

RecSys BrickMover's Source Code

Liang Pang, Yuyu Zhang, Xudong Liu,
Lei Shi, Qiang Li, Hanxiao Sun, Lu Liu

Copyright [2013] [BrickMover Team]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Source Code

1. Yuyu

- *benchmark.py*

Our naive benchmark depending on several types of averages.

```
1  # -*- coding: utf-8 -*-

from time import time
from collections import defaultdict
import pylab as pl
6  import json

def outputFinalPredict(pre, outfile, testdata):
    schema = []
    for line in file(testdata):
11         js = json.loads(line)
            rid = js['review_id']
            schema.append([rid])
    f = file(outfile, 'w')
    f.write('review_id,stars\n')
16    for i, parts in enumerate(schema):
        parts.append(str(pre[i]))
        f.write(','.join(parts) + '\n')
    f.close()

21 def readTrainingDict(filename):
    ku = dict()
    ki = dict()
    for line in file(filename):
        js = json.loads(line)
26         uid = js['user_id']
            iid = js['business_id']
            ku[uid] = 1
            ki[iid] = 1
    return ku, ki

31 def getSparseMat():
    user = defaultdict(lambda:defaultdict(lambda:0))
    item = defaultdict(lambda:defaultdict(lambda:0))
    ratings = []
36    for line in file(data_path + 'yelp_training_set_review.json'):
        js = json.loads(line)
        uid = js['user_id']
        iid = js['business_id']
        rating = js['stars']
41    ratings.append(rating)
```

```

        user[uid][iid] = rating
        item[iid][uid] = rating
    return user, item, ratings

46 def getSampleAvg(ratings):
    user_avg = dict()
    item_avg = dict()
    for u, items in user.items():
        user_avg[u] = pl.average(items.values())
51 for i, users in item.items():
        item_avg[i] = pl.average(users.values())
    global_avg = pl.average(ratings)
    return user_avg, item_avg, global_avg

56 def getPopAvg():
    user_avg_given = dict()
    item_avg_given = dict()
    item_avg_predict = dict()
    user_review_cnt_given = dict()
61 item_review_cnt_given = dict()
    for line in file (data_path + 'yelp-training-set-user.json'):
        js = json.loads(line)
        uid = js['user_id']
        avg = js['average_stars']
66 review_cnt = js['review_count']
        user_avg_given[uid] = avg
        user_review_cnt_given[uid] = review_cnt
    for line in file (data_path + 'yelp-training-set-business.json'):
        js = json.loads(line)
71 iid = js['business_id']
        avg = js['stars']
        review_cnt = js['review_count']
        item_avg_given[iid] = avg
        item_review_cnt_given[iid] = review_cnt
76 return user_avg_given, item_avg_given, user_review_cnt_given, item_review_cnt_given

def getInferAvg():
    user_avg_infer = dict()
    for uid in user_avg_given:
81 if uid in user_avg:
        cnt_given = user_review_cnt_given[uid]
        cnt_train = len(user[uid])
        avg_given = user_avg_given[uid]
        avg_train = user_avg[uid]
86 if cnt_given > cnt_train:
            #if cnt_given - cnt_train == 1:
                infer = (avg_given * cnt_given - avg_train * cnt_train)*1.0 / (cnt_given - cnt_train)
                if infer >= 1.0 and infer <= 5.0:
                    user_avg_infer[uid] = infer
91
        item_avg_infer = dict()
        for iid in item_avg_given:
            if iid in item_avg:
                cnt_given = item_review_cnt_given[iid]
                cnt_train = len(item[iid])
96 avg_given = item_avg_given[iid]
                avg_train = item_avg[iid]
                if cnt_given > cnt_train:
                    infer = (avg_given * cnt_given - avg_train * cnt_train)*1.0 / (cnt_given - cnt_train)
101 if infer >= 1.0 and infer <= 5.0:
                        item_avg_infer[iid] = infer

    return user_avg_infer, item_avg_infer

106 def getPrediction(filename):
    pre = []
    for line in file (testing):
        js = json.loads(line)

```

```

111     uid = js['user_id']
        iid = js['business_id']
        if uid in user_avg_given and iid in item_avg_given:
            pre.append(0.4*user_avg_given[uid] + 0.6*item_avg_given[iid])
        elif uid in user_avg_given and iid not in item_avg_given:
            if uid in user_avg_infer:
116                pre.append(user_avg_infer[uid])
            else:
                pre.append(0.4*user_avg_given[uid] + 0.6*global_avg)
        elif uid not in user_avg_given and iid in item_avg_given:
            pre.append(item_avg_given[iid])
121        elif uid not in user_avg_given and iid not in item_avg_given:
            if uid in user_avg:
                pre.append(0.4*user_avg[uid] + 0.6*global_avg)
            else:
                pre.append(global_avg)
126    return pre

begin = time()

131 data_path = './Data/'
    testing = data_path + 'final_test_set_review.json'
    outfile = 'Benchmark_InferAvg.csv'

    user, item, ratings = getSparseMat()
136    user_avg, item_avg, global_avg = getSampleAvg(ratings)
    user_avg_given, item_avg_given, user_review_cnt_given, item_review_cnt_given = getPopAvg()
    user_avg_infer, item_avg_infer = getInferAvg()

    pre = getPrediction(outfile)
141    outputFinalPredict(pre, outfile, testing)

```

```

print '\nTotal_Execution_Time:_%3fs' % (time() - begin)

```

- *model_easy_ml.py*

An easy linear machine learning model.

```

1  # -*- coding: utf-8 -*-

    from time import time
    from sklearn import svm
    from sklearn.preprocessing import StandardScaler
6    from sklearn.linear_model import LinearRegression, Ridge, RidgeCV, SGDRegressor
    from sklearn.datasets import load_svmlight_file, load_svmlight_files
    from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, ExtraTreesRegressor
    from sklearn.metrics import mean_squared_error
    from collections import defaultdict
11    from sklearn.tree import DecisionTreeRegressor
    import pylab as pl
    import numpy as np
    from math import sqrt
    import json

16    def readItemJson():
        js_dic = dict()
        for line in file (data_path + 'yelp_training_set_business.json'):
            js = json.loads(line)
21            iid = js['business_id']
            js_dic[iid] = js
        for line in file (data_path + 'yelp_test_set_business.json'):
            js = json.loads(line)
            iid = js['business_id']
26            js_dic[iid] = js
        return js_dic

    def readUserJson():

