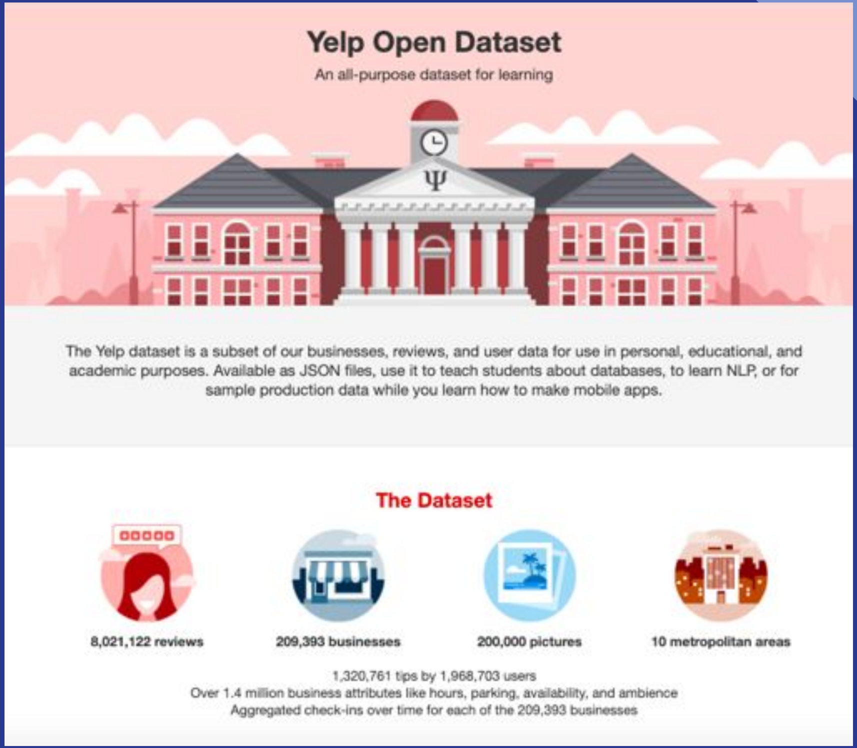
Big Data Project - Yelp Dataset

Data Aggregation and Recommendation System Implementation

Truc Cao

Kathia Teran



Problem and Motivation

Yelp dataset has a lot of data but hard to extract useful information

We want to:

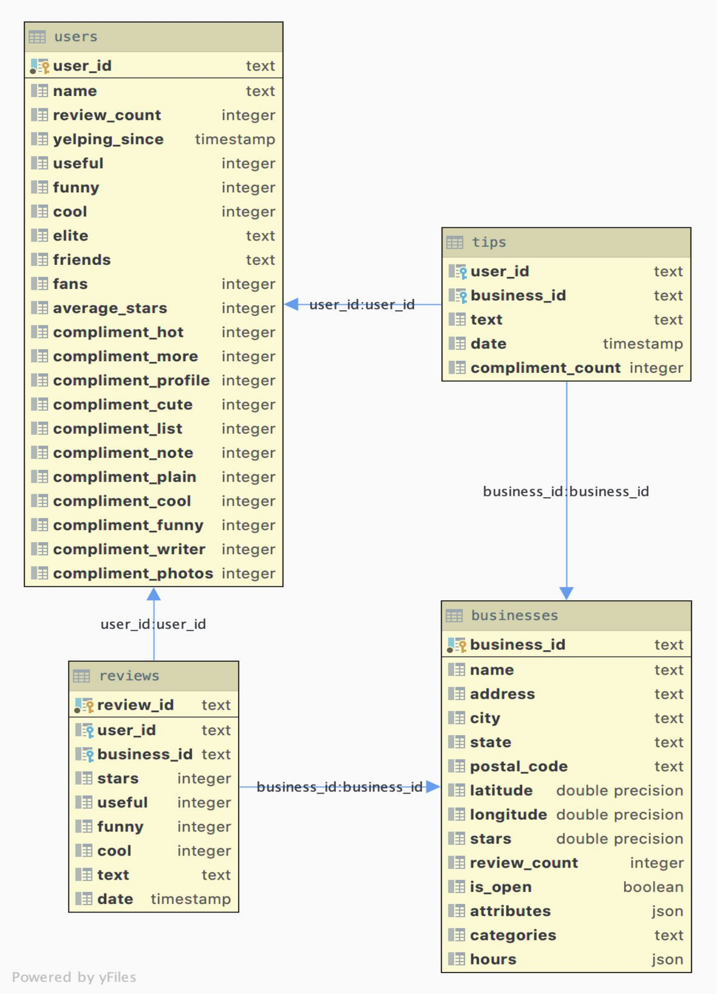
- Get statistics on restaurants (cuisines, popularity, …)
- Recommend restaurants to customers to increase revenue

Using:

- Spark (Big Data Framework) to cleanup, filter and aggregate data
- Research papers about recommendation system
- Spark ML (Machine Learning) to provide useful recommendation

Yelp Dataset

- Yelp is a popular online crowd-sourced local business review platform.
- Sample: 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 data files are in JSON format.



Samples


```
{
  "business_id": "0W27hbZN7Z-PrkhAb0y9Eg",
  "name": "B Montréal",
  "address": "1207-A Rue Rachel E",
  "city": "Montréal",
  "state": "QC",
  "postal_code": "H2J 2J8",
  "latitude": 45.5266477836,
  "longitude": -73.5735161233,
  "stars": 4.5,
  "review_count": 3,
  "is_open": 1,
  "attributes": {
    "RestaurantsPriceRange2": "1",
    "RestaurantsTableService": "False",
    "RestaurantsTakeOut": "True",
    "OutdoorSeating": "True",
    "RestaurantsReservations": "False",
    "Alcohol": "u'none'",
    "BikeParking": "True",
    "WheelchairAccessible": "True",
    "Caters": "True",
    "HasTV": "False",
    "GoodForMeal": "{'dessert': False, 'latenight': False, 'lunch': False, 'dinner': False, 'brunch': False, 'breakfast': False}",
    "GoodForKids": "True",
    "RestaurantsDelivery": "True",
    "WiFi": "u'free'"
  },
  "categories": "Coffee & Tea, Food, Juice Bars & Smoothies, Delis, Restaurants, Sandwiches",
  "hours": {
    "Wednesday": "9:0-20:0",
    "Thursday": "9:0-20:0",
    "Friday": "9:0-20:0",
    "Saturday": "8:0-16:0"
  }
}
```


business.json


```
{
  "business_id": "-Lw8Ve0NLbR0djHGw2fMOA",
  "date": "2014-06-30 22:57:27, 2014-06-30 22:58:28, 2017-05-19 18:24:18"
}
```


