Yelp_Dataset_-_Restaurant_Recommender_new

November 13, 2018

1 Yelp Data Challenge - Restaurant Recommender

BitTiger DS501
Nov 2018

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        % matplotlib inline
        plt.style.use("ggplot")

In [2]: df = pd.read_csv('data/last_2_years_restaurant_reviews.csv')

In [3]: df.head()

Out[3]:               business_id                  name  \
        0  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse
        1  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse
        2  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse
        3  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse
        4  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse

                                      categories  avg_stars  cool        date  \
        0  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2015-06-26
        1  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2015-06-29
        2  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2015-04-05
        3  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2016-02-16
        4  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2016-02-08

           funny               review_id  stars  \
        0      0  nCqdz-NW64KazpxqnDr0sQ      1
        1      0  iwx6s6yQxc7yjS7NFANZig      4
        2      0  2HrBENXZTiitcCJfzkELgA      2
        3      0  6YNPXoq41qTMZ2TEiOBYUA      2
        4      1  4bQrVUiRZ642odcKCS0OhQ      2

                                                        text    type  useful  \
```

```
0  I mainly went for the ceasar salad prepared ta...  review      0
1  Nice atmosphere and wonderful service. I had t...  review      0
2  To be honest it really quit aweful. First the ...  review      0
3  The food was decent, but the service was defin...  review      0
4  If you're looking for craptastic service and m...  review      1

                  user_id  count
0  0XVzm4kVIAaH4eQAxWbhvw    318
1  2aeNFntqY2QDZLADNo8iQQ    318
2  WFhv5pMJRDPWSyLnKiWFXA    318
3  2S6gWE-K3DHNcKYYSgN7xA    318
4  rCTVWx_Tws2jWi-K89iEyw    318
```

1.1 1. Clean data and get rating data

Select relevant columns in the original dataframe

```
In [99]: # Get business_id, user_id, stars for recommender

         df_select = df[['business_id', 'user_id', 'stars']]
         df_select.head(2)
```

```
Out[99]:               business_id                 user_id  stars
         0  --9e1ONYQuAa-CB_Rrw7Tw  0XVzm4kVIAaH4eQAxWbhvw      1
         1  --9e1ONYQuAa-CB_Rrw7Tw  2aeNFntqY2QDZLADNo8iQQ      4
```

There are many users that haven't given many reviews, exclude these users from the item-item similarity recommender **Q**: How do we recommend to these users anyways?

```
In [100]: # To be implemented
          Review_Num_Limit = 3
          df_user_count = df_select.copy()
          df_user_count['count'] = 1
          df_user_count.head(2)
          df_user_count = df_user_count.groupby('user_id')[['count']].sum()
          df_user_count = df_user_count[df_user_count['count'] > Review_Num_Limit]
          df_merge = pd.merge(df_select, df_user_count, how = 'inner', left_on='user_id', righ
          df_merge = df_merge.reset_index().drop('index', axis = 1)
          df_merge.head(2)
```

```
Out[100]:               business_id                 user_id  stars  count
          0  --9e1ONYQuAa-CB_Rrw7Tw  0XVzm4kVIAaH4eQAxWbhvw      1     16
          1  2iTsRqUsPGRH1li1WVRvKQ  0XVzm4kVIAaH4eQAxWbhvw      4     16
```

Create utility matrix from records

```
In [101]: df_data = df_merge.drop('count', axis = 1)
          df_data.head(2)
```

```
Out[101]:              business_id                 user_id  stars
          0  --9e1ONYQuAa-CB_Rrw7Tw  0XVzm4kVIAaH4eQAxWbhvw      1
          1  2iTsRqUsPGRH1li1WVRvKQ  0XVzm4kVIAaH4eQAxWbhvw      4
```

```
In [102]: # reconstruct business id
          unique_business_id = df_data['business_id'].unique()
          business_shape = unique_business_id.shape
          print('Number of Unique Business ID: %d' % business_shape[0])
          business_df = pd.DataFrame({
                  'business_id' :unique_business_id,
                  'business_index': xrange(business_shape[0])
              })
          business_df.head(2)
```

```
Number of Unique Business ID: 4138
```

```
Out[102]:              business_id  business_index
          0  --9e1ONYQuAa-CB_Rrw7Tw               0
          1  2iTsRqUsPGRH1li1WVRvKQ               1
```

```
In [103]: # reconstruct user id
          unique_users_id = df_data['user_id'].unique()
          user_shape = unique_users_id.shape
          print('Number of Unique User ID: %d' % user_shape[0])
          user_df = pd.DataFrame({
                  'user_id': unique_users_id,
                  'user_index': xrange(user_shape[0])
              })
          user_df.head(2)
```

```
Number of Unique User ID: 18901
```

```
Out[103]:                  user_id  user_index
          0  0XVzm4kVIAaH4eQAxWbhvw           0
          1  rCTVWx_Tws2jWi-K89iEyw           1
```

```
In [104]: # inner join the business_df and df_data
          df_data = pd.merge(df_data, business_df, how = 'inner', left_on='business_id', right_
          df_data.head(2)
```

```
Out[104]:              business_id                 user_id  stars  business_index
          0  --9e1ONYQuAa-CB_Rrw7Tw  0XVzm4kVIAaH4eQAxWbhvw      1               0
          1  --9e1ONYQuAa-CB_Rrw7Tw  rCTVWx_Tws2jWi-K89iEyw      2               0
```

```
In [105]: df_data = pd.merge(df_data, user_df, how = 'inner', left_on='user_id', right_on = 'us
          df_data.head(2)
```

Out[105]:
```
                business_id                user_id  stars  business_index  \
0  --9e1ONYQuAa-CB_Rrw7Tw  OXVzm4kVIAaH4eQAxWbhvw      1               0
1  2iTsRqUsPGRH1li1WVRvKQ  OXVzm4kVIAaH4eQAxWbhvw      4               1

   user_index
0           0
1           0
```

In [118]:
```
# Utility matrix can be used with sparse matrix
from scipy import sparse
from sklearn.metrics.pairwise import cosine_similarity
import random
from time import time
```

In [107]:
```
# construct sparse matrix
highest_user_id = unique_users_id.shape[0]
highest_business_id = unique_business_id.shape[0]
ratings_mat = sparse.lil_matrix((highest_user_id, highest_business_id))
ratings_mat
```

Out[107]:
```
<18901x4138 sparse matrix of type '<type 'numpy.float64'>'
        with 0 stored elements in LInked List format>
```

In [108]:
```
for _, row in df_data.iterrows():
    # subtract 1 from id's due to match 0 indexing
    ratings_mat[row.user_index, row.business_index] = row.stars
```

1.2 2. Item-Item similarity recommender

1.2.1 Let's reuse the ItemItemRecommender class derived from previous exercise

Hint: we need to make modification to accommodate the dense numpy array

In [126]:
```
class ItemItemRecommender(object):

    def __init__(self, neighborhood_size):
        self.neighborhood_size = neighborhood_size

    def fit(self, ratings_mat):
        self.ratings_mat = ratings_mat
        self.n_users = ratings_mat.shape[0]
        self.n_items = ratings_mat.shape[1]
        self.item_sim_mat = cosine_similarity(self.ratings_mat.T)
        self._set_neighborhoods()

    def _set_neighborhoods(self):
        least_to_most_sim_indexes = np.argsort(self.item_sim_mat, 1)
        self.neighborhoods = least_to_most_sim_indexes[:, -self.neighborhood_size:]
```