```

```

js_dic = dict()
31 for line in file (data_path + 'yelp_training_set_user.json'):
    js = json.loads(line)
    uid = js['user_id']
    js_dic[uid] = js
for line in file (data_path + 'yelp_test_set_user.json'):
36     js = json.loads(line)
    uid = js['user_id']
    js_dic[uid] = js
return js_dic

41 def readDict(filename):
    dic = dict()
    for line in file (filename):
        rec = line.strip().split('\t')
        dic[rec[0]] = int(rec[1])
46 return dic

def getBlankDict(fromdic):
    dic = dict()
    for k in fromdic.keys():
51         dic[k] = 0
    return dic

def getData(filename):
    X = []
56 y = []
    for line in file (filename):
        js = json.loads(line)
        uid = js['user_id']
        iid = js['business_id']
61 rating = js['stars']

        item_js = item_jsons[iid]
        ct = item_js['city']
        review_cnt = item_js['review_count']
66 longitude = item_js['longitude']
        latitude = item_js['latitude']
        cats = item_js['categories']
        addr = item_js['full_address']
        opened = item_js['open']

71 open_flag = 0
        if opened:
            open_flag = 1

76 city_feature = getBlankDict(city)
        city_feature[ct] = 1

        category_feature = getBlankDict(category)
        for c in cats:
81             category_feature[c] = 1

        code_feature = getBlankDict(code)
        cd = addr.split('.')[1]
        code_feature[cd] = 1

86 sample = [review_cnt, open_flag] \
            + category_feature.values() \
            + city_feature.values() \
            + code_feature.values()

91 X.append(sample)
    y.append(rating)
return X, y

def outputFinalPredict(pre, outfile, testdata):
96     schema = []
    for line in file (testdata):

```

```

        js = json.loads(line)
        uid = js['user_id']
        iid = js['business_id']
101     schema.append([uid, iid])
        f = file(outfile, 'w')
        f.write('user_id,business_id,stars\n')
        for i, parts in enumerate(schema):
            parts.append(str(pre[i]))
106         f.write(''.join(parts) + '\n')
        f.close()

begin = time()

111 data_path = './Data/'
    testing = data_path + 'yelp_test_set_review.json'

X_train, y_train, X_test, y_test = load_svmlight_files(('train', 'test'))
116 print '\nData loading complete in %.3fs\n' % (time() - begin)

#regr = LinearRegression()
#regr = Ridge()
regr = SGDRegressor()

121 regr.fit(X_train, y_train)
    pre = regr.predict(X_test)

    outfile = 'easyML.csv'
126 outputFinalPredict(pre, outfile, testing)

print '\nTotal Execution Time: %.3fs' % (time() - begin)

```

- *generate_cross_binary_features.py*

A general framework to generate cross binary features.

```

1  # -*- coding: utf-8 -*-

from time import time
from collections import defaultdict
import re
6  import json
import pylab as pl
import numpy as np

11 def getIdMap():
    iid = dict()
    for line in file('itemmap.final'):
        item, mapped = line.strip().split('\t')
        iid[item] = int(mapped)
16    uid = dict()
    for line in file('usermap.final'):
        user, mapped = line.strip().split('\t')
        uid[user] = int(mapped)
    return iid, uid

21

begin = time()

data_path = './Data/'
26 outfile = 'Yuyu_CategoryPopAvg_50bins.txt'

iid_map, uid_map = getIdMap()

biz_js = []
31 for line in file(data_path + 'yelp_training_set_business.json'):
    biz_js.append(json.loads(line))

```

```

category_reviewcnt = defaultdict(lambda:0)
category_reviewcnt_x_stars = defaultdict(lambda:0)
36 category_popavg = dict()
cnt = 0
for js in biz_js :
    iid = js['business_id']
    cats = js['categories']
41
    reviewcnt = js['review_count']
    stars = js['stars']
    for c in cats:
        category_reviewcnt[c] += reviewcnt
46        category_reviewcnt_x_stars[c] += stars*reviewcnt
        break
for c in category_reviewcnt:
    category_popavg[c] = category_reviewcnt_x_stars[c]*1.0/category_reviewcnt[c]

51
for line in file (data_path + ' final_test_set_business .json'):
    biz_js.append(json.loads(line))

ITEM_FEATURE__CatogoryPopAvg = dict()
56 for js in biz_js :
    mapped_iid = iid_map[js['business_id']]
    cats = js['categories']
    if len(cats) > 0 and cats[0] in category_popavg:
        ITEM_FEATURE__CatogoryPopAvg[mapped_iid] = category_popavg[cats[0]]
61 else :
    ITEM_FEATURE__CatogoryPopAvg[mapped_iid] = np.mean(category_popavg.values())

def get_bins_1_idx (value):
66     bincnt = 50
    minv = 1
    maxv = 5
    binwidth = (maxv-minv)*1.0/bincnt
    idx = int((value - minv)/binwidth)
71     return idx

f = file ( outfile , 'w')
for mapped_iid in sorted(iid_map.itervalues()):
76     if mapped_iid in ITEM_FEATURE__CatogoryPopAvg:
        f.write('%d\t%d:1\n' % (mapped_iid, get_bins_1_idx(ITEM_FEATURE__CatogoryPopAvg[mapped_iid])))
f.close()

81 print '\nTotal_Execution_Time:_.3fs' % (time() - begin)