checkin.json


```
{
  "user_id": "zHseuNq_3b246ZgzcY8BXA",
  "name": "Briana",
  "review_count": 44,
  "yelping_since": "2009-10-16 23:54:29",
  "useful": 40,
  "funny": 3,
  "cool": 12,
  "elite": "",
  "friends": "_SEBcjCwgneOV1VV_vESfQ, vtNHzdtfsxCCsbc_JUK8Cw, EPyRgySYsR365gAgF2r4Ww, 3mNk60ynkQYRNJGf3YAqiA, NfU0zDaTMEQ4-X9dbQWd9A, X
  "fans": 2,
  "average_stars": 4.07,
  "compliment_hot": 0,
  "compliment_more": 1,
  "compliment_profile": 0,
  "compliment_cute": 0,
  "compliment_list": 0,
  "compliment_note": 0,
  "compliment_plain": 0,
  "compliment_cool": 0,
  "compliment_funny": 0,
  "compliment_writer": 0,
  "compliment_photos": 0
}
```


user.json

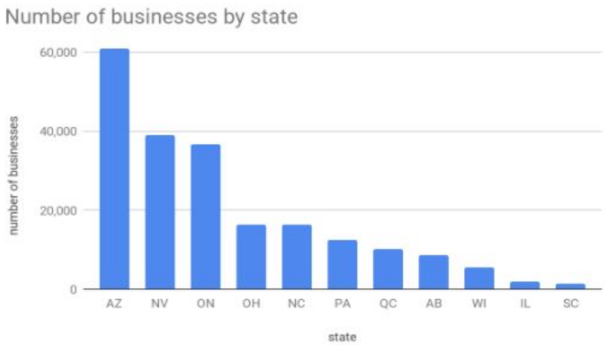

```
{
  "review_id": "BTDBNxb7m6wuSTy09_Zz4A",
  "user_id": "oV4PUFp4O2brd3bGhu5cjg",
  "business_id": "m97jaBYRscg-hqDjMVIIWg",
  "stars": 4,
  "useful": 0,
  "funny": 0,
  "cool": 0,
  "text": "This is our go-to place for lunch and just a friendly atmosphere. Very consistent food. A
  "date": "2017-03-03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

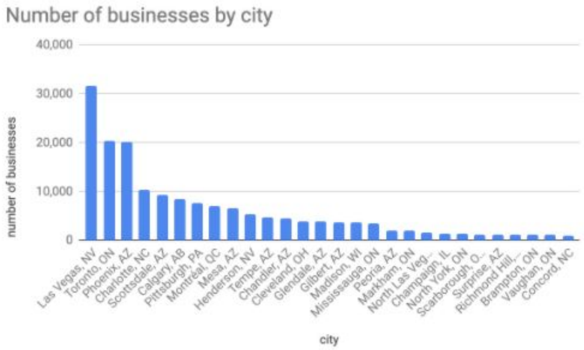

review.json

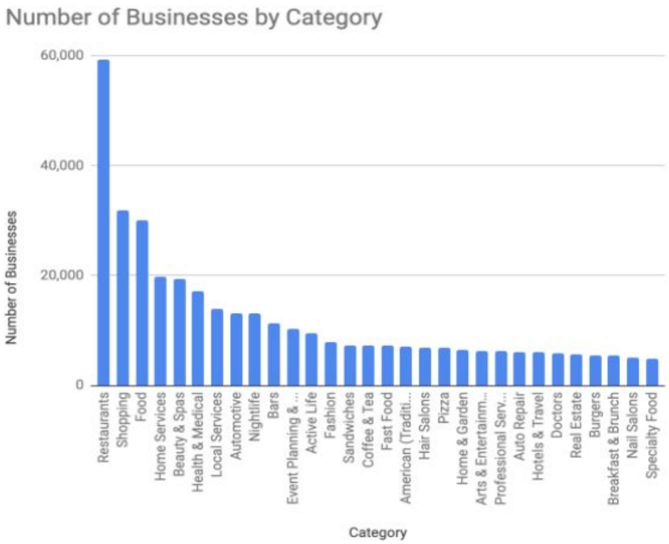

```
{
  "user_id": "ky3DB9i9lDJ7OAZdkZyv7g",
  "business_id": "m9ybLDUrbqgso1IT06bBLA",
  "text": "Excellent price on tires here!",
  "date": "2016-01-07 18:01:31",
  "compliment_count": 0
}
```


tip.json

Dataset: Distribution



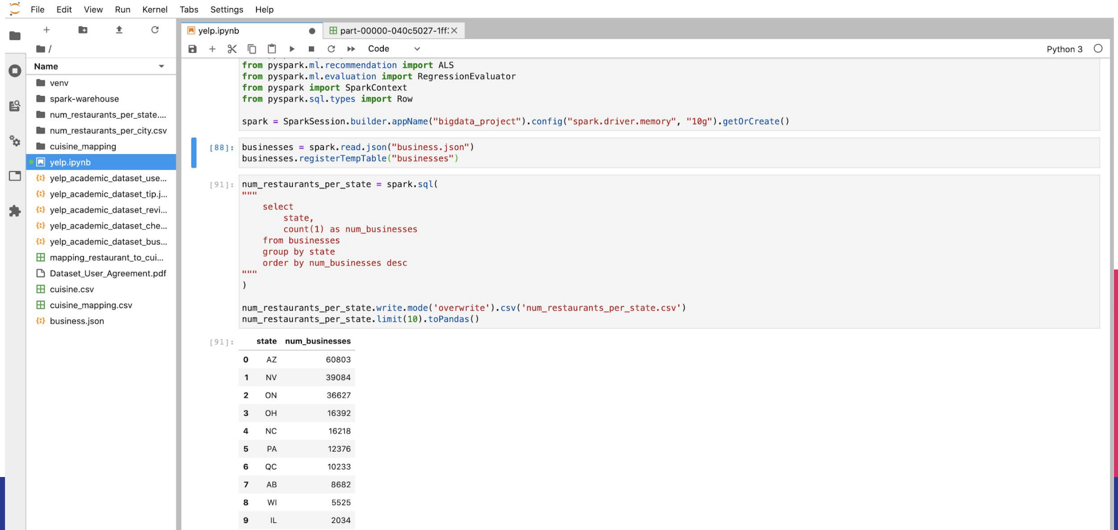




- Out of 36 states (US/Canada), 11 has more than 1,000 businesses
- Out of 1,307 cities, 28 has more than 1000 businesses
- Restaurant is by far the most represented category

Preparing the data

- Use Jupyter notebook and Spark SQL
- Jupyter notebook:
  - Interactive notebook in the browser
  - Can share easily with other people
- SparkSQL:
  - SQL queries are easier to write (and to read) than dataframe operations
  - Take advantage of Spark to scale to multiple machines to speed up the process



Preparing the data

Steps:

- Only keep restaurants: “Restaurants”
- Break down categories (one row per category):
  - “Ramen, Sushi” becomes 2 rows for each
- Recategorize using 32 kind of cuisines (American, Korean):
  - Use manually created mapping (114 mappings)
  - In case of multiple cuisine we choose one arbitrarily
- Only keep restaurants with a cuisine
- Only keep cities that have over 200 restaurants
- Only keep reviews and users associated to them

	cuisine	category
0	American	American (Traditional)
1	American	American (New)
2	American	Steakhouses
3	American	Bagels
4	American	Cajun/Creole
...	...	...
109	Asian	Cambodian
110	Korean	Korean
111	Asian	Laotian
112	Asian	BurmesePizza
113	Thai	Thai

cuisine.csv (Cuisine mapping)

Initialization

