item matrix)

Frequent Itemset - Association Rule Mining

Association rules [5] are if-then statements that help to show the probability of relationships between data items within large data sets in various types of databases. It can tell you what items customers frequently buy together called Frequent Itemset.

Association rules are given in the form as below:

$X => Y[Support, Confidence]$

$$Rule: X \Rightarrow Y \quad Support = \frac{frq(X,Y)}{N} \quad Confidence = \frac{frq(X,Y)}{frq(X)} \quad Lift = \frac{Support}{Supp(X) \times Supp(Y)}$$

Support and Confidence measure how interesting the rule is. It is set by the minimum support and minimum confidence thresholds. These thresholds set by client help to compare the rule strength according to your own or client's will. The closer to threshold the more the rule is of use to the client.

- **Frequent Itemsets:** Item-sets whose support is greater or equal than minimum support threshold (min_sup). This is set on user choice.
- **Strong rules:** If a rule $X => Y[Support, Confidence]$ satisfies min_sup and min_confidence then it is a strong rule.
- **Lift:** Lift gives the correlation between X and Y in the rule X=>Y. Correlation shows how one item-set X effects the item-set Y.

c. Algorithm

Our recommendation system attempts to reduce the complete Yelp dataset of North America businesses to a more manageable subset that can make relatively accurate results. Our initial assumption for our project to work is that people who live in the same neighborhood are likely to visit the same local places since people can be correlated only if they have common rating items. For those who travel to a new place, the chances for the collaborative filtering recommendation to show its effectiveness are relatively low. Hence, we can further infer that there is a greater possibility of correlating users who live in the same area (city, adjacent postal codes) than correlating people who live further apart.

To validate our assumption, we use Yelp rating data for location-based POIs. We develop our own collaborative filtering system for recommending restaurants and use the collected business rating data to prove or disprove our theory. Our scope of work and dataset details are presented above. Our initial objective is to recommend restaurants for mobile users whose locations can be given through GPS services. However, due to due to limited time, our project isn't able to provide such features. As a replacement for GPS services, postal codes are used to identify users' locations.

Select user with good history profile

We sort top 100 most review users in Canada, then take 5 random users and select one. That we can ensure that our user selection is random within the users with most reviews of restaurants. The selected user's rated restaurants will then be compared with others' in the given location to pick up ones with highly-correlated rating profiles.

Find the location and most reviewed postal codes of the selected user

To make sure the user's rating history and their selected locations (postal codes) are relevant, we will find the base city and top most reviewed postal codes of the user based on their rating history. Based on that, we then prompt the user input their desired location (select one of the top postal codes).

Find businesses within search area

After the user entered his/her location, we will find all postal codes and their associated businesses located within a 3-km radius from the chosen postal code. We can increase the search radius if needed. We then apply the collaborative filtering on that list of targeted businesses (such as from 3 to 5, and to 10km)

Apply and compare basic ALS recommender and global average recommender

We use Pyspark MLlib ALS library to do basic ALS recommender. In order to evaluate the effectiveness of ALS recommender, we compare it with global average recommender. We compute RMSE and MAE for both approaches and take recommendation from the better RMSE approach.

Apply frequent itemset - association rule mining

We use FP-Growth Library in Pyspark to perform frequent itemset which will list out top most frequently chosen items. The confidence and support values are expected as below:

Number of Canadian businesses is 50,644 (a). Number of Canadian reviews is b = 20a. Number of Canadian users c = 9a. This means

- 1 business is reviewed by 5 users on average
- 1 user rated 9 restaurants on average (8 reviews)

As we can see, to have at least two users having same two or more choices, the best scenario is that we need 2 out of 9 users rating the same restaurants twice or more. Therefore, our expectation for confidence and support values is:

$$Confidence_{max} = Support_{max} = \left(\frac{9}{20} \times \frac{9}{20}\right)^2 = 0.04$$

Whereas, for the worst case, it would take all 9 out of 9:

$$Confidence_{min} = Support_{min} = \left(\frac{9}{20}\right)^9 = 7.57 \times 10^{-4}$$

III. Results

As the result of step 1, 2 and 3, we have obtained lists of businesses for the chosen user and location of **tWBLn4k1M7PLBtAtwAg73g** and **M8X 1E9** respectively:

	3 km	**5 km**	**10 km**
Number of postal codes	2,835	6,731	21,050
Number of businesses	582	1,506	8,005

In order to test the accuracy of our results, we decided to use cross validation technique when checking our performance with several evaluation metrics and distance parameters. However, first of all, we have to break the Yelp dataset into two chunks of 80% and 20% each. The first set of 80% is used to train the system and the second set of 20% is used to test the system.