```

- *generate_feature_ReviewCountContainFinalTest.py*

Generate the feature of review count, taking testing set review pairs into consideration.

```
# -*- coding: utf-8 -*-
```

```

from time import time
4 import json
import pylab as pl
import numpy as np
from collections import defaultdict
import os
9
def getIdMap():
    iid = dict()
    for line in file ('itemmap.final'):
        item, mapped = line.strip().split('\t')
14        iid[item] = int(mapped)
    uid = dict()

```

```

    for line in file('usermap.final'):
        user, mapped = line.strip().split('\t')
        uid[user] = int(mapped)
19     return iid, uid

def readJsons():
    biz_js = []
    for line in file(data_path + 'yelp_training_set_business.json'):
24         biz_js.append(json.loads(line))
    for line in file(data_path + 'final_test_set_business.json'):
        biz_js.append(json.loads(line))

    user_js = []
29     for line in file(data_path + 'yelp_training_set_user.json'):
        user_js.append(json.loads(line))
    for line in file(data_path + 'final_test_set_user.json'):
        user_js.append(json.loads(line))

34     return biz_js, user_js

def readTest():
    t_item = defaultdict(lambda:defaultdict(lambda:0))
    t_user = defaultdict(lambda:defaultdict(lambda:0))
39     for line in file(testing_set):
        js = json.loads(line)
        uid = js['user_id']
        iid = js['business_id']
        t_item[iid][uid] = 0
44         t_user[uid][iid] = 0
    return t_user, t_item

def get_bins_1_idx(value):
    bincnt = 20
49     t = np.log(rvcnt.values())
    maxv = np.max(t)
    minv = np.min(t)
    binwidth = (maxv-minv)*1.0/bincnt
    idx = int((np.log(value) - minv)/binwidth)
54     return idx

data_path = '../Data/'
training_set = data_path + 'yelp_training_set_review.json'
59 testing_set = data_path + 'final_test_set_review.json'
outfile = 'Yuyu_UserReviewCountContainFinalTest_21bins.txt'

iid_map, uid_map = getIdMap()

64 t_user, t_item = readTest()

biz_js, user_js = readJsons()

rvcnt = dict()
69 for js in user_js:
    uid = js['user_id']
    reviewcnt = js['review_count']
    rvcnt[iid_map[uid]] = reviewcnt + len(t_user[uid])

74 f = file(outfile, 'w')
for mapped_uid in sorted(uid_map.itervalues()):
    if mapped_uid in rvcnt:
        f.write('%d\t%d:1\n' % (mapped_uid, get_bins_1_idx(rvcnt[mapped_uid])))
f.close()

```

- *generate_features_Gender.py*

Generate gender features

-*- coding: utf-8 -*-

```

2
from time import time
from collections import defaultdict
import re
import json
7 import numpy as np

def getIdMap():
    iid = dict()
12     for line in file('itemmap.final'):
        item, mapped = line.strip().split('\t')
        iid[item] = int(mapped)
    uid = dict()
    for line in file('usermap.final'):
17         user, mapped = line.strip().split('\t')
        uid[user] = int(mapped)
    return iid, uid

def getNameList(filename):
22     names = dict()
    for line in file(filename):
        names[line.split('_')[0].lower()] = 1
    return names

27
begin = time()

data_path = '../Data/'
outfile = 'Yuyu_UserGender.txt'

32
iid_map, uid_map = getIdMap()
male_name = getNameList('dist.male.first.txt')
female_name = getNameList('dist.female.first.txt')

37
user_js = []
for line in file(data_path + 'yelp_training_set_user.json'):
    user_js.append(json.loads(line))
for line in file(data_path + 'final_test_set_user.json'):
42     user_js.append(json.loads(line))

user_gender = defaultdict(lambda:0)
for js in user_js:
47     mapped_uid = uid_map[js['user_id']]
    name = js['name'].lower()
    if name in male_name:
        user_gender[mapped_uid] = 1
    elif name in female_name:
52         user_gender[mapped_uid] = 2

f = file(outfile, 'w')
for mapped_uid in sorted(uid_map.itervalues()):
    f.write('%d\t%d:1\n' % (mapped_uid, user_gender[mapped_uid]))
57 f.close()

print '\nTotal_Execution_Time: %.3fs' % (time() - begin)

```

2. Liang

- *CategoryKnn.py*

Business has the same category.

```

import sys
import json

```



```

Cat_Bid_Map = {}
5 Bid_Cat_Map = {}

idx_c = 0

for line in open('../Data/yelp_training_set_business.s.json'):
10 js = json.loads(line)
    bid = js['business_id']
    cset = tuple(js['categories'])
    Bid_Cat_Map[bid]=cset
    if cset not in Cat_Bid_Map:
15 Cat_Bid_Map[cset]=[bid]
    else:
        Cat_Bid_Map[cset].append(bid)

20 for line in open('../Data/yelp_test_set_business.s.json'):
    js = json.loads(line)
    bid = js['business_id']
    cset = tuple(js['categories'])
    Bid_Cat_Map[bid]=cset
25 if cset not in Cat_Bid_Map:
    Cat_Bid_Map[cset] = [bid]
    else:
        Cat_Bid_Map[cset].append(bid)

30 print len(Cat_Bid_Map)

outf = open('../GlobalFeature/i/KateKnn.txt','w')
outf.write('%d\n'%(14334))
for item in range(14334):
35 outf.write('%d\t'%(item))
    for k in Cat_Bid_Map[Bid_Cat_Map[item]]:
        outf.write('%d:%f\t'%(k,0.5/len(Cat_Bid_Map[Bid_Cat_Map[item]])))
    outf.write('\n')
outf.close()

```

- *ItemCategoryAll.py*

Business's category binary feature.

```

1 import sys
import json

Cat_Bid_Map = {}
Bid_Cat_Map = {}

6 idx_c = 0

for line in open('../Data/yelp_training_set_business.s.json'):
    js = json.loads(line)
11 bid = js['business_id']
    cset = tuple(js['categories'])

    if cset not in Cat_Bid_Map:
        Cat_Bid_Map[cset]=idx_c
16 idx_c+=1

    Bid_Cat_Map[bid]=Cat_Bid_Map[cset]

21 for line in open('../Data/yelp_test_set_business.s.json'):
    js = json.loads(line)
    bid = js['business_id']
    cset = tuple(js['categories'])
    Bid_Cat_Map[bid]=cset
26 if cset not in Cat_Bid_Map:
    Cat_Bid_Map[cset]=idx_c