```
    def pred_one_user(self, user_id, report_run_time=False):
        start_time = time()
        items_rated_by_this_user = self.ratings_mat[user_id].nonzero()[1]
        # Just initializing so we have somewhere to put rating preds
        out = np.zeros(self.n_items)
        for item_to_rate in range(self.n_items):
            relevant_items = np.intersect1d(self.neighborhoods[item_to_rate],
                                            items_rated_by_this_user,
                                            assume_unique=True)  # assume_unique spe
            out[item_to_rate] = self.ratings_mat[user_id, relevant_items] * \
                self.item_sim_mat[item_to_rate, relevant_items] / \
                self.item_sim_mat[item_to_rate, relevant_items].sum()
        if report_run_time:
            print("Execution time: %f seconds" % (time()-start_time))
        cleaned_out = np.nan_to_num(out)
        return cleaned_out

    def pred_all_users(self, report_run_time=False):
        start_time = time()
        all_ratings = [
            self.pred_one_user(user_id) for user_id in range(self.n_users)]
        if report_run_time:
            print("Execution time: %f seconds" % (time()-start_time))
        return np.array(all_ratings)

    def top_n_recs(self, user_id, n, verbose = False):
        pred_ratings = self.pred_one_user(user_id, report_run_time = verbose)
        item_index_sorted_by_pred_rating = list(np.argsort(pred_ratings))
        items_rated_by_this_user = self.ratings_mat[user_id].nonzero()[1]
        unrated_items_by_pred_rating = [item for item in item_index_sorted_by_pred_ra
                                        if item not in items_rated_by_this_user]
        return unrated_items_by_pred_rating[-n:]
```

```
In [134]: Top_N = 5
          recommender = ItemItemRecommender(neighborhood_size=100)
          recommender.fit(ratings_mat)
          lucky = random.randint(0, highest_user_id - 1)
          print('User Chosen is: %d' % lucky)
          result = recommender.top_n_recs(lucky, Top_N, verbose = True)
          print(result)
```

```
User Chosen is: 6730


/usr/local/lib/python2.7/site-packages/ipykernel_launcher.py:26: RuntimeWarning: invalid value


Execution time: 0.903347 seconds
[2840, 2738, 2610, 1885, 2171]
```

```
In [140]: # convert into valid business entity
          def printBusiness(result_list, unique_user_id):
              business_list = []
              for index in result:
                  business_list.append(unique_users_id[index])
              return business_list

In [141]: convert_result = printBusiness(result, unique_users_id)
          print(convert_result)
```

```
['g4NQ8YTArwKlCPQx2_GDVw', 'soaoXgD4VKTPxy4Dz8DvZw', '8OjhIQZAOBCEvMhHCnn2pw', 'D43OWyfzIQjL8f
```

```
In [148]: # save the data_df to csv
          df_data.to_csv('cleaned_data.csv')

In [149]: # print out results for all users
          Top_N = 5
          '''
          all_result = [recommender.top_n_recs(index, Top_N) for index in xrange(user_shape[0]
          convert_all_result = [printBusiness(result_list, unique_users_id) for result_list in
          print(convert_all_result)
          '''

Out[149]: '\nall_result = [recommender.top_n_recs(index, Top_N) for index in xrange(user_shape
```

1.3 3. Matrix Factorization recommender

Take a look at Graphlab Create examples

```
In [56]: # Using GraphLab Create APIs
         import graphlab as gl
         sf = gl.SFrame.read_csv('cleaned_data.csv')
```

```
Finished parsing file /Users/Peter/Desktop/5000/Data Science/Projects/Python/Yelp/cleaned_data


Parsing completed. Parsed 100 lines in 0.17552 secs.


------------------------------------------------------
Inferred types from first 100 line(s) of file as
column_type_hints=[int,str,str,int,int,int]
If parsing fails due to incorrect types, you can correct
the inferred type list above and pass it to read_csv in
the column_type_hints argument
------------------------------------------------------


Finished parsing file /Users/Peter/Desktop/5000/Data Science/Projects/Python/Yelp/cleaned_data
```

```
Parsing completed. Parsed 157275 lines in 0.196044 secs.
```

```
In [66]: # choose only related information: business_index, user_index, stars (ratings)
         data = sf[['business_index', 'user_index', 'stars']]
         (train_set, test_set) = data.random_split(0.8)

         kfolds = gl.cross_validation.KFold(train_set, 5)
         seed = 42
         data.head(2)
```

```
Out[66]: Columns:
                 business_index        int
                 user_index        int
                 stars        int

         Rows: 2

         Data:
         +----------------+------------+-------+
         | business_index | user_index | stars |
         +----------------+------------+-------+
         |       0        |     0      |   1   |
         |       1        |     0      |   4   |
         +----------------+------------+-------+
         [2 rows x 3 columns]
```

```
In [67]: params = dict(user_id='user_index',
                       item_id='business_index',
                       target='stars',
                       random_seed = seed)
         factor_search = gl.model_parameter_search.create(kfolds,
                                                          gl.factorization_recommender.create,
                                                          params, max_models = 100)
```

```
[INFO] graphlab.deploy.job: Validating job.
[INFO] graphlab.deploy.map_job: Validation complete. Job: 'Model-Parameter-Search-Jun-28-2017-1
[INFO] graphlab.deploy.map_job: Job: 'Model-Parameter-Search-Jun-28-2017-14-35-2200000' schedul
[INFO] graphlab.deploy.job: Validating job.
[INFO] graphlab.deploy.map_job: A job with name 'Model-Parameter-Search-Jun-28-2017-14-35-2200(
[INFO] graphlab.deploy.map_job: Validation complete. Job: 'Model-Parameter-Search-Jun-28-2017-1
[INFO] graphlab.deploy.map_job: Job: 'Model-Parameter-Search-Jun-28-2017-14-35-2200000-974ea' s
[INFO] graphlab.deploy.job: Validating job.
[INFO] graphlab.deploy.map_job: Validation complete. Job: 'Model-Parameter-Search-Jun-28-2017-1
[INFO] graphlab.deploy.map_job: Job: 'Model-Parameter-Search-Jun-28-2017-14-35-2200001' schedul
[INFO] graphlab.deploy.job: Validating job.
[INFO] graphlab.deploy.map_job: Validation complete. Job: 'Model-Parameter-Search-Jun-28-2017-1
[INFO] graphlab.deploy.map_job: Job: 'Model-Parameter-Search-Jun-28-2017-14-35-2200002' schedul
```

In [68]:
```python
from pprint import pprint
factor_search.get_results()
```

Out[68]:
```
Columns:
        item_id        str
        linear_regularization        float
        max_iterations        int
        num_factors        int
        random_seed        int
        regularization        float
        target        str
        user_id        str
        model_id        list
        mean_validation_rmse        float
        mean_training_rmse        float
        mean_training_recall@5        float
        mean_validation_recall@5        float
        fold_id        list
        mean_validation_precision@5        float
        num_folds        int
        mean_training_precision@5        float

