Big Data Project

Yelp Dataset

Data Aggregation and Recommendation System Implementation

Truc Cao

Tejinder Kaur

Amaan Shareef

Kathia Teran

Table of Contents

1. Abstract

Yelp is a popular online crowd-sourced local business review platform which allows users to browse and contribute reviews to businesses that they have visited. In the US alone the restaurant industry is projected to generate 899 billion dollars in sales for the year 2020 and there are more than a million restaurants employing in total 15 millions workers. Restaurants spend 3 to 6% of their sales in marketing. With the millennials generation heavily relying on reviews in their restaurants choices, being able to understand how the data are distributed across multiple criteria(cuisine, zip code, rating) and how to efficiently recommend restaurants based on their past visits can really be a game changer.
We will see through our aggregations what the most popular cuisines are, rating, how the number of checkins evolve over time and how we can efficiently recommend restaurants to users using ALS.

2. Problem statement and motivation

Yelp dataset contains millions of raw data on businesses, users and reviews. In our project we are trying to make sense of the data by finding interesting patterns in cuisine, rating and reviews using aggregations. We took it a step further by implementing a recommender system to suggest restaurants to potential customers and help restaurants increase their revenue.

To this end, we use Spark, a big data framework to cleanup, filter and aggregate data. And for our recommendation system we based our approach using some research papers and studies found online and we use SparkML, a machine learning library built on top of Spark Core.

3. Related work

In the paper “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions” [1], the authors present an overview of recommender systems and describe the current generation of recommendation methods that are usually classified into three main categories: content-based, collaborative, and hybrid recommendation approaches. This paper helped us have a more in-depth understanding on how to implement a recommendation system.

The paper “Matrix Factorization Techniques for Recommender Systems” [2] describes the Alternating Least Squares (ALS) algorithm which is used to learn and find latent factor matrices using collaborative filtering. This paper helped us understand how the algorithm works, so we are able to use it correctly and tune the parameters accordingly in our project.

In the paper “Large-Scale Parallel Collaborative Filtering for the Netflix Prize” [3] the author describes a parallel algorithm, alternating-least-squares with weighted-λ-regularization (ALS-WR) in the case of the Netflix prize. This algorithm is designed to be scalable to very large datasets, and helps avoid overfitting. After reading this paper we decided to use ALS-WR which is well suited for our project since our data is big and sparse.

In the article “Amazon.com recommendations: Item-to-item collaborative filtering” [4], the authors discuss item-to-item collaborative filtering. While the article was interesting, we quickly decided that their approach did not fit our need because in our dataset, we do not have much information about the restaurants.

The paper “Google news personalization: Scalable online collaborative filtering” [5] describes an approach to collaborative filtering for generating personalized recommendations for users of Google News. They generate recommendations using three approaches: collaborative filtering using MinHash clustering, Probabilistic Latent Semantic Indexing (PLSI), and co visitation counts. They combine recommendations from different algorithms using a linear model. This paper was interesting because it showed a novel way to implement a recommendation system. However it is quite difficult for us to implement so we decided not to use it but we might consider implementing in the future.

The paper “Restricted Boltzmann Machines for Collaborative Filtering” [6] proposes an innovative way to implement a scalable collaborative filtering for very large datasets using a class of two-layer undirected graphical models, called Restricted Boltzmann Machines (RBM’s) which can be used to model tabular data, such as user’s ratings of movies. This paper can help us scale our approach if we had to consider larger datasets (e.g., the full Yelp dataset). Since the Yelp dataset that we considered is relatively small, we decided to use ALS.

In the paper “Modeling relationships at multiple scales to improve accuracy of large recommender systems” [7], the authors proposed a neighborhood-based technique which combines k-nearest-neighbor (kNN) and low-rank approximation to obtain significantly better results compared to either technique alone. Their team won the progress prize in October 2007 improving the CineMatch score by 8.5%. If we had more time, we would have considered implementing it to provide more accurate recommendations.

The paper “Collaborative Filtering via Ensembles of Matrix Factorizations” [8] presents a Matrix Factorization (MF) based approach for the Netflix Prize competition. For the Netflix Prize, they used three different types of MF algorithms: regularized MF, maximum margin MF and non-negative MF. For our project, this paper helps to know different types of matrix factorization algorithms.

4. Dataset Description

4.1. Overview

Yelp is a popular online crowd-sourced local business review platform.

For our project we use a sample dataset provided by

Yelp from this link: https://www.yelp.com/dataset.