```

```
idx_c+=1
```

```
31 Bid_Cat_Map[bid]=Cat_Bid_Map[cset]
```

```
print len(Cat_Bid_Map)
```

```
outf = open('../GlobalFeature/i/ItemCategories.txt','w')
```

```
36 outf.write('%d\n'%(len(Cat_Bid_Map)))
```

```
for item in range(14334):
```

```
    if item in Bid_Cat_Map:
```

```
        outf.write('%d\t%d:1\n'%(item,Bid_Cat_Map[item]))
```

```
outf.close()
```

- *ItemLocationKnn.py*

Business nearby k Business (the distance caculate by Longitude and Latitude).

```
import numpy
```

```
import json
```

```
import math
```

```
from sklearn.neighbors import kneighbors_graph
```

```
5
```

```
k = 5
```

```
Item_Num = 12742
```

```
10 item_f={}
```

```
item_latitude=[0]*Item_Num
```

```
item_longitude=[0]*Item_Num
```

```
item_location=[0]*Item_Num
```

```
15
```

```
for line in open('../Data/yelp_training_set_business_s.json'):
```

```
    js = json.loads(line)
```

```
    bid = js['business_id']
```

```
    item_latitude[bid]=[js['latitude']]
```

```
20
```

```
    item_longitude[bid]=[js['longitude']]
```

```
    item_location[bid]=[js['latitude'], js['longitude']]
```

```
for line in open('../Data/yelp_test_set_business_s.json'):
```

```
    js = json.loads(line)
```

```
25
```

```
    bid = js['business_id']
```

```
    item_latitude[bid]=[js['latitude']]
```

```
    item_longitude[bid]=[js['longitude']]
```

```
    item_location[bid]=[js['latitude'], js['longitude']]
```

```
30 print 'load_lover!'
```

```
latitude_knn = kneighbors_graph(item_latitude,k).tolil().rows
```

```
print '1'
```

```
longitude_knn = kneighbors_graph(item_longitude,k).tolil().rows
```

```
35
```

```
print '2'
```

```
location_knn = kneighbors_graph(item_location,k).tolil().rows
```

```
print '3'
```

```
fout0 = open('../GlobalFeature/i/ItemKnnLatitude.txt','w')
```

```
40
```

```
fout1 = open('../GlobalFeature/i/ItemKnnLongitude.txt','w')
```

```
fout2 = open('../GlobalFeature/i/ItemKnnLocation.txt','w')
```

```
fout0.write('%d\n'%Item_Num)
```

```
fout1.write('%d\n'%Item_Num)
```

```
45
```

```
fout2.write('%d\n'%Item_Num)
```

```
for line in open('../Data/yelp_test_set_business_s.json'):
```

```
    js = json.loads(line)
```

```
    bid = js['business_id']
```

```
50
```

```
    fout0.write('%d\t'%(bid+1))
```

```
    fout1.write('%d\t'%(bid+1))
```

```
    fout2.write('%d\t'%(bid+1))
```

```

    if item_latitude[bid]==0:
        fout0.write('%d:1\n'%(Item_Num))
55      fout1.write('%d:1\n'%(Item_Num))
        fout2.write('%d:1\n'%(Item_Num))
        continue
    for item in latitude_knn[bid]:
        fout0.write('%d:1\t'%(item))
60      for item in longitude_knn[bid]:
        fout1.write('%d:1\t'%(item))
        for item in location_knn[bid]:
            fout2.write('%d:1\t'%(item))
        fout0.write('\n')
65      fout1.write('\n')
        fout2.write('\n')
    fout0.close()
    fout1.close()
    fout2.close()

```

- *SimpleItemFeature.py*

Generate the business feature from [yelp_training_set_business.json] and [yelp_test_set_business.json] directly, such as city, state, review count and category information.

```

1  import numpy
   import json
   import math

   item_f={}

6
   item_city={}
   idx_city=0
   item_review_count={}
   item_state={}
11  idx_state=0
   item_categories={}
   idx_cat=0

   maxv=0
16  minv=100

   for line in open('../Data/yelp_training_set_business_s.json'):
       js = json.loads(line)
       bid = js['business_id']
21      if js['city'] not in item_city:
           item_city[js['city']]=idx_city
           idx_city+=1
       log_review=math.log(js['review_count'],2)
       if log_review<minv:
26           minv=log_review
       if log_review>maxv:
           maxv=log_review
       if js['state'] not in item_state:
           item_state[js['state']]=idx_state
31      idx_state+=1
       for c in js['categories']:
           if c not in item_categories:
               item_categories[c]=idx_cat
               idx_cat+=1
36      item_f[bid]=[item_city[js['city']], log_review, item_state[js['state']], [item_categories[c] for c in js['categories']], js['
           open']]

   print 'city:%d'%idx_city
   print 'state:%d'%idx_state
   print 'categories:%d'%idx_cat
41
   for line in open('../Data/yelp_test_set_business_s.json'):
       js = json.loads(line)
       bid = js['business_id']
       if js['city'] not in item_city:

```

```

46     item_city[js['city']] = idx_city
        idx_city += 1
    log_review = math.log(js['review_count'], 2)
    if log_review < minv:
        minv = log_review
51     if log_review > maxv:
        maxv = log_review
    if js['state'] not in item_state:
        item_state[js['state']] = idx_state
        idx_state += 1
56     for c in js['categories']:
        if c not in item_categories:
            item_categories[c] = idx_cat
            idx_cat += 1
    item_f[bid] = [item_city[js['city']], log_review, item_state[js['state']], [item_categories[c] for c in js['categories']], js['
        open']]

61     print 'city:%d'%idx_city
    print 'state:%d'%idx_state
    print 'categories:%d'%idx_cat

66     fout0 = open('../GlobalFeature/i/ItemCity.txt', 'w')
    fout1 = open('../GlobalFeature/i/ItemReview.txt', 'w')
    fout2 = open('../GlobalFeature/i/ItemState.txt', 'w')
    fout3 = open('../GlobalFeature/i/ItemCategories.txt', 'w')
    fout4 = open('../GlobalFeature/i/ItemOpen.txt', 'w')

71     fout0.write('%d\n'%(idx_city))
    fout1.write('%d\n'%51)
    fout2.write('%d\n'%(idx_state))
    fout3.write('%d\n'%(idx_cat+1))
76     fout4.write('2\n')

    for item in item_f:
        fout0.write('%d\t%d:1\n'%(item, item_f[item][0]))
        fout1.write('%d\t%d:1\n'%((item, (item_f[item][1] - minv) / (maxv - minv) * 50)))
81     fout2.write('%d\t%d:1\n'%(item, item_f[item][2]))
        fout3.write('%d\t'%(item))
        weight = 1.8
        for c in item_f[item][3]:
            if len(item_f[item][3]) == 1:
                fout3.write('%d:0.3f\t'%(c, (1.0 / len(item_f[item][3])) * 0.5))
            else:
                fout3.write('%d:0.3f\t'%(c, (1.0 / len(item_f[item][3]) * weight) * 0.5))
                weight -= 1.0 / (len(item_f[item][3]) - 1)
        if len(item_f[item][3]) == 0:
91     fout3.write('%d:1'%idx_cat)
        fout3.write('\n')
        if item_f[item][4]:
            fout4.write('%d\t%d:1\n'%(item, 1))
        else:
96     fout4.write('%d\t%d:1\n'%(item, 0))

    fout0.close()
    fout1.close()
    fout2.close()
    fout3.close()
101    fout4.close()

```

- *SimpleLocationFeature.py*

Generate the business location feature from Latitude and Longitude. Just using k-means cluster.

```

import numpy
import json
import math
4 from scipy.cluster.vq import vq, kmeans, whiten, kmeans2

item_f = {}

```

```

    item_latitude=[]
9   idx_latitude={}
    item_longitude=[]
    idx_longitude={}
    item_location=[]
    idx_location={}

14
    feature_count1 = 200
    feature_count2 = 200
    feature_count3 = 200

19   maxv=0
    minv=100

    for line in open('../Data/yelp_training_set_business_s.json'):
        js = json.loads(line)
24     item_latitude.append(js['latitude'])
        item_longitude.append(js['longitude'])
        item_location.append((js['latitude'], js['longitude']))

    for line in open('../Data/yelp_test_set_business_s.json'):
29     js = json.loads(line)
        item_latitude.append(js['latitude'])
        item_longitude.append(js['longitude'])
        item_location.append((js['latitude'], js['longitude']))

34   print 'load_over!'

    data_latitude = whiten(item_latitude)
    c_latitude = kmeans(data_latitude,feature_count1)[0]
    cat_latitude = vq(data_latitude,c_latitude)[0]
39   for i in range(len(item_latitude)):
        idx_latitude[item_latitude[i]]=cat_latitude[i]

    print 'latitude_kmeans!'

44   data_longitude = whiten(item_longitude)
    c_longitude = kmeans(data_longitude,feature_count2)[0]
    cat_longitude = vq(data_longitude,c_longitude)[0]
    for i in range(len(item_longitude)):
        idx_longitude[item_longitude[i]]=cat_longitude[i]
49
    print 'longitude_kmeans!'

    data_location = whiten(item_location)
    c_location = kmeans(data_location,feature_count3)[0]
54   cat_location = vq(data_location,c_location)[0]
    for i in range(len(item_location)):
        idx_location[(item_location[i][0], item_location[i][1])]=cat_location[i]

    print 'location_kmeans!'

59   for line in open('../Data/yelp_training_set_business_s.json'):
        js = json.loads(line)
        bid = js['business_id']
        item_f[bid]=[idx_latitude[js['latitude']], idx_longitude[js['longitude']], idx_location[(js['latitude'], js['longitude'])]]

64   for line in open('../Data/yelp_test_set_business_s.json'):
        js = json.loads(line)
        bid = js['business_id']
        item_f[bid]=[idx_latitude[js['latitude']], idx_longitude[js['longitude']], idx_location[(js['latitude'], js['longitude'])]]

69

fout0 = open('../GlobalFeature/i/ItemLatitude.txt','w')
fout1 = open('../GlobalFeature/i/ItemLongitude.txt','w')
fout2 = open('../GlobalFeature/i/ItemLocation.txt','w')

```

```

74 fout0.write('%d\n'%feature_count1)
   fout1.write('%d\n'%feature_count2)
   fout2.write('%d\n'%feature_count3)

79 for item in item_f:
    fout0.write('%d\t%d:1\n'%(item,item_f[item][0]))
    fout1.write('%d\t%d:1\n'%(item,item_f[item][1]))
    fout2.write('%d\t%d:1\n'%(item,item_f[item][2]))

84 fout0.close()
   fout1.close()
   fout2.close()

• SimpleUserFeature.py
  Generate the user feature from [yelp_training_set_user.json] and [yelp_test_set_user.json] directly, such as user review
  count.

import numpy
import json
import math

4 item_f={}
  item_review_count={}

  maxv=0
9  minv=100

  for line in open('../Data/ yelp_training_set_user_s .json'):
      js = json.loads(line)
      bid = js['user_id']

14      log_review=math.log(js['review_count']+1,2)
      if log_review<minv:
          minv=log_review
      if log_review>maxv:
19          maxv=log_review

      item_f[bid]=[log_review]

  for line in open('../Data/ yelp_test_set_user_s .json'):
24      js = json.loads(line)
      bid = js['user_id']

      log_review=math.log(js['review_count']+1,2)
      if log_review<minv:
29          minv=log_review
      if log_review>maxv:
          maxv=log_review

      item_f[bid]=[log_review]

34 fout1 = open('../GlobalFeature/u/UserReview.txt','w')

   fout1.write('%d\n'%51)

39 for item in item_f:
    fout1.write('%d\t%d:1\n'%((item,(item_f[item][0]-minv)/(maxv-minv)*50)))

   fout1.close()

• svdpp.py
  Generate SVD++ Feature.

import numpy
import json

3 user Rated={}
  maxv=0