Rows: 69

Data:
+----------------+-----------------------+----------------+-------------+-------------
|    item_id     | linear_regularization | max_iterations | num_factors | random_seed
+----------------+-----------------------+----------------+-------------+-------------
| business_index |         1e-09         |       25       |      8      |      42
| business_index |         1e-09         |       50       |      64     |      42
| business_index |         1e-07         |       50       |      8      |      42
| business_index |         1e-09         |       25       |      32     |      42
| business_index |         1e-05         |       50       |      8      |      42
| business_index |         1e-09         |       25       |      64     |      42
| business_index |         1e-05         |       50       |      8      |      42
| business_index |         1e-07         |       25       |      64     |      42
| business_index |         1e-05         |       25       |      16     |      42
| business_index |         1e-09         |       25       |      64     |      42
+----------------+-----------------------+----------------+-------------+-------------
+----------------+--------+------------+-------------------------------+
| regularization | target |  user_id   |            model_id           |
+----------------+--------+------------+-------------------------------+
|     1e-07      | stars  | user_index |   [290, 291, 292, 293, 294]   |
|     1e-06      | stars  | user_index |   [409, 408, 407, 406, 405]   |
|     1e-07      | stars  | user_index |   [473, 472, 471, 470, 474]   |
|     1e-08      | stars  | user_index | [15, 17, 16, 19, 18, 484, ... |
|     1e-08      | stars  | user_index | [433, 432, 431, 430, 434, ... |
|     1e-09      | stars  | user_index | [91, 90, 93, 92, 94, 165, ... |
```

1e-07	stars	user_index	[9, 8, 5, 7, 6]
1e-08	stars	user_index	[358, 359, 355, 356, 357, ...
0.0001	stars	user_index	[402, 403, 400, 401, 411, ...
1e-08	stars	user_index	[498, 499, 495, 496, 497]

mean_validation_rmse	mean_training_rmse	mean_training_recall@5
1.21000442731	0.349043145332	0.00304379724192
1.20698228395	0.187415744686	0.0262731430746
1.212304989	0.288748729041	0.00190307549683
1.20603475576	0.480869995251	0.0190021214833
1.20680587588	0.290992601683	0.00185621509995
1.20597506597	0.496935979401	0.0191874623617
1.20588373403	0.286455317367	0.00185641405795
1.20595425725	0.496105314729	0.0193201088184
1.20089243683	0.975904476767	0.00071594537358
1.20597363622	0.496387886452	0.0191983071499

mean_validation_recall@5	fold_id	mean_validation_precisi
9.35918700305e-05	[0, 1, 2, 3, 4]	0.000156478491448
9.91810013845e-05	[4, 3, 2, 1, 0]	0.000140612245402
0.000140201691497	[3, 2, 1, 0, 4]	0.000255246113485
0.000130289833258	[0, 2, 1, 4, 3, 4, 2, 3, 0, 1]	0.000191549709322
0.00221029810349	[3, 2, 1, 0, 4, 4, 1, 0, 3, 2]	0.00361635407445
0.000130242753717	[1, 0, 3, 2, 4, 0, 1, 4, 3...	0.000182820901337
0.000718676882049	[4, 3, 0, 2, 1]	0.0018521308974
0.000127694959225	[3, 4, 0, 1, 2, 4, 3, 0, 2...	0.000184513417297
0.000309262126019	[2, 3, 0, 1, 1, 0, 4, 3, 2, 4]	0.000546277422622
0.000122505211161	[3, 4, 0, 1, 2]	0.000170440833247

num_folds	mean_training_precision@5
5	0.0030404243426
5	0.0266615364012
5	0.00173512603166
10	0.0212451515421
10	0.00170784017037
15	0.0227369620814
5	0.00169197204904
15	0.0228279228365
10	0.00117536789045
5	0.0225521532078

[69 rows x 17 columns]

```
Note: Only the head of the SFrame is printed.
You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.
```

In [69]:
```
factor_best_parameter = factor_search.get_best_params('mean_validation_rmse')
factor_best_parameter
```

Out[69]:
```
{'item_id': 'business_index',
 'linear_regularization': 1e-09,
 'max_iterations': 50,
 'num_factors': 8,
 'random_seed': 42,
 'regularization': 1e-08,
 'target': 'stars',
 'user_id': 'user_index'}
```

In [70]:
```
best_factor_model = gl.factorization_recommender.create(test_set, user_id = 'user_inde
                                                         target = 'stars', regularizati
```

```
Recsys training: model = factorization_recommender


Preparing data set.


    Data has 31344 observations with 13924 users and 3377 items.


    Data prepared in: 0.09205s


Training factorization_recommender for recommendations.


+--------------------------------+--------------------------------------------------+----------


| Parameter                      | Description                                      | Value


+--------------------------------+--------------------------------------------------+----------


| num_factors                    | Factor Dimension                                 | 8


| regularization                 | L2 Regularization on Factors                     | 1e-09


| solver                         | Solver used for training                         | sgd
```

```
| linear_regularization          | L2 Regularization on Linear Coefficients          | 1e-09

| max_iterations                 | Maximum Number of Iterations                      | 50

+--------------------------------+---------------------------------------------------+----------

  Optimizing model using SGD; tuning step size.

  Using 10000 / 31344 points for tuning the step size.

+---------+-------------------+------------------------------------------+

| Attempt | Initial Step Size | Estimated Objective Value                |

+---------+-------------------+------------------------------------------+

| 0       | 25                | Not Viable                               |

| 1       | 6.25              | Not Viable                               |

| 2       | 1.5625            | Not Viable                               |

| 3       | 0.390625          | 0.158034                                 |

| 4       | 0.195312          | 0.226432                                 |

| 5       | 0.0976562         | 0.417172                                 |

+---------+-------------------+------------------------------------------+

| Final   | 0.390625          | 0.158034                                 |

+---------+-------------------+------------------------------------------+
```

Starting Optimization.

Iter.	Elapsed Time	Approx. Objective	Approx. Training RMSE	Step Size
Initial	106us	1.62192	1.27355	
1	116.225ms	2.92931	1.71138	0.390625
2	254.468ms	DIVERGED	DIVERGED	0.232267
RESET	268.624ms	1.62191	1.27354	
1	402.339ms	1.59511	1.26296	0.116134
2	532.457ms	1.01107	1.0055	0.0690534
3	685.504ms	0.84785	0.920778	0.0509468
4	838.963ms	0.77127	0.878214	0.0410594
5	966.829ms	0.727464	0.852911	0.034732
8	1.33s	0.658249	0.81132	0.0244141
10	1.49s	0.634082	0.796287	0.0206518
18	2.47s	0.567593	0.75338	0.0132893

```
| 28      | 3.68s        | 0.474031         | 0.68849              | 0.00954089  |

| 38      | 4.92s        | 0.37032          | 0.608528             | 0.00758787  |

| 48      | 6.14s        | 0.278263         | 0.527492             | 0.00636835  |

| 50      | 6.47s        | 0.262403         | 0.512238             | 0.00617632  |

+---------+--------------+------------------+----------------------+-------------+

Optimization Complete: Maximum number of passes through the data reached.

Computing final objective value and training RMSE.