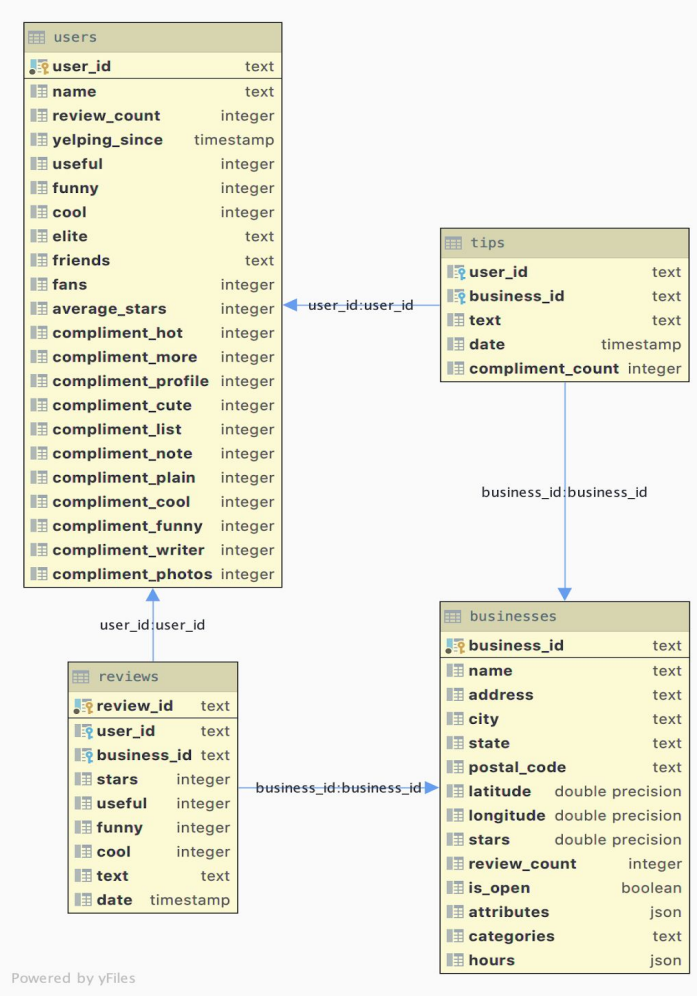
The dataset contains:

- 192,609 businesses
- 1,637,138 users
- 6,685,900 reviews
- 1,223,094 tips
- 161,950 check-ins
- 36 states
- 1,307 cities

All the files are in JSON format. Each line of the files is a dictionary that contains different fields that can be of type integer, string, float and also json themselves that contain additional information. We represented below the file review.json that we reformatted for better readability.

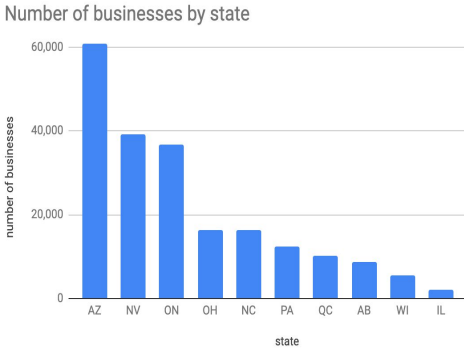

```
{
  "review_id": "BTDBNxb7m6wuSTy09_Zz4A",
  "user_id": "oV4PUFp4O2brd3bGhu5cjg",
  "business_id": "m97jaBYRscg-hqDjMVIIWg",
  "stars": 4,
  "useful": 0,
  "funny": 0,
  "cool": 0,
  "text": "This is our go-to place for lunch and just a friendly atmosphere.
  "date": "2017-03-03 22:35:42"
}
```

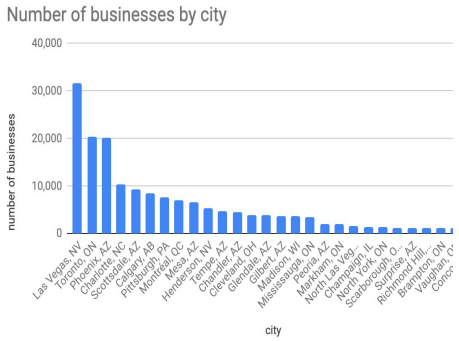

File review.json

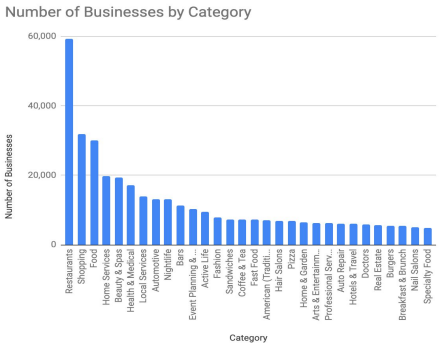


Entity-Relation Schema of the tableas

4.2. Dataset distribution







The dataset includes data for businesses from 36 states from the US and Canada but there are only 11 states that have over 1,000 businesses that we represent on this chart.

Out of 1,307 cities, only 28 have more than 1,000 businesses. Las Vegas has the most businesses followed by Toronto and Phoenix. We can also see that the most represented category is “Restaurant” so in this project we will focus on restaurants only.

4.3. Preparing the data

In order to prepare the data, we use Jupyter Lab and Spark SQL:

- Jupyter notebook is an interactive notebook in the browser, we can use it to share codes and execute them easily with the other team members as well as taking note.
- SparkSQL is a component of Spark built on the top of Spark Core. It’s easier to read and write than dataframe operations because we can use SQL to execute queries. Also using SparkSQL, we can take advantage of Spark to scale to multiple machines to speed up the process.

