Restaurant Recommendations

Ryan Chan

Notes:

- This project is an exploratory work for my own learning and is not suitable for use in a production environment.
- The dataset is too large to include in an upload of this project. You can find and download the Yelp Dataset at https://www.yelp.com/dataset. Place all dataset json files into the folder 'yelp_dataset'

In [2]:
```
import pandas as pd
import json
import numpy as np
from sklearn.linear_model import LinearRegression
```

In [3]:
```
df_b = pd.read_json("yelp_dataset/yelp_academic_dataset_business.json", lines
```

In [187…
```
display(df_b.head(3))
```

	business_id	name	address	city	state	postal_code	latitud
0	f9NumwFMBDn751xgFiRbNA	The Range At Lake Norman	10913 Bailey Rd	Cornelius	NC	28031	35.46272
1	Yzvjg0SayhoZgCljUJRF9Q	Carlos Santo, NMD	8880 E Via Linda, Ste 107	Scottsdale	AZ	85258	33.56940
2	XNoUzKckATkOD1hP6vghZg	Felinus	3554 Rue Notre-Dame O	Montreal	QC	H4C 1P4	45.47998

In [5]:
```
df_u = pd.read_json("yelp_dataset/yelp_academic_dataset_user.json", lines=Tru
```

In [6]:
```
TRAINING_REVIEW_COUNT_THRESHOLD_ = 250
df_u_abbrev = df_u[df_u["review_count"]>TRAINING_REVIEW_COUNT_THRESHOLD_]
```

In [188…
```
display(df_u_abbrev.head(5))
```

	user_id	name	review_count	yelping_since	useful	funny	coo
0	ntlvfPzc8eglqvk92iDIAw	Rafael	553	2007-07-06 03:27:11	628	225	22

	user_id	name	review_count	yelping_since	useful	funny	coo
1	FOBRPlBHa3WPHFB5qYDlVg	Michelle	564	2008-04-28 01:29:25	790	316	40
4	xvu8G900tezTzbbfqmTKvA	Anne	485	2008-08-09 00:30:27	1265	400	51
7	f4_MRNHvN-yRn7EA8YWRxg	Jennifer	822	2011-01-17 00:18:23	4127	2446	287
11	l_6wY8_RsewziNnKhGZg4g	Jeff	405	2010-08-05 18:42:29	799	244	31

5 rows × 22 columns

In [9]:
```python
df_r = pd.read_json("yelp_dataset/yelp_academic_dataset_review.json", lines=T
```

In [10]:
```python
df_r = df_r.drop(labels=["useful", "funny", "cool", "text", "date"], axis=1)
```

In [189…
```python
display(df_r.head(5))
```

	review_id	user_id	business_id	sta
0	xQY8N_XvtGbearJ5X4QryQ	OwjRMXRC0KyPrlIcjaXeFQ	-MhfebM0QIsKt87iDN-FNw	
1	UmFMZ8PyXZTY2QcwzsfQYA	nIJD_7ZXHq-FX8byPMOkMQ	lbrU8StCq3yDfr-QMnGrmQ	
2	LG2ZaYiOgpr2DK_90pYjNw	V34qejxNsCbcgD8C0HVk-Q	HQl28KMwrEKHqhFrrDqVNQ	
3	i6g_oA9Yf9Y31qt0wibXpw	ofKDkJKXSKZXu5xJNGiiBQ	5JxlZaqCnk1MnbgRirs40Q	
4	6TdNDKywdbjoTkizeMce8A	UgMW8bLE0QMJDCkQ1Ax5Mg	IS4cv902ykd8wj1TR0N3-A	

Create business feature vectors, place into a dictionary

In [14]:
```python
#df_b[df_b["attributes"].map(lambda x: "Alcohol" in x)]
#set(df_b["attributes"].map(lambda x: x["RestaurantsAttire"] if isinstance(x,
```