       Final objective value: 0.249954

       Final training RMSE: 0.499939
```

1.4 4. Other recommenders (optional)

What are other ways you can build a better recommender?

- Other features (have you noticed there are other features in the Yelp dataset, e.g. tips, etc.?)
- Popularity-based
- Content-based
- Hybrid

In [193]:
```
df.head(5)
```

Out[193]:
```
              business_id                  name  \
0  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse
1  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse
2  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse
3  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse
4  --9e1ONYQuAa-CB_Rrw7Tw  Delmonico Steakhouse

                                categories  avg_stars  cool        date  \
0  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2015-06-26
1  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2015-06-29
2  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2015-04-05
3  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2016-02-16
```

```
4  [Steakhouses, Restaurants, Cajun/Creole]        4.0     0  2016-02-08

   funny               review_id  stars  \
0      0  nCqdz-NW64KazpxqnDr0sQ      1
1      0  iwx6s6yQxc7yjS7NFANZig      4
2      0  2HrBENXZTiitcCJfzkELgA      2
3      0  6YNPXoq41qTMZ2TEi0BYUA      2
4      1  4bQrVUiRZ642odcKCS0OhQ      2

                                                text    type  useful  \
0  I mainly went for the ceasar salad prepared ta...  review       0
1  Nice atmosphere and wonderful service. I had t...  review       0
2  To be honest it really quit aweful. First the ...  review       0
3  The food was decent, but the service was defin...  review       0
4  If you're looking for craptastic service and m...  review       1

                  user_id  count
0  0XVzm4kVIAaH4eQAxWbhvw    318
1  2aeNFntqY2QDZLADNo8iQQ    318
2  WFhv5pMJRDPWSyLnKiWFXA    318
3  2S6gWE-K3DHNcKYYSgN7xA    318
4  rCTVWx_Tws2jWi-K89iEyw    318
```

1.4.1 When it comes to consider popularity, it will usually related to 'stars'.

In [203]:
```
# Popularity-based
popularity_model = gl.recommender.popularity_recommender.create(train_set, user_id='u
                                                                 item_id='business_inc
```

```
Recsys training: model = popularity


Warning: Ignoring columns X1, business_id, user_id;


    To use these columns in scoring predictions, use a model that allows the use of additional


Preparing data set.


    Data has 125853 observations with 18893 users and 4065 items.


    Data prepared in: 0.155502s


125853 observations to process; with 4065 unique items.
```

```
In [204]: rmse_train = popularity_model['training_rmse']
          rmse_test = gl.evaluation.rmse(test_set['stars'], popularity_model.predict(test_set))
          print('training set rmse: %f' % rmse_train)
          print('test set rmse: %f' % rmse_test)
```

```
training set rmse: 1.152739
test set rmse: 1.199568
```

1.4.2 When it comes to content based recommender, I think categories is more likely to be considered. Therefore, in this recommender, categories and business_id is considered.

```
In [87]: # Content-based recommender
         data = df[['business_id', 'categories']].drop_duplicates()
         print(data.shape)
         content_sf = gl.SFrame(data = data)
```

```
(4358, 2)
```

```
In [88]: content_sf.head(2)

Out[88]: Columns:
                 business_id        str
                 categories        str

         Rows: 2

         Data:
         +------------------------+------------------------------+
         |      business_id       |           categories         |
         +------------------------+------------------------------+
         | --9e1ONYQuAa-CB_Rrw7Tw | [Steakhouses, Restaurants,... |
         | -1vfRrlnNnNJ5boOVghMPA | [Restaurants, Korean, Sush... |
         +------------------------+------------------------------+
         [2 rows x 2 columns]
```

```
In [89]: content_sf.shape

Out[89]: (4358, 2)
```

```
In [90]: categories = content_sf['categories']
         content_sf.remove_column('categories')
         content_sf.head(2)

Out[90]: Columns:
                 business_id        str

         Rows: 2
```

```
Data:
+------------------------+
|      business_id       |
+------------------------+
| --9e1ONYQuAa-CB_Rrw7Tw |
| -1vfRrlnNnNJ5boOVghMPA |
+------------------------+
[2 rows x 1 columns]
```

In [91]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
vectorizer = TfidfVectorizer(stop_words='english', max_features=2000)
vectors = vectorizer.fit_transform(categories).toarray()
words = vectorizer.get_feature_names()
```

In [92]:
```python
vectors.shape
```

Out[92]:
```
(4358, 419)
```

In [93]:
```python
category_NLP = gl.SArray(vectors)
content_sf = content_sf.add_column(category_NLP, 'category_NLP')
```

In [94]:
```python
content_sf.head()
```

Out[94]:
```
Columns:
        business_id       str
        category_NLP       array

Rows: 10

Data:
+------------------------+-------------------------------+
|      business_id       |          category_NLP         |
+------------------------+-------------------------------+
| --9e1ONYQuAa-CB_Rrw7Tw | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -1vfRrlnNnNJ5boOVghMPA | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -3zffZUHoY8bQjGfPSoBKQ | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -8R_-EkGpUhBk55K9Dd4mg | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -9YyInW1wapzdNZrhQJ9dg | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -AD5PiuJHgdUcAK-Vxao2A | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -AGdGGCeTS-njB_8GkUmjQ | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -BS4aZAQm9u41YnB9MUASA | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -Bf8BQ3yMk8U2f45r2DRKw | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
| -BmqghX1sv7sgsxOIS2yAg | [0.0, 0.0, 0.0, 0.0, 0.0, ... |
+------------------------+-------------------------------+
[10 rows x 2 columns]
```

In [95]:
```python
# create recommender
category_content_based = gl.recommender.item_content_recommender.create(content_sf, "
```

```
WARNING: The ItemContentRecommender model is still in beta.
WARNING: This feature transformer is still in beta, and some interpretation rules may change i
('Applying transform:\n', Class             : AutoVectorizer

Model Fields
------------
Features          : ['category_NLP']
Excluded Features : ['business_id']

Column        Type   Interpretation  Transforms  Output Type
------------  -----  --------------  ----------  -----------
category_NLP  array  vector          None        array

)


Recsys training: model = item_content_recommender


Starting brute force nearest neighbors model training.


Starting blockwise querying.


max rows per data block: 7775


number of reference data blocks: 8


number of query data blocks: 1


+--------------+---------+-------------+--------------+


| Query points | # Pairs | % Complete. | Elapsed Time |


+--------------+---------+-------------+--------------+


| 4358         | 2375110 | 12.5057     | 695.061ms    |


| Done         | 1.9e+07 | 100         | 704.339ms    |


+--------------+---------+-------------+--------------+
```

```
Preparing data set.

    Data has 0 observations with 0 users and 4358 items.

    Data prepared in: 0.269214s

Loading user-provided nearest items.

Generating candidate set for working with new users.

Finished training in 0.017284s
```

1.4.3 When it comes to item-item similarity recommender, the business entity is considered as item and users are considered as user while stars are considered as ratings

In [381]:
```
# item-item similarity recommender
item_item_sf = gl.SFrame(df[['business_id', 'user_id', 'stars']])
item_item_sf = item_item_sf[['user_id', 'business_id', 'stars']]
item_item_sf.head(2)
```