In order to clean the data, we first filter the data to only keep restaurants. Then we break down the categories which are stored as a string that contains the categories separated by commas. So for example the string “ramen, sushi” should become 2 rows, one for ramen and one for sushi on the right hand side.

Then we want to categorize it using 32 different kinds of cuisines so for example Ramen and Sushi should be both mapped to Japanese cuisine. To do the mapping we manually create the mapping rules in the file *cuisine.csv*. We can see for example that American cuisine is mapped to American (traditional), American (new), Steakhouses and so on.

business.json

	business_id	categories
0	eBEfgOPG7pvFhb2wcG9I7w	Restaurants, Cheesesteaks, Poutineries
1	LoRef3ChgZKbxUio-sHgQg	Restaurants, Lebanese, Middle Eastern

	business_id	category
0	eBEfgOPG7pvFhb2wcG9I7w	Restaurants
1	eBEfgOPG7pvFhb2wcG9I7w	Cheesesteaks
2	eBEfgOPG7pvFhb2wcG9I7w	Poutineries
3	LoRef3ChgZKbxUio-sHgQg	Restaurants
4	LoRef3ChgZKbxUio-sHgQg	Lebanese

cuisine.csv

	cuisine	category
0	American	American (Traditional)
1	American	American (New)
2	American	Steakhouses
3	American	Bagels
4	American	Cajun/Creole

	business_id	cuisine
0	eBEfgOPG7pvFhb2wcG9I7w	American
1	LoRef3ChgZKbxUio-sHgQg	Middle Eastern

We first read the file and map it to the table *cuisine_mapping*. Then we run a SQL query to join it with our *restaurant_categories* table, in case of multiple cuisine we select the last one. By doing an inner join we make sure that restaurants with no cuisine are filtered out. Then we further clean up the *restaurant* table by removing those belonging to cities with less than 200 restaurants. And we clean up the other tables *users* and *reviews* by removing those which are not associated with any remaining restaurants.

After filtering, cleaning and normalizing, the final dataset contains:

- 41,029 restaurants (-78%)
- 1,006,767 users (-39%)
- 3,447,322 reviews (-48%)
- 668,619 tips (-45%)
- 39,896 check-ins (-75%)
- 10 states (-72%)
- 30 cities (-98%)

5. Methods Description

5.1. Data Distribution Analysis and time series

In order to compute aggregations and time series, we use Spark SQL using the data that we generated earlier. In the *checkins* table, we have a string that contains for each business and user all the dates that the user visited the restaurant. So again we have to convert it into multiple rows by using explode and split. And then we run an aggregation that we join with business to aggregate by date, city, state and cuisine. To illustrate it we're going to show the check ins for Toronto and we only keep the top cuisines and using Spark directly we pivot the table so that cuisines become columns and then we store it in a CSV file.

5.2. Influencers

To computer influencers, we thought it would be interesting to use Spark Core directly which is the lower level component of Spark upon which Spark SQL is based on.

Using Spark Core, we use the low level RDD (Resilient Distributed Dataset) which unlike dataframe is not structured and only allows basic operations. The goal of writing it using the low level API is just to show how tedious it is to do it compared to using Spark SQL. Below we show a simple example to do it using RDD and then in Spark SQL.

Doing it using Spark SQL offers 2 advantages:

- Easier to write and maintainable
- Faster since it avoids serialization/deserialization between the JVM and Python

Spark Core (RDD)

```
raw_reviews = spark.sparkContext.textFile("review.json")
raw_reviews.map(lambda line: json.loads(line))
review_user_ids = reviews.map(lambda x: x['user_id'])
reviews_per_user = review_user_ids.map(lambda x: (x, 1)).reduceByKey(lambda a, x: a + x)
top_users = reviews_per_user.sortBy(lambda user_id_num_reviews: -user_id_num_reviews[1])
```

Spark SQL

```
spark.sql("""
    SELECT
        user_id,
        count(1) as num_reviews
    FROM reviews
    GROUP BY user_id
    ORDER BY num_reviews DESC
""")
```

We used the 6 categories to define an influencer which are users with the highest amount of: friends, fans, useful rating, review count, elite status, and oldest accounts ('yelping since'). In this logic, a user with the highest amount of friends, doesn't necessarily have the highest amount of fans, or review count. But all those factors influence other users. Las Vegas has the highest amount of reviews in the dataset, so it is important to consider who are the leading influencers in Las Vegas. Restaurant categories (cuisine) is also important, and what other cities are reviewed by the strongest influencers.

5.3. Recommendation system

In our project, we are interested in implementing a recommendation system that suggests restaurants to customers.

There are several techniques [1] that we can use:

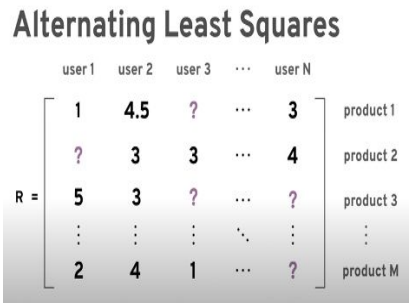
- Content based recommendation: which relies on the restaurant description and the user profiles.
- Collaborative Filtering: which uses similarities between users for the prediction.