```
spark = SparkSession.builder.appName("yelp").config("spark.driver.memory", "10g").getOrCreate()
businesses = spark.read.json("yelp_academic_dataset_business.json")
businesses.registerTempTable("businesses")
```

	address	attributes	business_id	categories	city	hours	is_open	latitude	longitude	name
0	404 E Green St	(None, None, 'none', None, None, None, None, F...	pQeaRpvuhoEqudo3uymHIQ	Ethnic Food, Food Trucks, Specialty Food, Impo...	Champaign	(11:30-14:30, 11:30-14:30, None, None, 11:30-1...	1	40.110446	-88.233073	The Empanadas House
1	4508 E Independence Blvd	(None, None, None, None, None, None, None, Non...	CsLQLiRoafpJPJSkNX2h5Q	Food, Restaurants, Grocery, Middle Eastern	Charlotte	None	0	35.194894	-80.767442	Middle East Deli
2	15480 Bayview Avenue, unit D0110	(None, None, u'none', None, None, None, None, ...	eBEfgOPG7pvFhb2wcG9I7w	Restaurants, Cheesesteaks, Poutineries	Aurora	(11:0-22:0, 11:0-22:0, 11:0-22:0, 11:0-21:0, 1...	1	44.010962	-79.448677	Philthy Phillys

Break down categories:

```
restaurant_categories_df = spark.sql("""
    SELECT
        business_id,
        explode(split(categories, ', ')) as category
    FROM businesses
    where categories like '%Restaurants%'
""")
restaurant_categories_df.registerTempTable('restaurant_categories')
```

	business_id	category
0	AtD6B83S4Mbmq0t7iDnUVA	Sushi Bars
1	AtD6B83S4Mbmq0t7iDnUVA	Dim Sum
2	AtD6B83S4Mbmq0t7iDnUVA	Ramen

Map to cuisines:

```
cuisine_mapping_df = spark.read.csv("cuisine.csv", header=True)
cuisine_mapping_df.registerTempTable("cuisine_mapping")

restaurant_cuisines_df = spark.sql("""
    SELECT
        business_id,
        LAST(cuisine) as cuisine
    FROM restaurant_categories
    JOIN cuisine_mapping USING (category)
    GROUP BY business_id
""")
restaurant_cuisines_df.registerTempTable('restaurant_cuisines')
```

business_id	cuisine
--9e1ONYQuAa-CB_Rrw7Tw	American
-VAsjhmAbKF3Pb_-8rh3xg	Coffee & Tea
-cxD1NimFldATDUsN-oa3A	Mexican
-r8SvltXXG6_T3mP5GXRAw	Coffee & Tea
0859wfd1BQHG46Zpwhc0ZQ	Bars
09OYbFNrS1n8u5gE6W9ltA	German
0DwMrcy7_X_C_mP8_QcXug	American
0bqV9uzFVz98Bn_RlmcJTg	Mexican
0owlRP_z5RcYKKmh5RnD7A	Fast Food
1NmGVWYlF4iMngM6arKJTQ	Vietnamese

Final dataset

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

→

- 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%)

Computing different aggregations of the data

- Do all the aggregations using Spark SQL
- Number of restaurants /ratings / cuisine

```
num_restaurant_stars_city = spark.sql("""
    SELECT
        stars,
        count(1) as num_restaurants
    FROM restaurants_by_city
    GROUP BY stars
""")
```

	stars	num_restaurants
0	3.5	1913
1	4.5	687
2	2.5	662
3	1.0	39
4	4.0	1710

```
num_restaurant_cuisine_city = spark.sql("""
            SELECT
                cuisine,
                count(1) as num_restaurants
            FROM restaurants_by_city
            GROUP BY cuisine
        """)
```

	cuisine	num_restaurants
0	Mexican	204
1	Thai	202
2	Indian	260
3	Chinese	642
4	African	39

Computing different aggregations of the data

- Most popular restaurant by city and all based on ratings and number of reviews

```
restaurants_by_city = restaurants.filter((restaurants.city == 'Toronto') & (restaurants.state ==
restaurants_by_city.registerTempTable('restaurants_by_city')
```

```
most_popular_restaurants_reviews_city = spark.sql("""
    SELECT
        business_id,
        count(1) as num_reviews
        FROM reviews rev
        JOIN restaurants_by_city res USING (business_id)
        WHERE res.stars >= 4 and res.is_open = 1
        GROUP BY business_id
        ORDER BY num_reviews desc
        LIMIT 5
""")
```

	business_id	num_reviews
0	r_BrIgzYcwo1NAuG9dLbpg	2177
1	aLcFhMe6DDJ430zelCpd2A	1467
2	RtUvSWO_UZ8V3Wpj0n077w	1425
3	iGEvDk6hsizigmXhDKs2Vg	1183
4	N93EYZy9R0sdlEvubu94ig	1078

- Number of restaurants that are still active/closed for each city

```
active_close_restaurants_city = spark.sql("""
    SELECT
        is_open,
        count(1) as num_restaurants
    FROM restaurants_by_city
    GROUP BY is_open
""")
```

	is_open	num_restaurants
0	0	2291
1	1	4652

Time series data

Number of checkins / cuisine / city over time

Map tables from file

```
restaurants_df = spark.read.json("final_restaurants_all.json")
restaurants_df.registerTempTable('restaurants')

checkins_raw_df = spark.read.csv("final_restaurant_checkin_all.json")
checkins_raw_df.registerTempTable('checkins_raw')
```

Master aggregation by city/state, cuisine, month