Out[381]:
```
Columns:
        user_id        str
        business_id        str
        stars        int

Rows: 2

Data:
+------------------------+------------------------+-------+
|        user_id         |      business_id       | stars |
+------------------------+------------------------+-------+
| 0XVzm4kVIAaH4eQAxWbhvw | --9e1ONYQuAa-CB_Rrw7Tw |   1   |
| 2aeNFntqY2QDZLADNo8iQQ | --9e1ONYQuAa-CB_Rrw7Tw |   4   |
+------------------------+------------------------+-------+
[2 rows x 3 columns]
```

In [382]:
```
(train_set, test_set) = item_item_sf.random_split(0.8)
```

In [387]:
```
# grid search to find the best parameter set
thresholds = [10**(-i) for i in range(3,10)]
top_k = [2**i for i in range(4, 10)]
similarity_type = ['jaccard', 'cosine', 'pearson']
```

```
rmse_train = []
rmse_test = []
parameters = []
for threshold in thresholds:
    for k in top_k:
        for similarity in similarity_type:
            model = gl.item_similarity_recommender.create(train_set,
                                                          user_id = 'user_id',
                                                          item_id = 'business_id',
                                                          target = 'stars',
                                                          similarity_type = similari
                                                          threshold = threshold,
                                                          only_top_k = k,
                                                          verbose = False
                                                         )
            training_rmse = gl.evaluation.rmse(train_set['stars'], model.predict(tra
            testing_rmse = gl.evaluation.rmse(test_set['stars'], model.predict(test_
            rmse_train.append(training_rmse)
            rmse_test.append(testing_rmse)
            parameters.append((threshold, k, similarity))
            print('rmse_train:%f, rmse_test:%f, threshold:%f, k:%f, similarity:%s' %
```

```
Recsys training: model = item_similarity

rmse_train:4.063914, rmse_test:4.062520, threshold:0.001000, k:16.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.054978, rmse_test:4.059265, threshold:0.001000, k:16.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.255216, rmse_test:1.286935, threshold:0.001000, k:16.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063506, rmse_test:4.062215, threshold:0.001000, k:32.000000, similarity:jaccard

Recsys training: model = item_similarity
```

```
rmse_train:4.050940, rmse_test:4.057468, threshold:0.001000, k:32.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254633, rmse_test:1.286907, threshold:0.001000, k:32.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063015, rmse_test:4.061828, threshold:0.001000, k:64.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.045941, rmse_test:4.054981, threshold:0.001000, k:64.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254162, rmse_test:1.286869, threshold:0.001000, k:64.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.062461, rmse_test:4.061353, threshold:0.001000, k:128.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.040427, rmse_test:4.051880, threshold:0.001000, k:128.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253856, rmse_test:1.286832, threshold:0.001000, k:128.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061922, rmse_test:4.060841, threshold:0.001000, k:256.000000, similarity:jaccard
```

```
Recsys training: model = item_similarity

rmse_train:4.035150, rmse_test:4.048148, threshold:0.001000, k:256.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253690, rmse_test:1.286794, threshold:0.001000, k:256.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061496, rmse_test:4.060347, threshold:0.001000, k:512.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.031258, rmse_test:4.044397, threshold:0.001000, k:512.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253611, rmse_test:1.286770, threshold:0.001000, k:512.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063914, rmse_test:4.062519, threshold:0.000100, k:16.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.054977, rmse_test:4.059258, threshold:0.000100, k:16.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.255218, rmse_test:1.286935, threshold:0.000100, k:16.000000, similarity:pearson

Recsys training: model = item_similarity
```

```
rmse_train:4.063506, rmse_test:4.062214, threshold:0.000100, k:32.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.050944, rmse_test:4.057468, threshold:0.000100, k:32.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254630, rmse_test:1.286906, threshold:0.000100, k:32.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063015, rmse_test:4.061829, threshold:0.000100, k:64.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.045943, rmse_test:4.054986, threshold:0.000100, k:64.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254163, rmse_test:1.286869, threshold:0.000100, k:64.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.062468, rmse_test:4.061362, threshold:0.000100, k:128.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.040431, rmse_test:4.051888, threshold:0.000100, k:128.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253859, rmse_test:1.286833, threshold:0.000100, k:128.000000, similarity:pearson
```

```
Recsys training: model = item_similarity

rmse_train:4.061942, rmse_test:4.060854, threshold:0.000100, k:256.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.035147, rmse_test:4.048147, threshold:0.000100, k:256.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253678, rmse_test:1.286794, threshold:0.000100, k:256.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061510, rmse_test:4.060352, threshold:0.000100, k:512.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.031308, rmse_test:4.044436, threshold:0.000100, k:512.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253615, rmse_test:1.286769, threshold:0.000100, k:512.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063914, rmse_test:4.062519, threshold:0.000010, k:16.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.054980, rmse_test:4.059261, threshold:0.000010, k:16.000000, similarity:cosine

Recsys training: model = item_similarity
```

```
rmse_train:1.255217, rmse_test:1.286935, threshold:0.000010, k:16.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063506, rmse_test:4.062214, threshold:0.000010, k:32.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.050946, rmse_test:4.057468, threshold:0.000010, k:32.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254630, rmse_test:1.286906, threshold:0.000010, k:32.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063013, rmse_test:4.061822, threshold:0.000010, k:64.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.045944, rmse_test:4.054983, threshold:0.000010, k:64.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254168, rmse_test:1.286869, threshold:0.000010, k:64.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.062466, rmse_test:4.061363, threshold:0.000010, k:128.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.040414, rmse_test:4.051892, threshold:0.000010, k:128.000000, similarity:cosine
```

```
Recsys training: model = item_similarity

rmse_train:1.253856, rmse_test:1.286832, threshold:0.000010, k:128.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061936, rmse_test:4.060851, threshold:0.000010, k:256.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.035158, rmse_test:4.048141, threshold:0.000010, k:256.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253668, rmse_test:1.286793, threshold:0.000010, k:256.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061499, rmse_test:4.060344, threshold:0.000010, k:512.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.031281, rmse_test:4.044420, threshold:0.000010, k:512.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253613, rmse_test:1.286770, threshold:0.000010, k:512.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063914, rmse_test:4.062520, threshold:0.000001, k:16.000000, similarity:jaccard

Recsys training: model = item_similarity
```

```
rmse_train:4.054979, rmse_test:4.059258, threshold:0.000001, k:16.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.255210, rmse_test:1.286935, threshold:0.000001, k:16.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063505, rmse_test:4.062213, threshold:0.000001, k:32.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.050940, rmse_test:4.057466, threshold:0.000001, k:32.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254630, rmse_test:1.286906, threshold:0.000001, k:32.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063015, rmse_test:4.061829, threshold:0.000001, k:64.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.045942, rmse_test:4.054986, threshold:0.000001, k:64.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254161, rmse_test:1.286868, threshold:0.000001, k:64.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.062469, rmse_test:4.061364, threshold:0.000001, k:128.000000, similarity:jaccard
```

```
Recsys training: model = item_similarity