In [15]:
```python
featureAttributesBool = [
    "RestaurantsPriceRange2",
    "RestaurantsReservations",
    "BikeParking",
    "BusinessAcceptsCreditCards",
    "ByAppointmentOnly",
    "DriveThru",
    "GoodForKids",
    'HappyHour',
    "HasTV",
    "OutdoorSeating",
    "RestaurantsDelivery",
    "RestaurantsGoodForGroups",
    "RestaurantsTakeOut",
    "WheelchairAccessible"
]

featureAttributesOneHot = [
    ['Alcohol', "None", "'none'", "u'none'"],
    ['Alcohol', "'beer_and_wine'", "u'beer_and_wine'"],
    ['Alcohol', "full_bar", "u'full_bar'"],
    ['RestaurantsAttire', "'None'"],
    ['RestaurantsAttire', "'casual'", "u'casual'"],
    ['RestaurantsAttire', "'dressy'", "u'dressy'"],
    ['RestaurantsAttire', "'formal'", "u'formal'"],
]

def extractAttributes(businessAttributes):
    features = list()
    for attr in featureAttributesBool:
        if isinstance(attr, str):
            if businessAttributes is not None and attr in businessAttributes:
                features.append(int(businessAttributes[attr] == 'True'))
            else:
                features.append(0)

    for attr in featureAttributesOneHot:
        if businessAttributes is not None and attr[0] in businessAttributes:
            found = False
            for attrTag in attr[1:]:
                if businessAttributes[attr[0]] == attrTag:
                    found = True
                    features.append(1)
                    break
            if not found:
                features.append(0)
        else:
            features.append(0)
    return features
```

In [16]:

```
businessFeatureDict = dict()

def addBusiness(business):
    features = [business["stars"]] + [business["review_count"]] + extractAttr
    businessFeatureDict[business["business_id"]] = features
```

In [17]:

```
df_b.apply(addBusiness, axis=1)
```

Out[17]:

```
0         None
1         None
2         None
3         None
4         None
          ...
209388    None
209389    None
209390    None
209391    None
209392    None
Length: 209393, dtype: object
```

In [19]:

```
#addBusiness(df_b.iloc[209388])
#print(businessFeatureDict["SYa2j1boLF8DcGVOYfHPcA"])
```

Testing on a random business

In [20]:

```
a = extractAttributes(df_b[df_b["business_id"]=="SYa2j1boLF8DcGVOYfHPcA"]["at
```

In [21]:

```
a
```

Out[21]:
```
[0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0]
```

Weight Generation

In [22]:
```python
def generateWeights(user_id):
    user_reviews = df_r[df_r["user_id"]==user_id]
    user_review_scores = user_reviews["stars"].values
    #business_features = user_reviews.apply(lambda x: businessFeatureDict[x['
    business_features = np.array(list(map(lambda x: businessFeatureDict[x], u
    #display(business_features)
    model = LinearRegression()
    model.fit(business_features, user_review_scores)
    return model.coef_
```

In [24]:
```python
weights = df_u_abbrev.apply(lambda x: generateWeights(x["user_id"]), axis=1)
```

In [192…
```python
display(weights.head(5))
print("Number of points: " + str(len(df_u_abbrev["user_id"].values)))
```

```
0     [1.0741416610495274, 0.0001243782230512169, 2....
1     [0.36931201740440167, 0.00036807714185722526, ...
4     [0.11302586666171928, 0.0002609166948490038, 0...
7     [0.5066197242566907, 0.0002036361778916583, -6...
11    [-1.9369369369368967, -0.00450450450450395, -9...
dtype: object
Number of points: 25763
```

Format weights to be acceptable input for Milvus

In [50]:
```python
weightsnp = np.stack(weights.values)
```

In [52]:
```python
weightsnp.shape
```

Out[52]:
```
(25763, 23)
```

Locally hosted Milvus instance, partially taken from https://raw.githubusercontent.com/milvus-io/pymilvus/0.2.15/examples/example.py

In [101…
```python
from milvus import Milvus, IndexType, MetricType, Status

# Milvus server IP address and port.
# You may need to change _HOST and _PORT accordingly.
_HOST = '127.0.0.1'
_PORT = '19530'  # default value
# _PORT = '19121'  # default http value

# Vector parameters
_DIM = 23  # dimension of vector

_INDEX_FILE_SIZE = 32  # max file size of stored index
```

In [102…
```python
milvus = Milvus(_HOST, _PORT)

# Create collection demo_collection if it dosen't exist.
collection_name = 'recomendation_collection_'
```

```
status, ok = milvus.has_collection(collection_name)
if not ok:
    param = {
        'collection_name': collection_name,
        'dimension': _DIM,
        'index_file_size': _INDEX_FILE_SIZE,  # optional
        'metric_type': MetricType.L2  # optional
    }

    milvus.create_collection(param)

# Show collections in Milvus server
_, collections = milvus.list_collections()

# Describe demo_collection
_, collection = milvus.get_collection_info(collection_name)
print(collection)
```

```
CollectionSchema(collection_name='recomendation_collection_', dimension=23, i
ndex_file_size=32, metric_type=<MetricType: L2>)
```

In [104…

```
#milvus.drop_collection(collection_name)
```

Retype ids into native python int types. Milvus does not seem to accept ids from numpy types, such as numpy.int64 or numpy.int32.