```
checkins_by_date_df = spark.sql("""
    WITH checkin_date AS (
        SELECT
            business_id,
            explode(split(date, ', ')) as date
        FROM checkins
    )
    SELECT
        city,
        state,
        cuisine,
        substring(date, 0, 7) as date,
        count(1) as num_checkins
    FROM checkin_date
    JOIN restaurants USING (business_id)
    GROUP BY city, state, cuisine, date
    ORDER BY date
""")
```

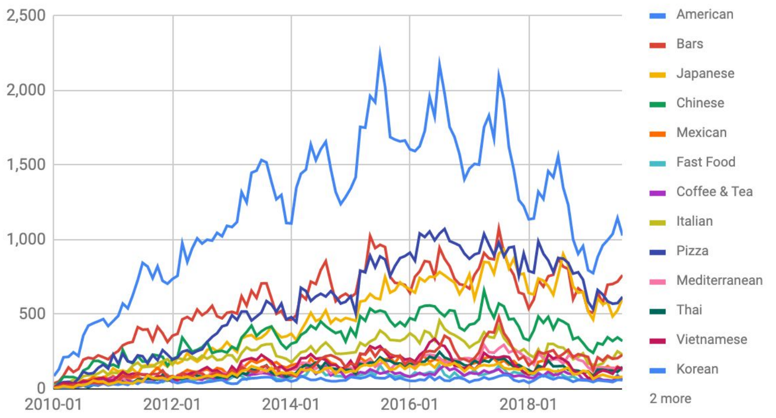
Pivot table for number of checkins for each cuisine in Toronto over time