By looking at the Yelp dataset, we can see that the information on each restaurant is very limited (name, address, categories and other optional fields with so many missing data). Therefore, we can not use the content based approach. In addition, most of the users wrote less than 4 reviews which makes the data very sparse. In this situation, using collaborative filtering through matrix factorization would be a good choice.

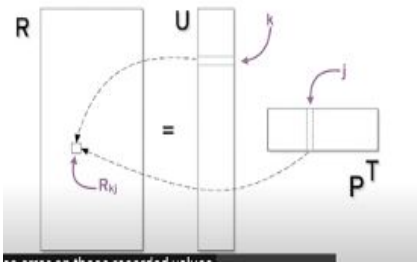
We will use ALS (Alternative Least Square) a popular industry standard algorithm used in collaborative filtering. It is very

efficient in terms of performance during the training process. It is implemented in SparkML machine learning library, a main component built on top of Spark core. Spark allows us to run tasks on multiple machines to parallelize the computation to reduce the computation time. ALS was used during the Netflix recommendation prize competition.

5.3.1. ALS algorithm



The idea is to factorize a ratings matrix R into two factor matrices U and P (U is our matrix of user factors , P is our matrix of product factors). When we multiplied them back together, it gives an approximation of the original ratings matrix. The approximation matrix will have all the cells filled with a rating prediction. So by looking at the values that were not present in the original matrix we can have the predicted rating of the user for the product.



It's alternating because the process that generates those matrices U and P is done first by fixing U optimizing for P and then fixing P and optimizing for U and we repeat this process to minimize the cost. It is done alternatively to make the computation more efficient.

The cost function is defined as follow:

$$\min_{x_\star, y_\star} \sum_{r_{u,i} \text{ is known}} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

It uses weighted lambda regularization to prevent overfitting (ALS-WR).

5.3.2. Preparing the data

In order to prepare the data, we use Jupyter Notebook, a web interactive notebook that allows us to write Python code and see the result on the same page. Because it is interactive, we can tune the parameters easily by changing the parameters and directly see the result updated on the page.

For our recommendation, we are going to split the dataset by city to personalize the recommendation by city. From the dataset, we can see that there are 2 cities which have the most restaurants: Las Vegas and Toronto. However, Las Vegas is a very touristic city, so most of the customers only stay for a few days and can only review a handful of restaurants. Whereas Toronto has more local users. Therefore, we choose Toronto to train our first model and tune the hyperparameters accordingly. Once we got relevant results, we trained the model for the 10 other cities using the same hyperparameters.

For Toronto, we have 7,000 restaurants and 85,000 users. With this number of restaurants and users, our utility matrix would be too large with 600 millions cells and cannot be computed on a single laptop. In this context, we want to have a sample that allows the utility matrix to be dense enough so the support can be high enough and at the same time not require too much memory and computation power. To this end, we limit the dataset by keeping only the most relevant users(more than 5 reviews) and restaurants(more than 40 reviews).

We end up with 2,200 restaurants and 13,000 users which gives a more manageable matrix with 26 millions cells. Number of Reviews: 144,621 => sparse = 0.5%.

5.3.3. Splitting training and testing data

To build the model, we have to split the dataset into a training and testing dataset.

To take into account the fact the users are influenced by their past visits, we are going to sort the reviews by time and use the 80% oldest reviews as the training dataset. And the 20% more recent reviews as the testing dataset.

In the chart you can see that 80% of the reviews are before 2017.

5.3.4. Tuning the parameters

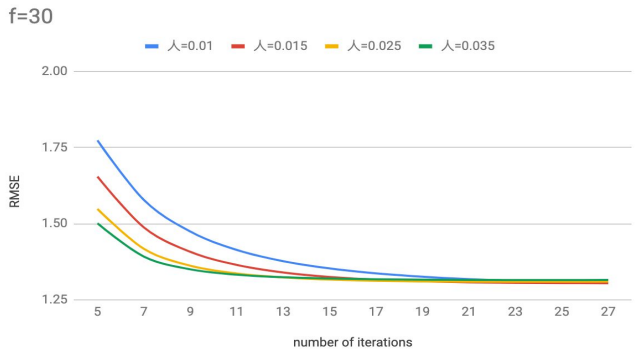
```
Model = ALS.train(rank=?, lambda=?, iterations=?, userCol="user_id", itemCol="business_id", ratingCol="stars")
```