```

```

for line in open('../Data/yelp_training_set_review_s.json'):
8     js = json.loads(line)
    if js['business_id'] > maxv:
        maxv = js['business_id']
    if js['user_id'] in user_rated:
        user_rated[js['user_id']].append(js['business_id'])
13 else:
    user_rated[js['user_id']] = [js['business_id']]

print maxv

18 for line in open('../Data/yelp_test_set_review_s.json'):
    js = json.loads(line)
    if js['business_id'] > maxv:
        maxv = js['business_id']
    if js['user_id'] in user_rated:
23     user_rated[js['user_id']].append(js['business_id'])
    else:
        user_rated[js['user_id']] = [js['business_id']]

print maxv

28 fout = open('../GlobalFeature/u/UserSVDpp.txt', 'w')

fout.write('%d\n' % (maxv + 1))
for user in user_rated:
33     fout.write('%d\t' % (user))
    for t in user_rated[user]:
        fout.write('%d:%f\t' % (t, 1.0 / len(user_rated[user]))) # (t, 1)
    fout.write('\n')
fout.close()

```

- *svdppr.py*

Generate Reverse SVD++ Feature.

```

import numpy
import json

3 business_rated = {}
maxv = 0

for line in open('../Data/yelp_training_set_review_s.json'):
8     js = json.loads(line)
    if js['user_id'] > maxv:
        maxv = js['user_id']
    if js['business_id'] in business_rated:
        business_rated[js['business_id']].append(js['user_id'])
13 else:
    business_rated[js['business_id']] = [js['user_id']]

print maxv

18 for line in open('../Data/yelp_test_set_review_s.json'):
    js = json.loads(line)
    if js['user_id'] > maxv:
        maxv = js['user_id']
    if js['business_id'] in business_rated:
23     business_rated[js['business_id']].append(js['user_id'])
    else:
        business_rated[js['business_id']] = [js['user_id']]

print maxv

28 fout = open('../GlobalFeature/i/businessSVDpp.txt', 'w')

fout.write('%d\n' % (maxv + 1))
for business in business_rated:

```

```

33     fout.write('%d\t'%(business))
        for t in business_rated[business]:
            fout.write('%d:%f\t'%(t,(1.0/len(business_rated[business]))))
        fout.write('\n')
    fout.close()

```

- *UserAvgStar.py*

User average star feature.

```

import numpy
import json
3 import math

item_f={}

item_star={}
8 idx_star=0

maxv=0
minv=100

13 for line in open('../Data/yelp_training_set_user_s.json'):
    js = json.loads(line)
    bid = js['user_id']
    item_f[bid]=js['average_stars']

18 fout0 = open('../GlobalFeature/u/UserAvgStar.txt','w')

    fout0.write('%d\n'%(2))

    for item in range(51296):
23         if item in item_f:
            fout0.write('%d\t0:%f\n'%(item,item_f[item]/5))
        else:
            fout0.write('%d\t1:1\n'%(item))

28 fout0.close()

```

- *UserCat.py*

The business categories user rate.

```

import numpy
2 import json
import math

item_f={}

7 item_categories={}
    idx_cat=0

    maxv=0
    minv=100

12 def WordCount( wlist ):
    temp = {}
    for word in wlist:
        temp[word]=temp.get(word,0)+1
17 temp = sorted(temp.items(),key = lambda x:x[1],reverse = True)
    return temp[1:]

for line in open('../Data/yelp_training_set_business_s.json'):
    js = json.loads( line )
22 bid = js['business_id']
    for c in js['categories']:
        if c not in item_categories:
            item_categories[c]=idx_cat
            idx_cat+=1
27 item_f[bid]=[item_categories[c] for c in js['categories']]

```



```

print 'categories:%d'%idx_cat

for line in open('../Data/yelp_test_set_business_s.json'):
32     js = json.loads(line)
    bid = js['business_id']
    for c in js['categories']:
        if c not in item_categories:
            item_categories[c]=idx_cat
37         idx_cat+=1
    item_f[bid]=[item_categories[c] for c in js['categories']]

print 'categories:%d'%idx_cat

42 UserCate = {}

for line in open('../Data/yelp_training_set_review_s.json'):
    js = json.loads(line)
    uid = js['user_id']
47     bid = js['business_id']
    cid = item_f[bid]
    if uid in UserCate:
        UserCate[uid]+=cid
    else:
52         UserCate[uid]=cid[:]

for line in open('../Data/yelp_test_set_review_s.json'):
    js = json.loads(line)
    uid = js['user_id']
57     bid = js['business_id']
    cid = item_f[bid]
    if uid in UserCate:
        UserCate[uid]+=cid
    else:
62         UserCate[uid]=cid[:]

fout3 = open('../GlobalFeature/u/UserCategories.txt','w')

fout3.write('%d\n'%(idx_cat))

67 for user in UserCate:
    Top = WordCount(UserCate[user])
    fout3.write('%d\t%s\n'%(user,'\t'.join(['%d:1'%x[0] for x in Top])))

72 fout3.close()

```

- *UserGender.py*

User gender feature.

```

import numpy
import json

3 UserGenderMap={}
  NameGenderMap={}

for line in open('NameGender.txt'):
8     line = line.split('\t')
    NameGenderMap[line[0].lower()] = line[1].rstrip()

for line in open('../Data/yelp_training_set_user_s.json'):
    js = json.loads(line)
13     userid = js['user_id']
    username = js['name'].lower()
    if username in NameGenderMap:
        UserGenderMap[userid]=NameGenderMap[username]

18 for line in open('../Data/yelp_test_set_user_s.json'):
    js = json.loads(line)
    userid = js['user_id']