rmse_train:4.040434, rmse_test:4.051903, threshold:0.000001, k:128.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253858, rmse_test:1.286834, threshold:0.000001, k:128.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061930, rmse_test:4.060850, threshold:0.000001, k:256.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.035153, rmse_test:4.048161, threshold:0.000001, k:256.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253691, rmse_test:1.286794, threshold:0.000001, k:256.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061508, rmse_test:4.060360, threshold:0.000001, k:512.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.031272, rmse_test:4.044403, threshold:0.000001, k:512.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253624, rmse_test:1.286771, threshold:0.000001, k:512.000000, similarity:pearson

Recsys training: model = item_similarity
```

```
rmse_train:4.063915, rmse_test:4.062519, threshold:0.000000, k:16.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.054983, rmse_test:4.059268, threshold:0.000000, k:16.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.255217, rmse_test:1.286935, threshold:0.000000, k:16.000000, similarity:pearson
Recsys training: model = item_similarity


rmse_train:4.063506, rmse_test:4.062214, threshold:0.000000, k:32.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.050945, rmse_test:4.057468, threshold:0.000000, k:32.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254630, rmse_test:1.286907, threshold:0.000000, k:32.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063015, rmse_test:4.061828, threshold:0.000000, k:64.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.045932, rmse_test:4.054967, threshold:0.000000, k:64.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254170, rmse_test:1.286868, threshold:0.000000, k:64.000000, similarity:pearson
```

```
Recsys training: model = item_similarity

rmse_train:4.062475, rmse_test:4.061364, threshold:0.000000, k:128.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.040407, rmse_test:4.051883, threshold:0.000000, k:128.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253855, rmse_test:1.286833, threshold:0.000000, k:128.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061922, rmse_test:4.060843, threshold:0.000000, k:256.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.035167, rmse_test:4.048169, threshold:0.000000, k:256.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253692, rmse_test:1.286793, threshold:0.000000, k:256.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061500, rmse_test:4.060351, threshold:0.000000, k:512.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.031209, rmse_test:4.044377, threshold:0.000000, k:512.000000, similarity:cosine

Recsys training: model = item_similarity
```

```
rmse_train:1.253614, rmse_test:1.286772, threshold:0.000000, k:512.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063914, rmse_test:4.062519, threshold:0.000000, k:16.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.054977, rmse_test:4.059258, threshold:0.000000, k:16.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.255218, rmse_test:1.286935, threshold:0.000000, k:16.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063506, rmse_test:4.062214, threshold:0.000000, k:32.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.050941, rmse_test:4.057464, threshold:0.000000, k:32.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254631, rmse_test:1.286906, threshold:0.000000, k:32.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063017, rmse_test:4.061828, threshold:0.000000, k:64.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.045937, rmse_test:4.054989, threshold:0.000000, k:64.000000, similarity:cosine
```

```
Recsys training: model = item_similarity

rmse_train:1.254165, rmse_test:1.286868, threshold:0.000000, k:64.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.062470, rmse_test:4.061366, threshold:0.000000, k:128.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.040430, rmse_test:4.051893, threshold:0.000000, k:128.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253860, rmse_test:1.286833, threshold:0.000000, k:128.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061929, rmse_test:4.060844, threshold:0.000000, k:256.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.035159, rmse_test:4.048152, threshold:0.000000, k:256.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253689, rmse_test:1.286793, threshold:0.000000, k:256.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061493, rmse_test:4.060338, threshold:0.000000, k:512.000000, similarity:jaccard

Recsys training: model = item_similarity
```

```
rmse_train:4.031288, rmse_test:4.044453, threshold:0.000000, k:512.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253607, rmse_test:1.286770, threshold:0.000000, k:512.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063914, rmse_test:4.062519, threshold:0.000000, k:16.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.054976, rmse_test:4.059260, threshold:0.000000, k:16.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.255214, rmse_test:1.286935, threshold:0.000000, k:16.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063506, rmse_test:4.062215, threshold:0.000000, k:32.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.050946, rmse_test:4.057466, threshold:0.000000, k:32.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254630, rmse_test:1.286906, threshold:0.000000, k:32.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.063017, rmse_test:4.061829, threshold:0.000000, k:64.000000, similarity:jaccard
```

```
Recsys training: model = item_similarity

rmse_train:4.045941, rmse_test:4.054982, threshold:0.000000, k:64.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.254168, rmse_test:1.286869, threshold:0.000000, k:64.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.062468, rmse_test:4.061359, threshold:0.000000, k:128.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.040435, rmse_test:4.051895, threshold:0.000000, k:128.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253858, rmse_test:1.286833, threshold:0.000000, k:128.000000, similarity:pearson

Recsys training: model = item_similarity

rmse_train:4.061930, rmse_test:4.060851, threshold:0.000000, k:256.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.035160, rmse_test:4.048163, threshold:0.000000, k:256.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253682, rmse_test:1.286792, threshold:0.000000, k:256.000000, similarity:pearson

Recsys training: model = item_similarity
```

```
rmse_train:4.061506, rmse_test:4.060360, threshold:0.000000, k:512.000000, similarity:jaccard

Recsys training: model = item_similarity

rmse_train:4.031261, rmse_test:4.044423, threshold:0.000000, k:512.000000, similarity:cosine

Recsys training: model = item_similarity

rmse_train:1.253611, rmse_test:1.286769, threshold:0.000000, k:512.000000, similarity:pearson
```

```
In [388]: best_index = np.argmin(rmse_test)
          best_parameter = parameters[best_index]
          print(best_parameter)
```

```
(1e-09, 512, 'pearson')
```

```
In [390]: best_item_item_model = gl.item_similarity_recommender.create(train_set,
                                                                       user_id = 'user_id',
                                                                       item_id = 'business_id',
                                                                       target = 'stars',
                                                                       similarity_type = 'pearson
                                                                       threshold = 10**(-9),
                                                                       only_top_k = 512,
                                                                       verbose = True
                                                                      )
```

```
Recsys training: model = item_similarity

Preparing data set.

    Data has 278458 observations with 138725 users and 4311 items.

    Data prepared in: 0.381983s

Training model from provided data.

Gathering per-item and per-user statistics.
```

```
+--------------------------------+------------+
| Elapsed Time (Item Statistics) | % Complete |
+--------------------------------+------------+
| 778us                          | 9.25       |
| 22.003ms                       | 100        |
+--------------------------------+------------+
Setting up lookup tables.
Processing data in one pass using dense lookup tables.
+-------------------------------------+------------------+-----------------+
| Elapsed Time (Constructing Lookups) | Total % Complete | Items Processed |
+-------------------------------------+------------------+-----------------+
| 80.208ms                            | 0                | 0               |
| 262.962ms                           | 100              | 4311            |
+-------------------------------------+------------------+-----------------+
Finalizing lookup tables.
Generating candidate set for working with new users.
Finished training in 0.306563s
```