For evaluation metrics, we use two popular metrics used for measuring the performance of recommendation systems, which are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) [4]. Given that our system generates predicted ratings $r^*_{u,b}$ of a user u for a business b in test set T, RMSE and MAE are defined as:

$$RMSE = \sqrt{\frac{1}{|T|}\sum_{(u,b)\in T}(r^*_{u,b} - r_{u,b})^2} \qquad MAE = \sqrt{\frac{1}{|T|}\sum_{(u,b)\in T}|r^*_{u,b} - r_{u,b}|}$$

Then, we use those two metrics to evaluate the results of our algorithm. We also want to compare our algorithm performance with global average recommender and bias ALS recommender. As for global average recommendation system, it is a widely known simple technique where the average of the given set and use it as prediction value for all items. On the other hand, for bias ALS, we have to define formula for bias system. In particular, we computed user-item interaction bias for user u and item b by taking the actual stars user *u* given to *b* subtracts with the mean of stars by *u*, then adding it with the value of subtracting global average with the mean stars of item *b* given by all users in the set.

Below we compare the performance of running different collaborative filtering strategies:

RMSE	**3 km**	**5 km**	**10 km**
Basic ALS Recommender RMSE	2.127391175	2.147847095	1.852597978
Global Average Recommender RMSE	1.435300565	1.435300565	1.329057787
Bias ALS Recommender RMSE	3.042489076	3.02685857	3.322852162

MAE	**3 km**	**5 km**	**10 km**
Basic ALS Recommender MAE	1.722178277	1.717569799	1.460206047
Global Average Recommender MAE	1.242512298	1.242512298	1.112019584
Bias ALS Recommender MAE	2.554701563	2.589625447	2.923318758

Frequent itemset

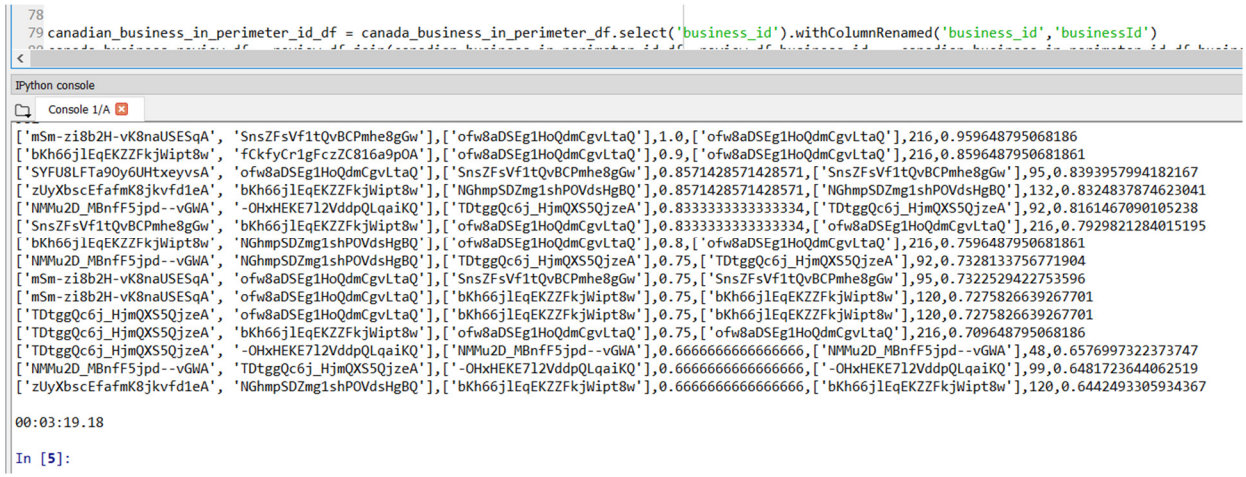
As for frequent itemset, we want to have start with the maximum support and confidence sequentially decrease down to the minimum in our rage of expectation. Thus, we firstly tried 0.04 in the FP-Growth algorithm. Unfortunately, it gave us empty result. Since our searching parameter is small, there is a possibility that there indeed no frequent item around it. Hence, we want to increase the length radius to 5 km, and sequentially to 10 km if the algorithm is still return empty result.