```

```

        username = js['name'].lower()
        if username in NameGenderMap:
23             UserGenderMap[user_id]=NameGenderMap[username]

fout = open('../GlobalFeature/u/UserGender.txt','w')
fout.write('%d\n'%(3))
for user in range(55965):
28     fout.write('%d\t'%(user))
        if user in UserGenderMap:
            if UserGenderMap[user]=='m':
                fout.write('0:1\n')
            elif UserGenderMap[user]=='f':
33                 fout.write('1:1\n')
        else:
            fout.write('2:1\n')
fout.close()

```

- *UserName.py*

User name feature.

```

import numpy
import json

4  UserMap={}
   NameMap={}

   idx=0

9  for line in open('../Data/yelp_training_set_user_s.json'):
       js = json.loads(line)
       user_id = js['user_id']
       username = js['name'].lower()
       if username not in NameMap:
14           NameMap[username]=idx
           idx+=1
       UserMap[user_id]=NameMap[username]

for line in open('../Data/yelp_test_set_user_s.json'):
19     js = json.loads(line)
       user_id = js['user_id']
       username = js['name'].lower()
       if username not in NameMap:
           NameMap[username]=idx
24         idx+=1
       UserMap[user_id]=NameMap[username]

fout = open('../GlobalFeature/u/UserName.txt','w')
fout.write('%d\n'%(idx))
29 for user in UserMap:
       fout.write('%d\t%d:1\n'%(user,UserMap[user]))
fout.close()

```

- *UserNameLen.py*

Length of user name feature.

```

import numpy
import json

4  UserMap={}
   NameMap={}

   idx=0

9  for line in open('../Data/yelp_training_set_user_s.json'):
       js = json.loads(line)
       user_id = js['user_id']
       username = len(js['name'].lower())
       if username not in NameMap:
14           NameMap[username]=idx

```

```

        idx+=1
        UserMap[userid]=NameMap[username]

for line in open('../Data/yelp_test_set_user_s.json'):
19     js = json.loads(line)
        userid = js['user_id']
        username = len(js['name']).lower()
        if username not in NameMap:
            NameMap[username]=idx
24         idx+=1
        UserMap[userid]=NameMap[username]

fout = open('../GlobalFeature/u/UserNameLen.txt','w')
fout.write('%d\n'%(idx))
29 for user in UserMap:
    fout.write('%d\t%d:1\n'%(user,UserMap[user]))
fout.close()

• makelibfm1.py
Make LibFM formate feature with bid and uid.

import sys
import os
import json

4
print '-----Make_LibFM_Feature-----'

info=open('Data/info.txt')
user_count=int(info.readline())
9 item_count=int(info.readline())
info.close()

num_factor = int(sys.argv[2])
num_round = int(sys.argv[3])

14
group_info=[]

#From directories load features
def AppendFromDir(path,base_count):
19     f_map={}
        feature_count=0
        for file in os.listdir(path):
            header=True
            base_count+=feature_count
24         for line in open(path+file):
            if header:
                feature_count=int(line)
                group_info.append(feature_count)
                header=False
29         else:
            line = line.split('\t')
            idx = int(line[0])
            flist = {}
            for f in line[1:]:
34                 if f.rstrip().lstrip()=="": continue
                x = f.split(':')
                x = [int(x[0]), float(x[1])]
                flist[x[0]+base_count]=x[1]
            if idx in f_map:
39                 f_map[idx].update(flist)
            else:
                f_map[idx]=flist
        return f_map,base_count+feature_count

44 def AppendFromDirUI(path,base_count):
    f_map={}
    feature_count=0
    for file in os.listdir(path):

```

```

header=True
base_count+=feature_count
49     for line in open(path+file):
        if header:
            feature_count=int(line)
            group_info.append(feature_count)
            header=False
54         else:
            line = line.split('\t')
            idx = (int(line[0]),int(line[1]))
            flist = {}
59             for f in line[2:]:
                if f.rstrip().lstrip()=="": continue
                x = f.split(':')
                x = [int(x[0]),float(x[1])]
                flist[x[0]+base_count]=x[1]
64             if idx in f_map:
                f_map[idx].update(flist)
            else:
                f_map[idx]=flist
        return f_map,base_count+feature_count
69
global_f=1
user_f=0
item_f=0

74 #Global Feature
Global_Feature_u,global_f=AppendFromDir('GlobalFeature/u/',user_count+item_count)
Global_Feature_i,global_f=AppendFromDir('GlobalFeature/i/',global_f)
Global_Feature_ui,global_f=AppendFromDirUI('GlobalFeature/ui/',global_f)
print 'Global_Feature_Count:%d'%global_f
79 #User Feature
User_Feature,global_f=AppendFromDir('UserFeature/',global_f)
print 'User_Feature_Count:%d'%global_f
#Item Feature
Item_Feature,global_f=AppendFromDir('ItemFeature/',global_f)
84 print 'Item_Feature_Count:%d'%global_f

global_f.c=0
user_f.c=1
item_f.c=1

89 global_f_list=[]
user_f_list=[]
item_f_list=[]

94 global_f.v=[]
user_f.v=[]
item_f.v=[]

ftrain = open('Result/'+sys.argv[1]+'/'+'global.train.libfm','w')
99 for line in open('Data/training_data.txt'):
    line = map(int,map(float,line.split('\t')))
    u,i,r=line
    #generate global feature
    global_f.c=len(Global_Feature_u.get(u,{}))+len(Global_Feature_i.get(i,{}))+len(Global_Feature_ui.get((u,i),{}))
104    global_f_list =Global_Feature_u.get(u,{}).keys()+Global_Feature_i.get(i,{}).keys()+Global_Feature_ui.get((u,i),{}).
        keys()
    global_f.v=Global_Feature_u.get(u,{}).values()+Global_Feature_i.get(i,{}).values()+Global_Feature_ui.get((u,i),{}).
        values()
    #generate user feature
    user_f.c=1+len(User_Feature.get(u,{}))
    user_f_list =[u]+User_Feature.get(u,{}).keys()
109    user_f.v=[1]+User_Feature.get(u,{}).values()
    #generate item feature
    item_f.c=1+len(Item_Feature.get(i,{}))
    item_f_list =[i + user_count]+Item_Feature.get(i,{}).keys()
    item_f.v=[1]+Item_Feature.get(i,{}).values()