1.4.4 For the Hybrid recommender, I have considered content-based recommender and collaborative filtering recommender to be the best choice

In [86]:
```
# Let's say I am going to
sf_all = gl.SFrame(df[['business_id', 'user_id', 'categories', 'stars']])
sf_all.head(2)
```

Out[86]:
```
Columns:
        business_id        str
        user_id        str
        categories        str
        stars        int

Rows: 2

Data:
+------------------------+------------------------+
|      business_id       |        user_id         |
+------------------------+------------------------+
| --9e1ONYQuAa-CB_Rrw7Tw | 0XVzm4kVIAaH4eQAxWbhvw |
| --9e1ONYQuAa-CB_Rrw7Tw | 2aeNFntqY2QDZLADNo8iQQ |
+------------------------+------------------------+
+-------------------------------+-------+
|           categories          | stars |
+-------------------------------+-------+
| [Steakhouses, Restaurants,... |   1   |
| [Steakhouses, Restaurants,... |   4   |
+-------------------------------+-------+
[2 rows x 4 columns]
```

Train a content-based recommender

In [96]:
```
# set up a category map for all business entities
data = df[['business_id', 'categories']].drop_duplicates()
print(data.shape)
content_sf = gl.SFrame(data = data)
```

```
(4358, 2)
```

In [97]:
```
categories = content_sf['categories']
content_sf.remove_column('categories')
content_sf.head(2)
```

Out[97]:
```
Columns:
        business_id        str

Rows: 2
```

```
Data:
+------------------------+
|      business_id       |
+------------------------+
| --9e1ONYQuAa-CB_Rrw7Tw |
| -1vfRrlnNnNJ5boOVghMPA |
+------------------------+
[2 rows x 1 columns]
```

In [98]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
vectorizer = TfidfVectorizer(stop_words='english', max_features=2000)
vectors = vectorizer.fit_transform(categories).toarray()
words = vectorizer.get_feature_names()
```

In [99]:
```
category_NLP = gl.SArray(vectors)
content_sf = content_sf.add_column(category_NLP, 'category_NLP')
category_content_based = gl.recommender.item_content_recommender.create(content_sf, "b
```

```
WARNING: The ItemContentRecommender model is still in beta.
WARNING: This feature transformer is still in beta, and some interpretation rules may change in
('Applying transform:\n', Class             : AutoVectorizer

Model Fields
------------
Features          : ['category_NLP']
Excluded Features : ['business_id']

Column        Type   Interpretation  Transforms  Output Type
------------  -----  --------------  ----------  -----------
category_NLP  array  vector          None        array

)


Recsys training: model = item_content_recommender


Starting brute force nearest neighbors model training.


Starting blockwise querying.


max rows per data block: 7775


number of reference data blocks: 8
```

```
number of query data blocks: 1

+--------------+---------+------------+--------------+

| Query points | # Pairs | % Complete. | Elapsed Time |

+--------------+---------+------------+--------------+

| 4358         | 2375110 | 12.5057     | 617.069ms    |

| Done         | 1.9e+07 | 100         | 629.35ms     |

+--------------+---------+------------+--------------+

Preparing data set.

    Data has 0 observations with 0 users and 4358 items.

    Data prepared in: 0.317056s

Loading user-provided nearest items.

Generating candidate set for working with new users.

Finished training in 0.013906s
```

1.4.5 Cascade Hybrid Recommender System

This hybrid recommender system is to make use of both content based and collaborative filtering recommendation system to only evaluate the business entities that are very close to chosen category and predict the ratings based on only these entities.

```
In [104]: import random
          shape = sf_all.shape
          rand_index = random.randint(0, shape[0])
          chosen_row = sf_all[rand_index]
          chosen_row
```

```
Out[104]: {'business_id': 'iTCH7y2KLRMKMxdEkgUK1g',
           'categories': '[Restaurants, Food Trucks, Venezuelan, Food, Latin American]',
           'stars': 5,
           'user_id': 'e3c6w6sM8sJ_g0IlF6CSog'}
```

```
In [108]: # choose 300 closest business entities
          choose_k = 300
          chosen_k_entities = category_content_based.recommend_from_interactions([chosen_row['
          chosen_k_entities.head(2)
```

```
Out[108]: Columns:
                  business_id        str
                  score        float
                  rank        int

          Rows: 2

          Data:
          +------------------------+----------------+------+
          |      business_id       |     score      | rank |
          +------------------------+----------------+------+
          | 1uDUisf3ro5V3vC60cxfFw |      1.0       |  1   |
          | EnCIojgP5KTr1leaysFE3A | 0.763970792294 |  2   |
          +------------------------+----------------+------+
          [2 rows x 3 columns]
```

```
In [116]: df_chosen_business_id = chosen_k_entities[['business_id']].to_dataframe()
          df_filtered = pd.merge(df, df_chosen_business_id, left_on = 'business_id', right_on =
          df_filtered.head(2)
```

```
Out[116]:              business_id                  name  \
          0  -mN7z9oY01Mh_-dwTyzpqg  Einstein Bros Bagels
          1  -mN7z9oY01Mh_-dwTyzpqg  Einstein Bros Bagels

                                   categories  avg_stars  cool        date  funny  \
          0  [Breakfast & Brunch, Restaurants]        3.5     1  2016-12-10      0
          1  [Breakfast & Brunch, Restaurants]        3.5     1  2015-06-27      0

                          review_id  stars  \
          0  bwTUmpkPpZKbtVuHl-vE9Q      4
          1  5occn6g-CNryBX1Eojgp7A      5

                                                          text    type  useful  \
          0  I think that this Einstein's is one of the bet...  review       0
          1  I have been to Einstein locations all over Veg...  review       0

                            user_id  count
          0  E63DvbIQptvI05F63F1Bdg     36
          1  lqKZ0uHeqoY3WdwBQj7gZw     36
```

```
In [117]: cf_sf = gl.SFrame(df_filtered[['business_id', 'user_id', 'stars']])
          cf_sf.head(2)

Out[117]: Columns:
                  business_id         str
                  user_id         str
                  stars         int

          Rows: 2

          Data:
          +------------------------+------------------------+-------+
          |      business_id       |        user_id         | stars |
          +------------------------+------------------------+-------+
          | -mN7z9oY01Mh_-dwTyzpqg | E63DvbIQptvI05F63F1Bdg |   4   |
          | -mN7z9oY01Mh_-dwTyzpqg | lqKZOuHeqoY3WdwBQj7gZw |   5   |
          +------------------------+------------------------+-------+
          [2 rows x 3 columns]

In [118]: cf_sf.shape

Out[118]: (22040, 3)

In [119]: (train_set, test_set) = cf_sf.random_split(0.7)

In [124]: CF_model = gl.item_similarity_recommender.create(train_set,
                                                            user_id = 'user_id',
                                                            item_id = 'business_id',
                                                            target = 'stars',
                                                            similarity_type = 'pearson
                                                            threshold = 10**(-9),
                                                            only_top_k = 512,
                                                            verbose = True
                                                           )

Recsys training: model = item_similarity


Preparing data set.


    Data has 15351 observations with 13326 users and 296 items.


    Data prepared in: 0.040412s


Training model from provided data.
```

Gathering per-item and per-user statistics.

Elapsed Time (Item Statistics)	% Complete
759us	7.5
7.023ms	100

Setting up lookup tables.

Processing data in one pass using dense lookup tables.

Elapsed Time (Constructing Lookups)	Total % Complete	Items Processed
12.483ms	0	0
27.819ms	100	296

Finalizing lookup tables.

Generating candidate set for working with new users.