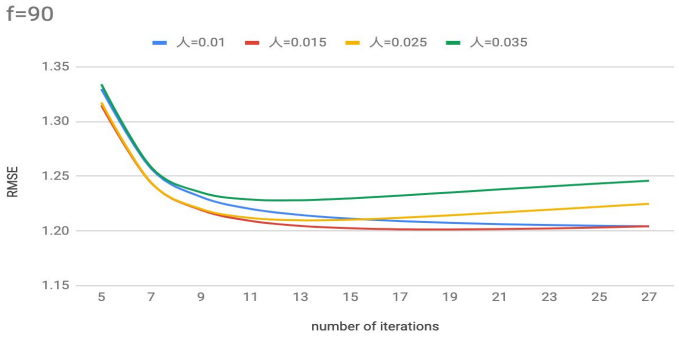
Tuning parameters

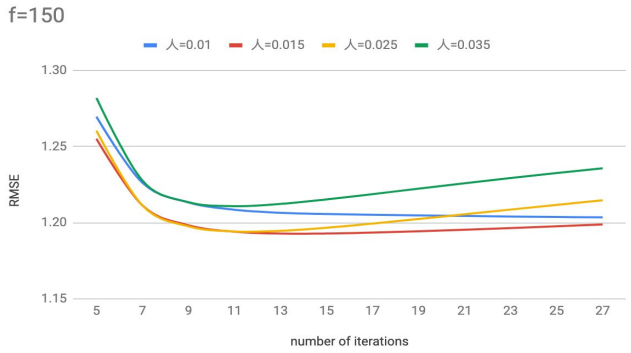
ALS relies on 3 hyperparameters:

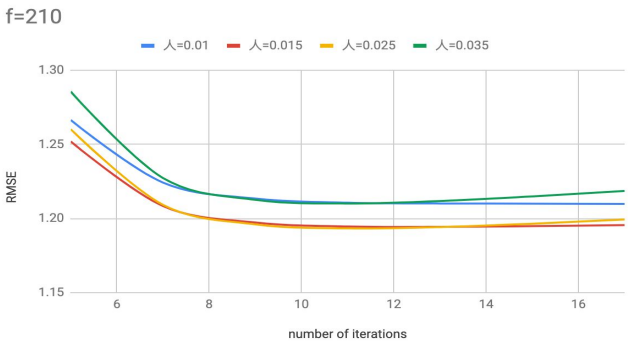
- The rank which is the number of features for the latent factor matrices
- Iterations which is the number of iterations
- Lambda which is the regularization parameter used in the cost function to avoid overfitting.

We try to find the hyperparameter values that minimize the cost (RMSE). We first started to run the experiments with different combinations but after a full day it did not even finish. So for the tuning only, we decided to reduce the dimension by only considering restaurants having more than 100 reviews and users who wrote more than 10 reviews. It allows us to run the 420 experiments in about 10 hours. We end up having: 895 restaurants x 5458 users = 4.8 millions, reviews = 76942 ⇒ sparse = 1.5751%.







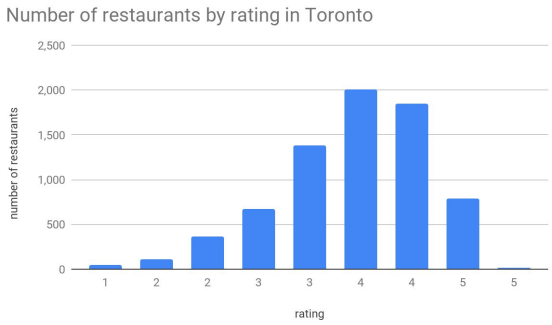


We can see from the charts that the number of iterations have a significant impact on the cost (RMSE). After 12 iterations, the cost does not get better and can become worse in some cases. We can also see that the red line (alpha=0.015) is lower than the other lines. Finally we can see that rank f=150 gives the lowest cost and even if we increase it to 210 it does not improve much. So we use 150 to not overfit the model.

The best experiment is with 150 features(rank), lambda=0.015, and 10 iterations which has the lowest cost. We will use those values to train our model on the bigger dataset.