In [103…

```
weightindex = weights.index.values
weightids = [weightindex.item(i) for i in range(len(weightindex))]
```

In [105…

```
# Insert vectors into demo_collection, return status and vectors id list
status, ids = milvus.insert(collection_name=collection_name, records=np.stack
if not status.OK():
    print("Insert failed: {}".format(status))

# Flush collection  inserted data to disk.
milvus.flush([collection_name])
```

Out[105…

```
Status(code=0, message='OK')
```

In [193…

```
# Get collection row count
status, result = milvus.count_entities(collection_name)
print("Row count: " + str(result))
```

```
Row count: 25763
```

Build Index

We use the recommended value of nlist: 4 × sqrt(n), where n is the number of items in a segment

In [107…

```
_, info = milvus.get_collection_stats(collection_name)
print(info)
```

```
{'partitions': [{'row_count': 25763, 'segments': [{'data_size': 2576300, 'ind
ex_name': 'IDMAP', 'name': '1613784371630425000', 'row_count': 25763}], 'ta
g': '_default'}], 'row_count': 25763}
```

In [109…

```
25763 ** (1/2) * 4
```

Out[109…

```
642.0342669982655
```

In [112…

```
ivf_param = {'nlist': 642}
milvus.create_index(collection_name, IndexType.IVF_SQ8, ivf_param)
```

Out[112…

```
Status(code=0, message='Build index successfully!')
```

nprobe is chosen semi-arbitrarially

In [194…

```
search_param = {'nprobe': 16}
```

In [195…

```
NUM_USERS_REC_ = 3
NUM_REC_PER_USER_ = 3

def retrieveRecs(uid, num_recs):
    user_id = df_u_abbrev.iloc[uid]["user_id"]
    toprecs = df_r[df_r["user_id"] == user_id].sort_values(by="stars", ascend
    return df_b[df_b["business_id"].isin(toprecs.head(num_recs)["business_id'

def generateRecommendations(user_id):
    user_weights = np.stack(generateWeights(user_id)).reshape(1, 23)
    status, results = milvus.search(collection_name=collection_name, query_re
    print(status)
    rec_df = pd.DataFrame()
    for searchResult in results[0]:
        rec_df = rec_df.append(retrieveRecs(searchResult.id, NUM_REC_PER_USEF
    return rec_df
```

Running

To generate a set of reccomendations for a user, run `generateRecommendations(user_id)` , where user_id is their id. This assumes that the user is present in the dataset. It will generate and output `NUM_USERS_REC_` * `NUM_REC_PER_USER_` recommended businesses.

Example with a randomly selected user:

In [196…

```
generateRecommendations("fL0jIsxSR2DSBeIRI8OcTA")
```

```
Status(code=0, message='Search vectors successfully!')
```

Out[196…

	business_id	name	address	city	state	postal_cod
148713	-Miw03v5yXJWjH9MN1aglw	The Nash	925 11th Street SE	Calgary	AB	T2G 0S
172724	bcCfoAUpHY5SVrYd4alasA	Village Ice Cream	2406 34th Avenue SW	Calgary	AB	T2T 2C
205504	TSzsZRpN09mqu54HWNn1PA	Mari Bakeshop	529 Riverfront Avenue SE	Calgary	AB	T2G 1K

	business_id	name	address	city	state	postal_cod
5257	Ul6JwluSTm12PVDIqnNaTg	Kaya	2000 Smallman St	Pittsburgh	PA	1522
46641	TrCiLMGy_bxbeAQcuSSUeQ	Larry & Carols Pizza	410 Semple St	Pittsburgh	PA	1521
175195	dIsUtYng6lzaaLOqHlkOMA	Jerry's Records	2136 Murray Ave	Pittsburgh	PA	1521
8892	MN6HfA76VrdU4RjiGLwSug	A Elvis Chapel	727 S 9th St	Las Vegas	NV	8910
82036	Bkkwt8E9MHvgCHn4lUFtow	Roy's Restaurant	5350 E Marriott Dr	Phoenix	AZ	8505
198269	dUffgo9Lh_Vk9TLuFR5ywg	Oregano's Pizza Bistro	1008 E Camelback Rd	Phoenix	AZ	8501