```
Finished training in 0.037583s
```

```
In [125]: score = gl.evaluation.rmse(test_set['stars'], CF_model.predict(test_set))
          score
```

```
Out[125]: 1.2495618522895826
```

```
In [127]: recommend_result = CF_model.recommend(new_observation_data=test_set)
```

```
recommendations finished on 1000/13326 queries. users per second: 88896.8

recommendations finished on 2000/13326 queries. users per second: 89710.2

recommendations finished on 3000/13326 queries. users per second: 90435

recommendations finished on 4000/13326 queries. users per second: 92504.8

recommendations finished on 5000/13326 queries. users per second: 94158.4

recommendations finished on 6000/13326 queries. users per second: 92296.3

recommendations finished on 7000/13326 queries. users per second: 90377.4

recommendations finished on 8000/13326 queries. users per second: 88537.7

recommendations finished on 9000/13326 queries. users per second: 88247.4

recommendations finished on 10000/13326 queries. users per second: 87849.5

recommendations finished on 11000/13326 queries. users per second: 88340.6

recommendations finished on 12000/13326 queries. users per second: 88140

recommendations finished on 13000/13326 queries. users per second: 87423.8
```

```
In [128]: recommend_result
```

```
Out[128]: Columns:
                user_id         str
                business_id         str
                score       float
                rank        int

        Rows: 133260

        Data:
        +------------------------+------------------------+-------+------+
        |        user_id         |      business_id       | score | rank |
        +------------------------+------------------------+-------+------+
        | E63DvbIQptvI05F63F1Bdg | EUskCVPgHoIGl3Ao0VUNLA |  5.0  |  1   |
        | E63DvbIQptvI05F63F1Bdg | 8hZjiPzJIojA1k7_W4hELA |  5.0  |  2   |
        | E63DvbIQptvI05F63F1Bdg | 3nWXxSCR6q7SVxHvsUUeWg |  5.0  |  3   |
        | E63DvbIQptvI05F63F1Bdg | 3PshdJtSwd_poaPL7fIOHg |  5.0  |  4   |
        | E63DvbIQptvI05F63F1Bdg | 39BNnMLzx0UUwkWKxGeQKA |  5.0  |  5   |
        | E63DvbIQptvI05F63F1Bdg | 35tWXOOJpWB2feUAEJyJyg |  5.0  |  6   |
        | E63DvbIQptvI05F63F1Bdg | 2sjfncn3GNHpKZDAcsm1Uw |  5.0  |  7   |
        | E63DvbIQptvI05F63F1Bdg | 0i9S0BejjRv0ZDwdO9XymA |  5.0  |  8   |
        | E63DvbIQptvI05F63F1Bdg | 0N53m33GANYeHHl-s22d4Q |  5.0  |  9   |
        | E63DvbIQptvI05F63F1Bdg | -xbQQR_ydEJGqYzHSF4DnQ |  5.0  |  10  |
        +------------------------+------------------------+-------+------+
        [133260 rows x 4 columns]
        Note: Only the head of the SFrame is printed.
        You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.
```

1.4.6 Switching.

When a (relatively) new user comes, very few ratings has been made, then content-based recommender is used. This might on some level solve the 'cold start' problem

```
In [165]: category_content_based = gl.recommender.item_content_recommender.create(content_sf, 
```

```
WARNING: The ItemContentRecommender model is still in beta.
WARNING: This feature transformer is still in beta, and some interpretation rules may change i
('Applying transform:\n', Class             : AutoVectorizer

Model Fields
------------
Features          : ['category_NLP']
Excluded Features : ['business_id']

Column        Type   Interpretation  Transforms  Output Type
------------  -----  --------------  ----------  -----------
category_NLP  array  vector          None        array

)
```

```
Recsys training: model = item_content_recommender

Starting brute force nearest neighbors model training.

Starting blockwise querying.

max rows per data block: 7775

number of reference data blocks: 8

number of query data blocks: 1

+--------------+---------+------------+-------------+

| Query points | # Pairs | % Complete. | Elapsed Time |

+--------------+---------+------------+-------------+

| 4358         | 2370752 | 12.4828    | 747.275ms   |

| Done         | 1.9e+07 | 100        | 759.194ms   |

+--------------+---------+------------+-------------+

Preparing data set.

    Data has 0 observations with 0 users and 4358 items.

    Data prepared in: 0.311917s

Loading user-provided nearest items.

Generating candidate set for working with new users.
```

```
Finished training in 0.018105s
```

```
In [169]: shape = df.shape
          random_row = random.randint(0, shape[0])
          chosen_user = df.iloc[random_row,].user_id
          df_chosen_user = df[df['user_id'] == chosen_user]
          print('chosen user is: %s' % chosen_user)
          df_chosen_user
```

```
chosen user is: sa4b69yUwJ6QeHV20XPnNg
```

```
Out[169]:                   business_id                           name  \
          309713  rbDqCV2g23K3ZrTxmgoNBg  Biscayne Steak, Sea and Wine

                                                       categories  avg_stars  cool  \
          309713  [American (New), Steakhouses, Restaurants, Sea...        4.0     0

                        date  funny              review_id  stars  \
          309713  2016-11-05      1  c5WPPFXpvB2aL1dUTZ9rcw      4

                                                        text    type  useful  \
          309713  I had the bone in filet and my partner had the...  review       1

                                 user_id  count
          309713  sa4b69yUwJ6QeHV20XPnNg     49
```

```
In [170]: high_rate_business_for_user = list(df_chosen_user[df_chosen_user['stars'] >= 3]['bus
          high_rate_business_for_user
```

```
Out[170]: ['rbDqCV2g23K3ZrTxmgoNBg']
```

```
In [173]: if (len(high_rate_business_for_user) > 0):
              recommend_result = None
              threshold = 2
              if df_chosen_user.shape[0] >= 10:
                  # use item-similarity recommender
                  CF_model = gl.item_similarity_recommender.create(sf_all,
                                                                   user_id = 'user_id',
                                                                   item_id = 'business_id',
                                                                   target = 'stars',
                                                                   similarity_type = 'pearson'
                                                                   threshold = 10**(-9),
                                                                   only_top_k = 512,
                                                                   verbose = True
                                                                  )
                  recommend_result = CF_model.recomend(k = 5)
              else:
```

```
                # use content-based recommender
                recommend_result = category_content_based.recommend_from_interactions(high_ra
            recommend_df = recommend_result.to_dataframe()
            all_information = pd.merge(df[['business_id', 'avg_stars', 'name', 'categories']]
                                       how='inner', left_on = 'business_id', right_on= 'busin
            print(all_information.head())
        else:
            print('No quality rating for this user')
```

```
             business_id  avg_stars                        name  \
0  3Gt3xskppi9jZuTrwrhLNg        4.0      Stack Restaurant & Bar
1  7UFDAX4wLi6ux5otguYldA        4.0         Andiron Steak & Sea
2  bjSC_jbrypke0l-bXXBmwQ        4.5  Vic & Anthony's Steakhouse
3  jCR-xC4NqoEajjmstqX8sA        4.0                 Twin Creeks
4  tQifTiY-vutj8orxcMJKfQ        4.0         Triple George Grill

                                          categories     score  rank
0  [Seafood, Restaurants, American (New), Steakho...  1.000000     4
1  [Steakhouses, Seafood, American (New), Restaur...  1.000000     3
2  [American (Traditional), Restaurants, Steakhou...  0.903751     5
3  [Restaurants, Seafood, Steakhouses, American (...  1.000000     2
4  [Restaurants, Steakhouses, Seafood, American (...  1.000000     1
```