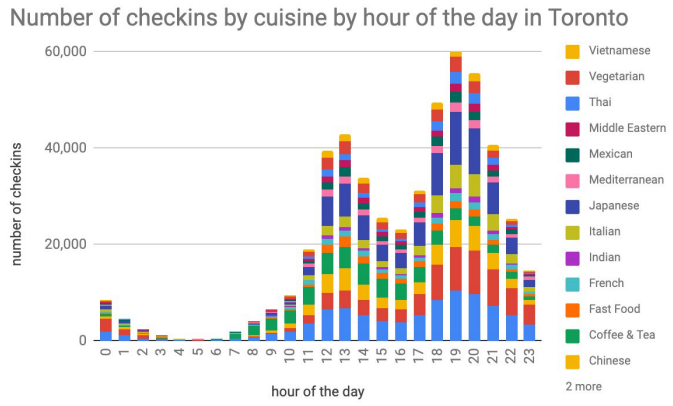
6. Results

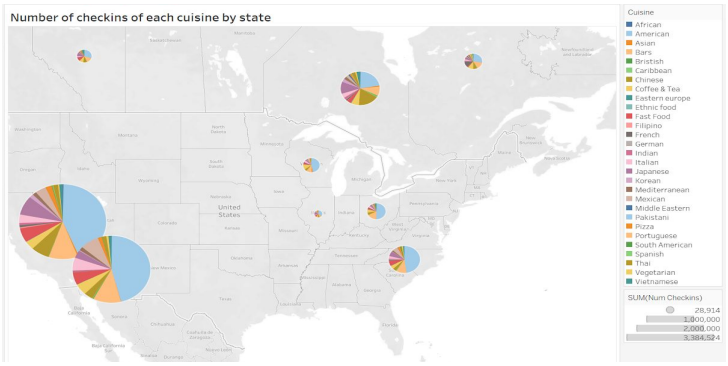
6.1. Data Distribution Analysis



Top restaurants by review in Toronto

	name	num_reviews
0	Subway	54
1	McDonald's	51
2	Tim Hortons	49
3	Starbucks	42
4	Freshii	41

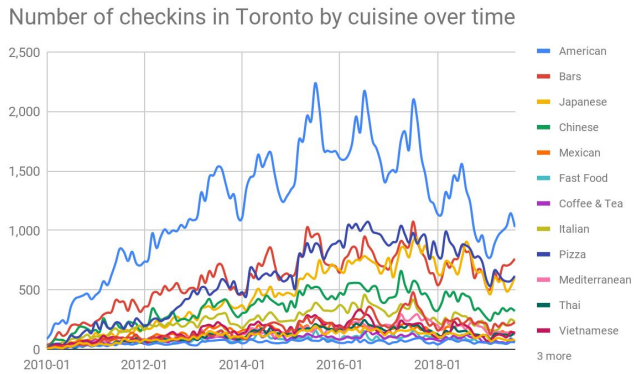


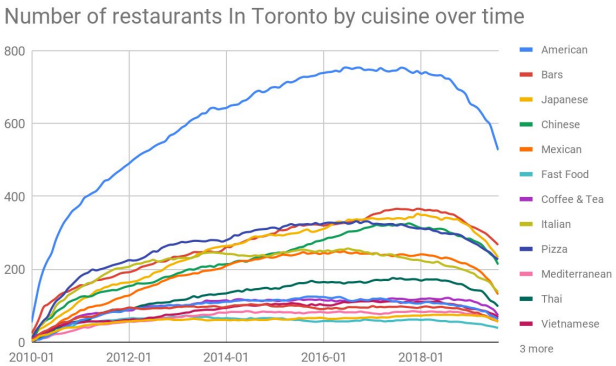


We can see most people check-in around 12pm and 7pm which makes sense since it's lunch and dinner time. On the map we can see that the most popular cuisine is American.

6.2. Time series

Number of checkins / cuisine / city over time





The number of checkins belonging to different cuisines over time. So we can see in the chart that for Toronto, the number of checkins in American restaurants increased significantly from 2010 to 2016 and then decreased until now.

6.3. Influencers

Below is the full list of the most influential users based on what we considered essential factors:

- Influencer with the most **'Friends'** in Las Vegas

user_id	business_id	name	city	friend_count	name
y4aEhXdfKAX8klEvOszSCA	n4E8ltk6CidI4_9sSSTVtA	Primeburger	Las Vegas	148	Sarah
8bvDfT1CiN8baG-TRy3Emw	fL-b760btOaGa8SOJ9ut3w	Rollin Smoke Barbeque	Las Vegas	148	BZ

- **'Useful'** influencer in Las Vegas (review count is from the business)

user_id	business_id	name	city	review_count	name	useful
--2vR0DIsmQ6WfcSzKWigw	IB8zLlGraOg9LU7qQVLPyg	Fashion Show	Las Vegas	739	Harald	154202

- 'Oldest (**yelping since**)' influencer in Las Vegas (business review count)

user_id	business_id	name	city	review_count	name	yelping_since
nkN_do3fJ9xekchVC-v68A	ubz4CaZXagQuGv2N9gFAdw	Botero	Las Vegas	434	Jeremy	2004-10-12 08:46:43

- Influencer with the most **'Fans'** in Las Vegas (business review count)

user_id	business_id	name	city	review_count	name	fans
37cpUoM8hlkSQfReIEBd-Q	0qet57CmMASqUm6gPFUTpg	Di Fara Pizza	Las Vegas	150	Mike	9538

Based On **City**

- Restaurant and city with the highest review count:

business_id	name	city	review_coun
4JNXUYY8wbaaDmk3BPzlWw	Mon Ami Gabi	Las Vegas	8348
f4x1YBxkLrZg652xt2KR5g	Hash House A Go Go	Las Vegas	5763
cYwJA2A6I12KNkm2rtXd5g	Gordon Ramsay BurGR	Las Vegas	5484

- Influencer with the most **'review counts'** in Las Vegas

user_id	business_id	name	city	name	review_count
8k3aO-mPeyhbR5HUucA5aA	6Q7-wkCPc1KF75jZLOTcMw	Circus Circus Las Vegas Hotel & Casino	Las Vegas	Victor	13278