Unfortunately, it is. Thus, we opt to the option that decreasing our confidence and support until the algorithm gave us result. It is only when we downgrade it to 0.001 that the algorithm gave non-empty result.

Below is the result of FP-Growth library performing on out dataset:

FP-Growth	3 km		5 km	10 km
	max	reduced	max	max
Antecedent	(empty)	See screenshot below	(empty)	(empty)
Consequence	(empty)		(empty)	(empty)
Confidence	(empty)		(empty)	(empty)
Lift	(empty)		(empty)	(empty)

Stepping down support and confidence values to 0.001, we have the below itemset:



IV. Discussion

From all of the results above, we therefore suspect that Yelp official open dataset is not suitable for small scale locality recommendation as there is a low possibility of finding similar rated business among users. In particular, we would like to elaborate it into two specific problems:

Issue 1: ALS performance is much worse than global average

Although the performance of ALS recommenders is improved as the increase in parameter size, the bias ALS Recommenders are going the opposite direction. Thus, we can deduct possible scenario that Yelp dataset is very random, resulting in the inaccuracy of user-item interaction results. Another possible reason is that there are not many similar users within small scale location, resulting in wrong prediction when performing matrix factorization table.

It should be noted that we also tried to increase rank and iteration in ALS parameter but still having no success, global average recommender still beats ALS one on both RMSE and MAE.

In order to make improvement for our recommendation system, we are looking at two possible solutions:

1. Look for alternative way to do recommendation, such as Cluster Weighted BiPartite Projection or Multi-Step Random Walks. For example, the project performed by Sawant - "Collaborative Filtering using Weighted BiPartite GraphProjection - A Recommendation System for Yelp" shows remarkable improvement.
2. Additional data attributes and information from Yelp could be taken into account, such as type of restaurant and its price range to improve algorithm, in order to give a more precise result.

Issue 2: FP-Growth returns empty

FP-Growth gives out empty result when confidence and support values are higher than 0.001. This could mean there is not much of similarity between users' frequent itemsets within small-distance localities. Although, for confidence and support values great than 0.001, we have inspected it several times by increasing search radius from 3km to 5km and to 10km as well as stepping down confidence and support values, we kept receiving empty rdds until the values being reduced to 0.001. We may conclude that there is low possibility of frequent patterns in a small-scale location-based recommendation system. To increase the similarity between users' rating profiles, we could group businesses by types of food or by original countries. This should improve the results of collaborative filtering and frequent itemset algorithms. This also require further preprocessing of input data to group businesses by categories.

REFERENCES

[1] Sumedh Sawant and Gina Pai, " Yelp Food Recommendation System"; http://cs229.stanford.edu/proj2013/SawantPai-YelpFoodRecommendationSystem.pdf

[2] Sumedh Sawant, " Collaborative Filtering using Weighted BiPartite Graph Projection - A Recommendation System for Yelp "; https://pdfs.semanticscholar.org/fd9a/bc146ef857b55eebfe38977cd65e976f36db.pdf.

[3] Tzvetan Horozov, Nitya Narasimhan, Venu Vasudevan, " Using location for personalized POI recommendations in mobile environments "; http://horozov.soundmore.com/tzvetan/docs/icccn.pdf

[4] Kevin Liao, " Prototyping a Recommender System Step by Step Part 2: Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering "; Towards Data Science [Online] https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1.

[5] Hafsa Jabeen, " Market Basket Analysis using R", Data Camp [Online], https://www.datacamp.com/community/tutorials/market-basket-analysis-r?fbclid=IwAR0704r07BBLOzGnb0LyHoCGnquq6obggQZVF28x-AbXqkR5aEN_ortquw4