```
popular_cuisines = ['American', 'Bars', 'Japanese', 'Chinese', 'Mexican', 'Fast Food', 'Coffee & Tea', 'Italian', 'Pizza', 'Mediterranean', 'Thai',
                    'Vietnamese', 'Korean', 'French', 'Indian']
tmp_checkins_cuisine_toronto_df = checkins_by_date_df.filter(f.col('city') == 'Toronto').filter(f.col('cuisine').isin(popular_cuisines)) \
                                                      .groupBy('date').pivot('cuisine').sum('num_checkins').sort('date')
checkins_cuisine_toronto_df = tmp_checkins_cuisine_toronto_df.na.fill(0)
checkins_cuisine_toronto_df.repartition(1).write.mode('overwrite').csv('checkins_cuisine_toronto.csv')
```

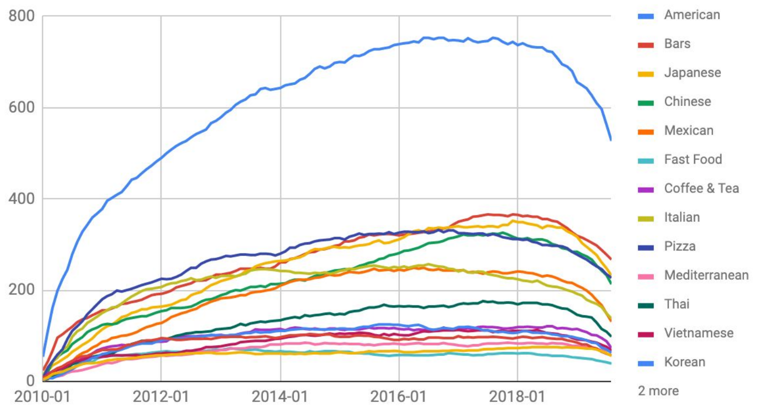
```
checkins_cuisine_toronto_df.toPandas()
```

	date	American	Bars	Chinese	Coffee & Tea	Fast Food	French	Indian	Italian	Japanese	Korean	Mediterranean	Mexican	Pizza	Thai	Vietnamese
0	2010-01	86	37	8	22	3	4	6	19	8	2	4	14	5	9	4
1	2010-02	132	50	27	41	4	8	19	32	35	6	5	18	6	17	10
2	2010-03	212	88	32	81	21	10	14	39	45	10	6	40	3	25	18
3	2010-04	215	144	35	80	8	20	16	45	47	11	9	36	7	22	13
4	2010-05	245	113	30	79	18	17	16	60	37	9	12	35	13	27	10

Number of checkins in Toronto by cuisine over time

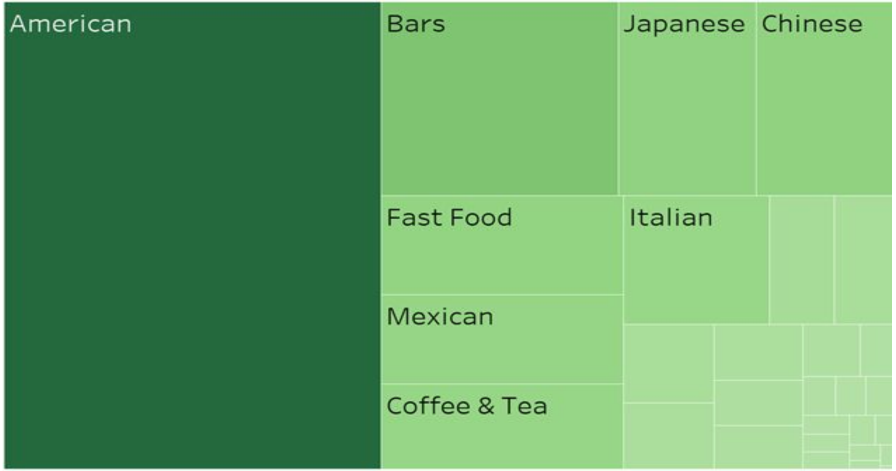


Number of restaurants In Toronto by cuisine over time



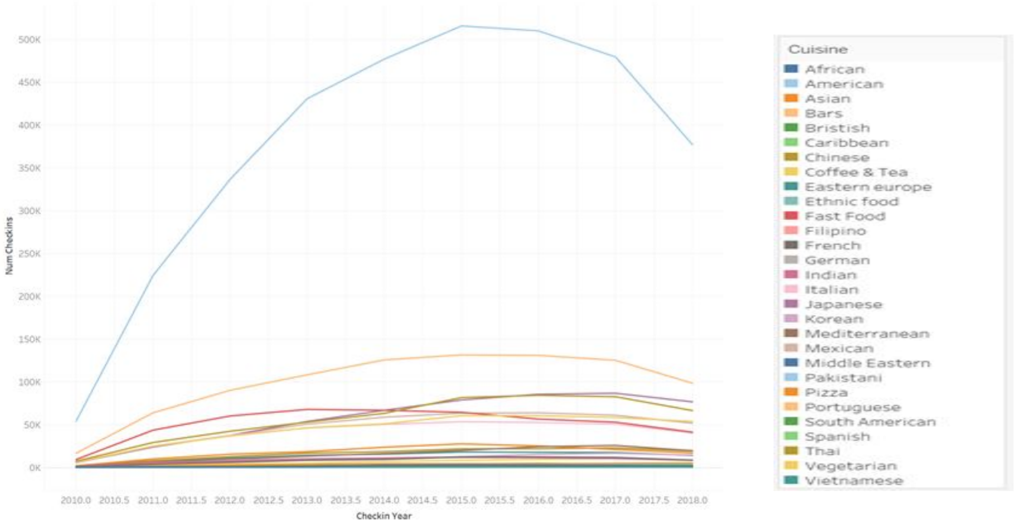
Number of check-ins by cuisine

cuisine	no_of_restaurants	no_of_checkins
American	7355	3405265
Bars	1928	891295
Japanese	1389	516445
Chinese	2402	510902
Fast Food	3493	463586
Mexican	1607	418008
Coffee & Tea	1800	399348
Italian	1558	362969
Pizza	1539	161191
Thai	459	144560
Mediterranean	972	137597
Vietnamese	515	116616
Korean	310	95565
Indian	676	77960
French	389	74545
Vegetarian	286	57780
Asian	94	32229
South American	165	24780
Caribbean	328	22264
Filipino	116	20099



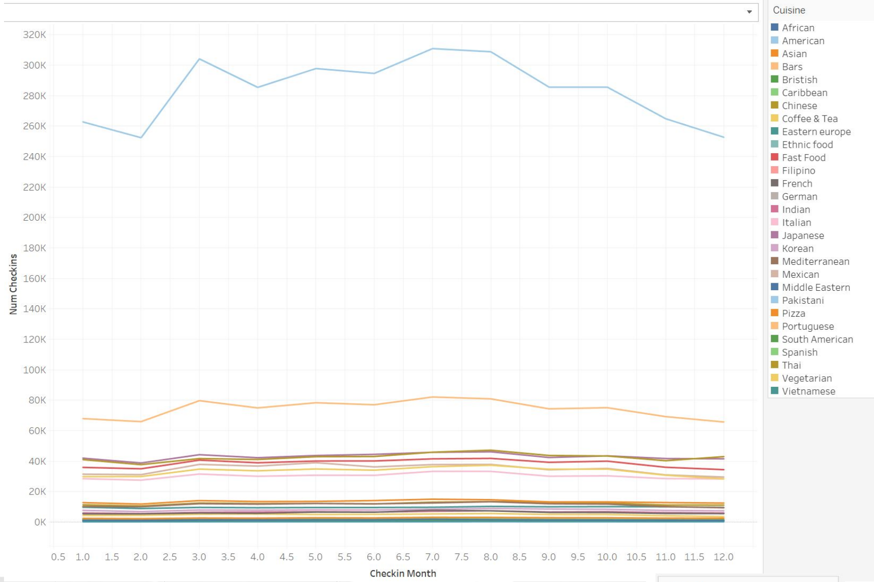
Number of check-ins by cuisine over years

cuisine	checkin_year	num_checkins
Pizza	2012	15594
Mediterranean	2012	9792
Asian	2016	4426
African	2016	2806
Pizza	2017	21134
Indian	2015	11859
Vegetarian	2014	7052
Mexican	2011	24581
Mediterranean	2011	6382
Fast Food	2010	9331
Japanese	2010	5907
African	2012	1050
Coffee & Tea	2011	24989
Bars	2013	108349
Pizza	2010	2090
Thai	2015	21913
Thai	2011	8113
Italian	2018	40307
South American	2018	3374
Ethnic food	2012	900



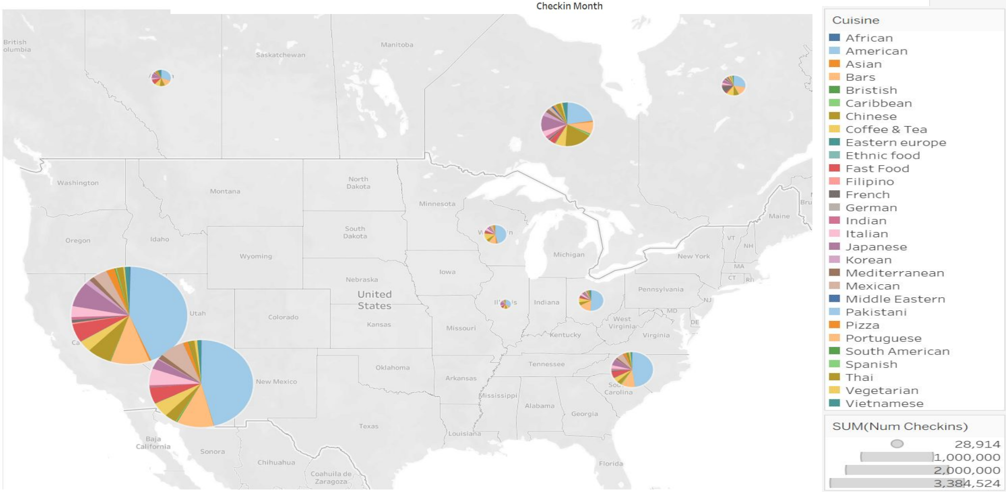
Number of check-ins by cuisine by month of the year

cuisine	1	2	3	4	5	6	7	8	9	10	11	12
Mexican	31451	31145	37865	36765	38972	36205	37733	37834	34336	35191	30963	29548
Thai	11285	10513	12463	12120	12197	11908	13044	13400	12924	12458	11144	11104
Indian	6001	5564	6393	6577	6760	6554	7073	7410	6632	6839	6194	5963
Chinese	41010	37705	41694	41105	42932	43060	45865	47176	43715	43380	40297	42963
African	1205	1122	1406	1428	1536	1523	1848	1604	1428	1368	1257	1135
Eastern europe	324	335	375	388	390	408	420	489	442	426	385	386