II. Based On **Category**:

- Influencers with the most **'Friends'**

user_id	business_id	name	categories	friend_count	name
UDATvQLaHkx-FiMPQIGh0Q	cRBIvRyynPBjDi2y49PV6g	Strings Ramen	Ramen, Japanese	148	Thai
6yRliUr4a4NOa87y86oEKQ	iFlrQeG_qH-cgXFhji7gbQ	Erin's Snug Irish Pub	Breakfast & Brunch, Irish, American	148	Marla

- Influencer with the most **'Fans'**

user_id	business_id	name	city	categories	name	fans
37cpUoM8hlkSQfReIEBd-Q	lYCeqldIiOggsbByH3RRhw	Di Fara Pizza	Las Vegas	Italian	Mike	9538

- 'Oldest (**yelping since**)' influencer

user_id	business_id	name	city	categories	name	yelping_since
c6HT44PKCaXqzN_BdgKPCw	u8C8pRvaHXg3PgDrsUHJHQ	Papa Del's Pizza	Champaign	Food Delivery Ser..	Russel	2004-10-12 08:40:43

- Influencer with the most '**review counts**'

user_id	business_id	name	city	categories	name	review_count
8k3aO-mPeyhbR5HUucA5aA	6Q7-wkCPc1KF75jZLOTcMw	Circus Circus …	Las Vegas	Arts & Ent., Restaurants..	Victor	13278
RtGqdDBvvBCjcu5dUqwfzA	oUX2bYbqjqST-urKbOHG6w	Loftti Cafe	Las Vegas	Desserts, Juice Bars, etc.	Shila	12390

- Most '**Useful**' influencer in Las Vegas

user_id	business_id	name	city	categories	name	useful
--2vR0DIsmQ6WfcSzKWigw	uanCi4OGc1mHLGl_AT4JhQ	Treasure Island	Las Vegas	Hair Salons, Arts...	Harald	154202

III. Cities reviewed by the strongest influencers

- Las Vegas on top:

user_id	business_id	name	city	name	review_count
8k3aO-mPeyhbR5HUucA5aA	6Q7-wkCPc1KF75jZLOTcMw	Circus Circus..	Las Vegas	Victor	13278

- Las Vegas has the Influencer with the most **'Useful'** reviews (business review count)

user_id	business_id	name	city	review_count	name	useful
--2vR0DIsmQ6WfcSzKWigw	IB8zLlGraOg9LU7qQVLPyg	Fashion Show	Las Vegas	739	Harald	154202

- Champaign has the 'oldest (**yelping since**)' influencer (business review count)

user_id	business_id	name	city	review_count	name	yelping_since
c6HT44PKCaXqzN_BdgKPCw	u8C8pRvaHXg3PgDrsUHJHQ	Papa Del's Pizza	Champaign	402	Russel	2004-10-12 08:40:43

- Las Vegas has the influencer with the most **'fans'** (business review count)

user_id	business_id	name	city	review_count	name	fans
37cpUoM8hlkSQfReIEBd-Q	0qet57CmMASqUm6gPFUTpg	Di Fara Pizza	Las Vegas	150	Mike	9538

IV. Top influencers:

- Users with the most **'Friends'**

count	user_id	name
148	m2G3Az7rhKzOM6GmRLWU2Q	Kristina
148	6yRliUr4a4NOa87y86oEKQ	Marla

- The most **'Useful'**

user_id	name	useful
--2vR0DIsmQ6WfcSzKWigw	Harald	154202
JjXuiru1_QNzDkYVrHN0aw	Richard	99162
W7DHyQlY_kXls2iXt-_2Ag	Maggie	89792

- The most **'review counts'**

user_id	name	review_count	average_stars
8k3aO-mPeyhbR5HUucA5aA	Victor	13278	3.28
RtGqdDBvvBCjcu5dUqwfzA	Shila	12390	3.85
hWDybu_KvYLSdEFzGrniTw	Bruce	10022	3.61

- The most **'fans'**

user_id	name	fans
37cpUoM8hlkSQfReIEBd-Q	Mike	9538
hizGc5W1tBHPghM5YKCAtg	Katie	2964
eKUGKQRE-Yw15dY55_zChg	Cherylynn	2434

- The oldest **'yelping since'**

user_id	name	yelping_since
c6HT44PKCaXqzN_BdgKPCw	Russel	2004-10-12 08:40:43
nkN_do3fJ9xekchVC-v68A	Jeremy	2004-10-12 08:46:43
wqoXYLWmpkEH0YvTmHBsJQ	Michael	2004-10-12 08:51:07

- Most recent **Elite** influencer

user_id	name	elite
3Fmj7MfGfsUUK1kTWCSL_g	Matthew	2018

- Cities with users with the most **'Friends'**

user_id	city	friend_count	name
RR8z_wcpxlH8Hx3E8MrPjg	Madison	148	Debbie
8bvDfT1CiN8baG-TRy3Emw	Scottsdale	148	BZ

6.4. Recommendation system

Best Hyper Parameters: rank = 150, iterations = 10, lamba = 0.015

We use those parameters to train the full dataset for 11 cities. It takes about 10 minutes per city so about 2 hours for 11 cities. By testing the model on the testing dataset, we got the RMSE shown on the right hand side.