Japanese	41975	38787	44237	42253	43695	44461	45796	46133	42392	43478	41713	41525
Filipino	1464	1386	1654	1625	1850	1918	1977	1909	1630	1695	1563	1428
Fast Food	35901	34982	40680	38917	40082	40114	41503	41804	39225	39991	35976	34411
Spanish	1308	1361	1239	1173	1277	1421	1502	1539	1358	1299	1126	1112
Vietnamese	9755	8895	9631	9405	9573	9571	9721	10270	10037	10121	10039	9598
Pakistani	66	46	55	66	60	69	69	85	72	67	69	63
Pizza	12748	11809	14096	13468	13563	14116	15015	14650	13240	13225	12794	12467
Portuguese	284	332	387	400	479	479	522	546	494	442	365	292
Coffee & Tea	29822	29898	34767	33665	34858	34075	36365	37319	34681	34769	30769	28360
Italian	28482	27539	31527	30067	30757	30685	33293	33283	30091	30375	28528	28342
Caribbean	1481	1715	2005	1935	2015	1979	2124	2182	1933	1951	1511	1433
Vegetarian	4298	4409	5127	5221	4990	5077	5321	5626	4990	4945	4175	3601
Korean	7671	6871	7916	7824	8026	7951	8457	8924	8571	8384	7605	7365
French	5401	5401	5930	5777	6560	6481	7788	7377	6339	6279	5642	5570



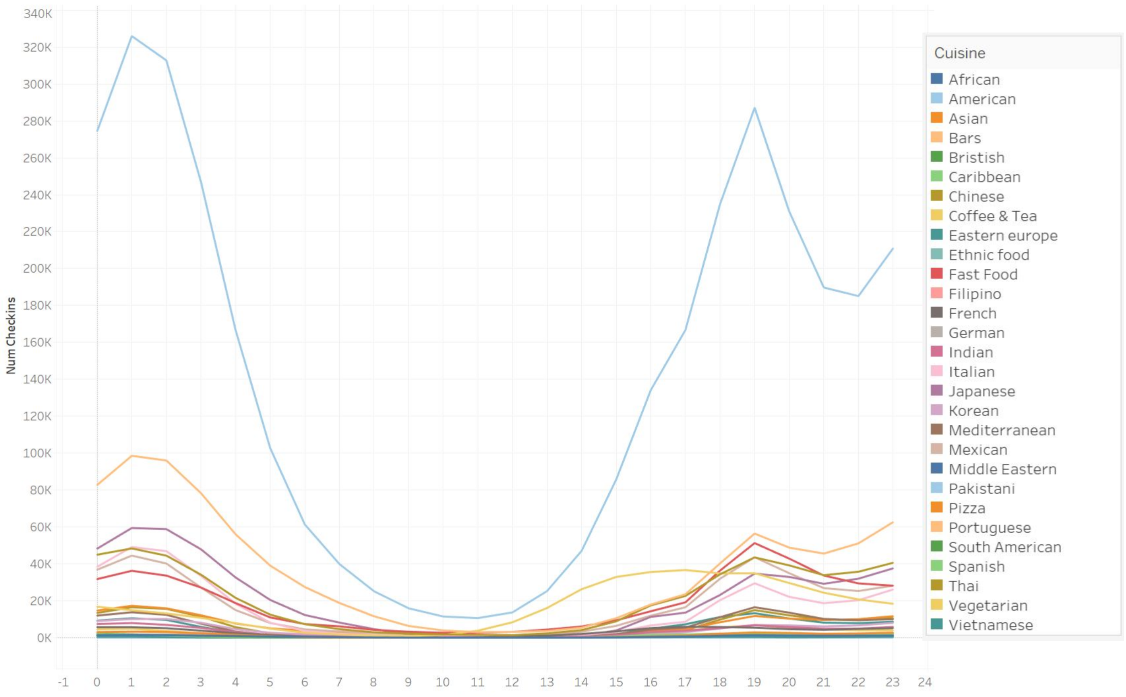
Number of check-ins/restaurants by cuisine in different states

state	cuisine	no_of_restaurants	no_of_checkins
WI	American	245	56904
WI	Coffee & Tea	56	13989
WI	Bars	51	13127
WI	Mexican	38	5884
WI	Chinese	45	5557
WI	Italian	38	5255
WI	Fast Food	66	4035
WI	Asian	7	2805
WI	Japanese	17	2522
WI	Pizza	26	2292
WI	Mediterranean	12	1672
WI	Korean	5	1663
WI	French	6	1510
WI	Indian	11	1217
WI	Thai	8	823
WI	Ethnic food	3	569
WI	Vegetarian	3	556
WI	Caribbean	6	527
WI	South American	4	434
WI	Vietnamese	2	333



Number of check-ins each hour by cuisine

- Peaks at 7pm in the evening and 1 am at midnight.



	cuisine	0	1	2	3	4	5	6	7	8	...	14	15	16	17	18	19	20	21	22	23
0	Mexican	36851	44494	40218	27201	14973	7965	4520	3456	2634.0	...	3270	6426	11967	16502	31973	43614	34968	26866	25324	28212
1	Thai	13436	16511	15630	11428	5722	2392	1157	783	532.0	...	273	1239	3592	4055	9808	14958	12135	9821	9497	10540
2	Indian	7431	7960	6976	5339	3065	1408	849	684	540.0	...	574	1332	3011	3796	5150	6762	5791	4899	4960	5872
3	Chinese	44977	48371	44469	34153	21608	12509	7345	4669	2934.0	...	4136	8887	17526	22644	34352	43531	39192	33895	35747	40595
4	African	1381	1522	1415	1170	975	653	497	283	167.0	...	80	208	490	694	1078	1324	1230	1131	1156	1208

Influencers: how top reviewers rate businesses

- Using RDD lower level API: (key, value) pairing, mostly using user_id as the key. Joining user, business and review datasets to obtain results.
- Influencers being those users with the highest amount of: friends, fans, useful rating, 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.

I. BASED ON CITY (considering Las Vegas as the most yelped city for restaurants)

- Restaurant and city with the highest review count:

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

- Code example - finding the user with the most fans in Las Vegas:

```
from pyspark.sql import SparkSession

path1 = "/Volumes/KATHIA/A.\ SPARK/final_restaurants_all.json"
path2 = "/Volumes/KATHIA/A.\ SPARK/final_restaurant_reviews_all.json"
path3 = "/Volumes/KATHIA/A.\ SPARK/final_restaurant_users_all.json"

# Create a SparkSession
spark = SparkSession.builder.appName("Popular_Reviewer").getOrCreate()

# Json raw data
business_dataFrame = spark.read.json(path1)
reviews_dataFrame = spark.read.json(path2)
users_dataFrame = spark.read.json(path3)

top_business = business_dataFrame.select("business_id", "name", "city", "review_count").orderBy("review_count", ascending=False)
top_reviews = reviews_dataFrame.select("business_id", "user_id").orderBy("user_id", ascending=False)
top_reviewer = users_dataFrame.select("user_id", "name", "fans").orderBy("fans", ascending=False)

top_reviewers = top_business.join(top_reviews, on=['business_id'], how='inner').orderBy("review_count", ascending=False).\
                     join(top_reviewer, on=['user_id'], how='inner').orderBy("fans", ascending=False)

top_reviewers.show(10, False) #False shows full column content

# Print the results
print("\n")

# Stop the session
spark.stop()
```

Most Popular Influencers

TOP

- **'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	0qet57CmMA5qUm6gPFUTpg	Di Fara Pizza	Las Vegas	150	Mike	9538

- 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:

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

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

Top influencers:

- The most **'Useful'**

user_id	name	useful
--2vR0DIsmQ6WfcSzKWigw	Harald	154202
JjXuiru1_ONzDkYVrHN0aw	Richard	99162
W7DHyQlY_kXls2iXt-_2Ag	Maggie	89792
Hi10sGSZNxQH3NLyWSZ1oA	Fox	89418
ax7SnXOTIpatbsmqHLqVow	Rohlin	81003

- The most **'fans'**

user_id	name	fans
37cpUoM8hlkSQfReIEBd-Q	Mike	9538
hizGc5W1tBHPghM5YKCAtg	Katie	2964
eKUGKQRE-Ywi5dY55_zChg	Cherylynn	2434
iLjMdZi0Tm7DQxX1C1_2dg	Ruggy	2383
j14WgRoU_-2ZE1aw1dXrJg	Daniel	2132

- '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

friends	count
None	370441
wd3xoNaDLib8dhQ7B...	148
Wc5L6iuvSNF5WGBlq...	111
GGTF7hnQi6D5W77_q...	98
WeVkkF5L39888IPPl...	74
u3ZPMVVEzneq8x856...	73
-xDW3gYiYaoeVASXy...	57
O_GWZZfQx7qv-n-CN...	55
lYp818T-xh8Ss79To...	55
6Uup5yoodhI5upHNa...	50
PhUqhfyk3jdaS0Xb6...	48
WwulXySQN8t2hwqH_...	43
ptL7YoBv_zDhzRViT...	42
ZWD8UH1T7QXQr0Eq-...	38
Ryxj0u0AW3mRsRypd...	38
oUK6Xs5dPPnP4whFe...	37
NhgU7RhuYYFmpkb1j...	35
WFZIMqctBkMzzBV6I...	35
PkeDOqXbgEOkR-aKU...	34
NrSURtBigpxbdfL4n...	34

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
RtGqdDBvvBCjcu5dUqwfzA	oUX2bYbqjqST-urKbOHG6w	Loftti Cafe	Las Vegas	Shila	12390

- 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
--2vR0DIsmQ6WfcSzKWigw	uanCi4OGc1mHLGl_AT4JhQ	Treasure Island	Las Vegas	2487	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

- 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
P5bUL3Engv-2z6kKohB6qQ	Kim	9821	3.8
8RcEwGrFIgkt9WQ35E6SnQ	George	7750	3.49

- 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
sE3ge33huDcNJGW3V4obww	Ken	2004-10-12 09:16:01
5iOHz6pHmXi9SoB5qomRWQ	Nader	2004-10-12 17:42:24

Recommendation system

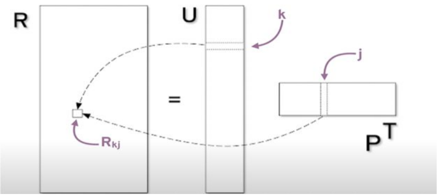
We want to recommend restaurants to users

- 2 main type of recommenders:
  - Content-based recommendations (good to recommend when user has no history)
  - Collaborative filtering (good when user has a history)
- Yelp Dataset:
  - Restaurant data contains very limited information
  - Very few reviews compared to the number of restaurants and user(sparse matrix)
- Chosen approach:
  - We use collaborative filtering through matrix factorization to predict user ratings based on past ratings
  - Use Alternative Least Square (ALS) algorithm in SparkML

ALS Algorithm

- Factorize a ratings matrix R into two latent factor matrices which when multiplied back, will give a approximation of the original ratings matrix. In the approximation matrix, all the cells will be filled by an estimated rating.
- if we want to predict how user K might rate product J we just multiply those two vector together
- Cost function:

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



- It´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 that process alternately.⟶ effective performance
- Using ASL-WR to avoid overfitting

ALS Algorithm

Ratings

	Au Bon Pain	Chipotle	Panda Express
John	2	5	?
Mary	?	4	?
Peter	1	?	5

Restaurants x Users^T

	Au Bon Pain	Chipotle	Panda Express
John	1.88	4.15	4.97
Mary	1.68	3.9	3.9
Peter	1.76	3.9	4.69

Restaurants

	feature1	feature2
Au Bon Pain	1.2	0.8
Chipotle	2.5	2
Panda Express	2.8	2.7

X

Users

T

	feature1	feature2
John	1.1	0.7
Mary	0.6	1.2
Peter	1	0.7

Preparing the data

- Let's consider Toronto
- 7,000 restaurants x 85,000 restaurants = 595,000,000 cells in the utility matrix!
  LOT OF MEMORY AND PROCESSING
- We need to reduce the dataset:
  - Only keep restaurants with over 40 reviews: 2,200 restaurants
  - Only keep users with over 5 reviews: 13,000 users
- 2,200 * 13,000 users = 28,600,000 MORE MANAGEABLE!
- Reviews: 144,621 => sparse = 0.5%

Split training and testing data

To train and evaluate the recommender system, the dataset will be split by time.

Training set: 80%
(before Jun 2017)

Testing set: 20%
(after Jun 2017)

Tuning the recommender

Model = ALS.train(rank=?, lambda=?, iterations=?, alpha = ?, userCol="user_id", itemCol="product_id", ratingCol="frequen

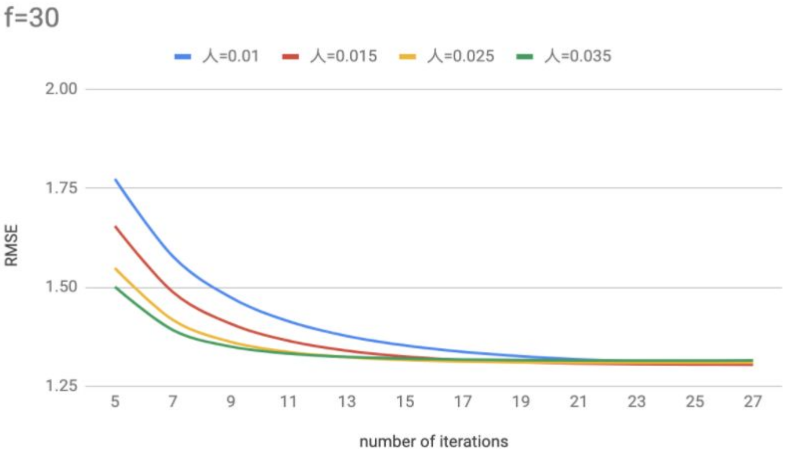
Tuning parameters

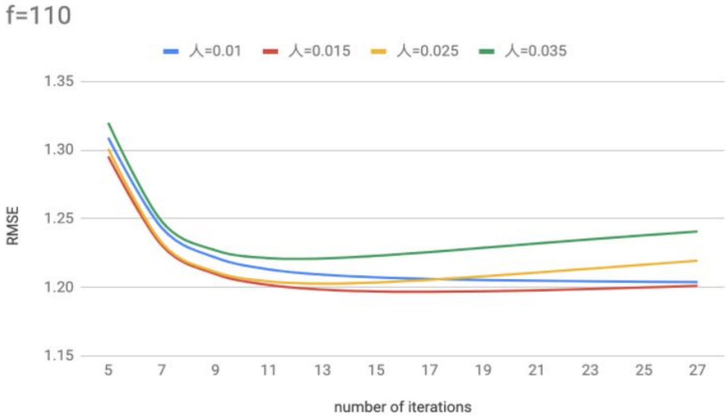
ALS relies on 3 hyperparameters:

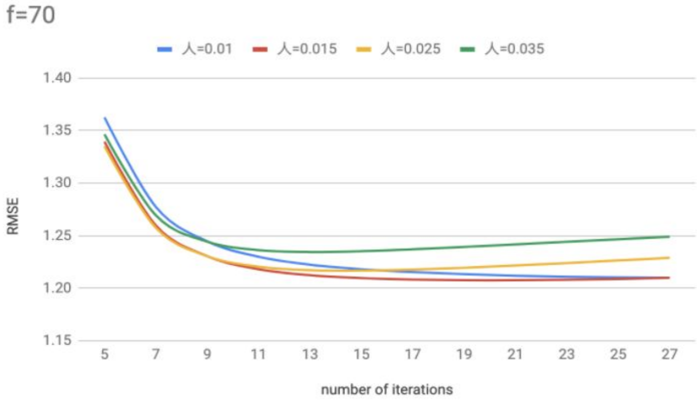
- Rank: Number of latent features
- Iterations: Number of iterations
- Lambda: Regularization parameter

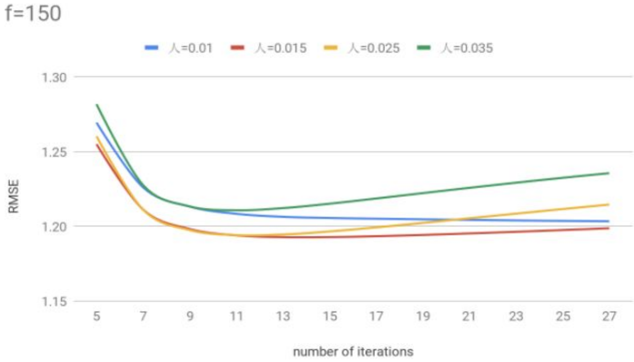
Find hyperparameters values that minimize the RMSE cost function

For tuning, we get a smaller sample from our data, we choose restaurants > 100 reviews (base on the pic), then user >10 reviews. 895 restaurants x 5458 users = 4.8 millions, reviews = 76942 ⇒ sparse = 1.57509555%









We run 420 experiments with different hyperparameters for 10 hours.
The best set of parameter is 150, 0,015, 10

Experiment result

Best Hyper Parameters:

- Rank=150
- Iterations=10
- Lamba=0.015

We use those parameters to train the full dataset for each city: Take 10 minutes per city

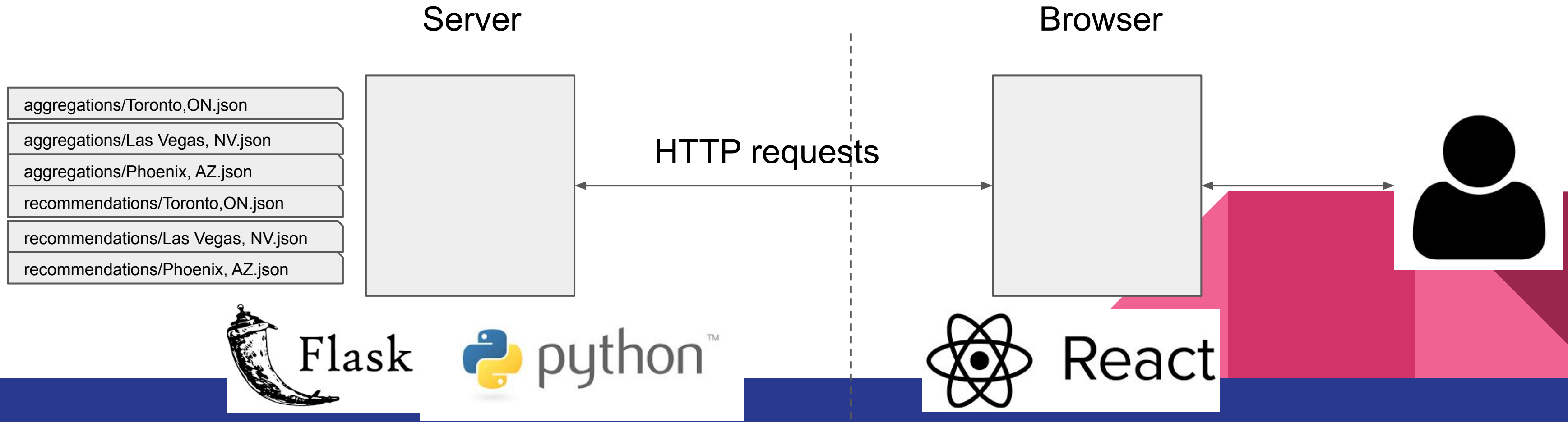
For Toronto, RMSE = 1.40

RMSE for other cities:

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

Demo

- Front-end:
    - React Framework: Popular Javascript Framework
    - Styling: Materialize
    - Map Library: Leaflet
    - Chart Library: e-chart
- Backend:
    - Flask: Popular Python Framework for Web Services



Recommendation

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 & Tea	2
Vegetarian	2
Indian	1
African	1
Spanish	1
Pakistani	1
Mediterranean	1

Future works

- Try with larger data using a large cluster of machines
- Influencer: use the pagerank algorithm and other graph algorithms and visualization tools
- For recommendation: Show recommendation to real users and evaluate performance using A/B testing and also try other models: hybrid, neural network (RBM)
- For web application: Use database as backend and putting more features

Questions?