For Toronto, the RMSE is 1.40. For the other cities, it is higher as the hyperparameters were not tuned for them.

City	RMSE
Las Vegas	1.502491158
Toronto	1.408828704
Phoenix	1.479543317
Montreal	1.873554089
Calgary	2.004483722
Charlotte	1.473693724
Pittsburgh	1.591344719
Scottsdale	1.669494006
mississauga	2.015476996
Cleveland	1.873677623
Mesa	2.376086108

For the post preprocessing, before presenting the recommendation to a user, we filter out the recommendations of the restaurants that the user has already visited. This way, we encourage people to try new restaurants.

Example:
User: Alana
City: Toronto

Previous visits:

cuisine	number_restaurants
American	44
Bars	15
Mexican	9
Japanese	8
Italian	7
French	6
Chinese	3
Fast Food	3
Thai	2
Coffee & Tes	2
Vegetarian	2
Indian	1
African	1
Spannish	1
Pakistani	1
Mediterranean	1

Recommendations:

Restaurants	cuisine
Gallery Grill	American
Let's Be Frank	Bars
The Keg Steakhouse + Bar - Esplanade	American
Almond Butterfly Cafe & Bakeshop	Coffee & Tea
Voodoo Child	Bars

We see that the recommended restaurants are American and bars. And by looking at the type of restaurants Alana visited in the past, we can see that she mostly visits American restaurants and bars which confirms the recommendations.

7. Conclusion

7.1. Summary

In this project, we first analyzed the Yelp dataset by plotting multiple charts of the data and then aggregated the data to get general information, time series, and influencer. After that, we implemented a recommendation system using collaborative filtering through matrix factorization based on the ALS algorithm. We implemented most of our data analysis using Spark and the Jupyter notebook and finally presented our result in a web application.

7.2. Challenges

In this project, we had to deal with a large amount of data. The ALS recommendation method that we used consumes a large amount of memory so we had to find reasonable ways to sample down our dataset to run it on our laptop using limited memory and CPU.

7.3. Future Work

In this project we ran all of our tests on a single laptop. If we had more resources we could have considered using a cluster of machines to run our algorithm on a bigger datasets. We could also use more memory using an instance on AWS EC2 (r5.metal which provides 768GB of RAM).

We used Spark to run the computation but other frameworks such as Tensorflow that takes advantage of the hardware (using GPUs) can be used to run it more efficiently.

As for the recommendations, in addition to ALS, we might consider other models such as neural networks (RBM) and other hybrid approaches.

Finally it would be interesting to deploy this approach on a production setup using A/B testing to see if our recommendations can drive more sales.

7.4. Contribution

Here is the breakdown of our contributions:

- Truc Cao: Restaurant recommendation using Spark ML

- Tejinder Kaur: Data distribution analysis and time series using Spark SQL
- Amaan Shareef: Data preprocessing using Spark SQL
- Kathia Teran: Influencer using Spark Core (low level API)

We all contributed to the web application.

References

1. Gediminas, A., Alexander, T. (2005). “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”, retrieved from Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions
2. Yehuda, K., Robert, B., Chris, V. (2009). “Matrix Factorization Techniques for Recommender Systems”, retrieved from MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS
3. Yunhong, Z., Dennis, W., Robert, S., Rong, P. (2008). “Large-scale Parallel Collaborative Filtering for the Netflix Prize”, retrieved from Large-scale Parallel Collaborative Filtering for the Netflix Prize
4. Greg, L., Brent, S., Jeremy, Y. (2003). “Amazon.com Recommendations Item-to-Item Collaborative Filtering”, retrieved from Amazon.com recommendations item-to-item collaborative filtering - Internet Computing, IEEE
5. Abhinandan, D., Mayur, D., Ashutosh, G. (2007). “Google News Personalization: Scalable Online Collaborative Filtering”, retrieved from Google News Personalization: Scalable Online Collaborative Filtering
6. Ruslan, S., Andriy, M., Geoffrey, H. (2007). “Restricted Boltzmann Machines for Collaborative Filtering”, retrieved from Restricted Boltzmann Machines for Collaborative Filtering
7. Robert, B., Yehuda, K., Chris, V. (2007). “Modeling Relationships at Multiple Scales to Improve Accuracy of Large Recommender Systems”, retrieved from Modeling Relationships at Multiple Scales to Improve Accuracy of Large Recommender Systems
8. Mingrui, W. (2007). “Collaborative Filtering via Ensembles of Matrix Factorizations”, retrieved from Collaborative Filtering via Ensembles of Matrix Factorizations

Notes

We submitted the source code (source_code.zip) that contains:

- Jupyter Notebook for the data processing (code_data_processing.ipynb)
- Web application (code_web_application)

Just in case there is an issue with the Jupyter Notebook, we also put the PDF version of it (but it does not look as nice as the notebook when opened in Jupyter).