```

```

114         ftrain.write('%d\'%(r))

        for j in range(user_f.c):
            ftrain.write('%d:%.3f\'%(user_f_list[j], user_f_v[j]))
119     for j in range(item_f.c):
            ftrain.write('%d:%.3f\%(item_f_list[j], item_f_v[j]))
        for j in range(global_f.c):
            ftrain.write('%d:%.3f\%(global_f_list[j], global_f_v[j]))
        ftrain.write('\n')

124     ftrain.close()

    print 'Generate_Training_Over.'

129     ftest = open('Result/'+sys.argv[1]+'global.test.libfm', 'w')
    for line in open('Data/testing.data.txt'):
        line = map(int, map(float, line.split('\t')))
        u, i, r = line
        #generate global feature
134     global_f.c = len(Global_Feature_u.get(u, {})) + len(Global_Feature_i.get(i, {})) + len(Global_Feature_ui.get((u, i), {}))
        global_f_list = Global_Feature_u.get(u, {}).keys() + Global_Feature_i.get(i, {}).keys() + Global_Feature_ui.get((u, i), {}).
            keys()
        global_f_v = Global_Feature_u.get(u, {}).values() + Global_Feature_i.get(i, {}).values() + Global_Feature_ui.get((u, i), {}).
            values()
        #generate user feature
        user_f.c = 1 + len(User_Feature.get(u, {}))
139     user_f_list = [u] + User_Feature.get(u, {}).keys()
        user_f_v = [1] + User_Feature.get(u, {}).values()
        #generate item feature
        item_f.c = 1 + len(Item_Feature.get(i, {}))
        item_f_list = [i + user_count] + Item_Feature.get(i, {}).keys()
144     item_f_v = [1] + Item_Feature.get(i, {}).values()

        ftest.write('%d\'%(r))

        for j in range(user_f.c):
149             ftest.write('%d:%.3f\%(user_f_list[j], user_f_v[j]))
        for j in range(item_f.c):
            ftest.write('%d:%.3f\%(item_f_list[j], item_f_v[j]))
        for j in range(global_f.c):
            ftest.write('%d:%.3f\%(global_f_list[j], global_f_v[j]))

154     ftest.write('\n')
    ftest.close()

    print 'Generate_Testing_Over.'

159     fmeta = open('Result/'+sys.argv[1]+'meta.txt', 'w')
    group_idx = 0

    for i in range(user_count):
164         fmeta.write('%d\n'%group_idx)
        group_idx += 1

    for i in range(item_count):
        fmeta.write('%d\n'%group_idx)
169     group_idx += 1

    for i in group_info:
        for j in range(i):
            fmeta.write('%d\n'%group_idx)
174         group_idx += 1
    fmeta.close()

    print 'Generate_Meta_Over.'

179 #Generate Run.bat

```

```

bat = open('Result/'+sys.argv[1]+'run.bat','w')

bat.write(r'''
    ../../ Lab/libfm.exe" -task r -train global.train.libfm -test global.test.libfm -dim '1,1,%d' -iter %d -method mcmc -
    out pred.txt
184 '''%(num_factor,num_round))
bat.write(r'''
python ../../ Lab/CatFile.py" ../../Data/Local/%1/Submit.txt" "StepResult/pred.txt"
''' )

189 bat.write('pause')
bat.close()

```

```

print '-----Make_libFM_Feature-----'

```

- *makelibfm2.py*

Make LibFM formate feature with uid but without bid.

```

import sys
import os
3 import json

print '-----Make_libFM_Feature-----'

info=open('Data/info.txt')
8 user_count=int(info.readline())
item_count=int(info.readline())
info.close()

num_factor = int(sys.argv[2])
13 num_round = int(sys.argv[3])

group_info=[]

#From directories load features
18 def AppendFromDir(path,base_count):
    f_map={}
    feature_count=0
    for file in os.listdir(path):
        header=True
        base_count+=feature_count
        23 for line in open(path+file):
            if header:
                feature_count=int(line)
                group_info.append(feature_count)
                header=False
            28 else:
                line = line.split('\t')
                idx = int(line[0])
                flist = {}
                33 for f in line[1:]:
                    if f.rstrip().lstrip()=="": continue
                    x = f.split(':')
                    x = [int(x[0]), float(x[1])]
                    flist[x[0]+base_count]=x[1]
                38 if idx in f_map:
                    f_map[idx].update(flist)
                else:
                    f_map[idx]=flist
            return f_map,base_count+feature_count
43

def AppendFromDirUI(path,base_count):
    f_map={}
    feature_count=0
    for file in os.listdir(path):
        header=True
        48 base_count+=feature_count
        for line in open(path+file):

```

```

        if header:
            feature_count=int(line)
            group_info.append(feature_count)
            header=False
        else :
            line = line.split('\t')
            idx = (int(line[0]),int(line[1]))
            flist = {}
            for f in line[2:]:
                if f.rstrip().lstrip()=="": continue
                x = f.split(':')
                x = [int(x[0]), float(x[1])]
                flist[x[0]+base_count]=x[1]
            if idx in f_map:
                f_map[idx].update(flist)
            else :
                f_map[idx]=flist
        return f_map,base_count+feature_count

global_f=0
user_f=0
item_f=0

73 #Global Feature
Global_Feature_u,global_f=AppendFromDir('GlobalFeature/u/',user_count)
Global_Feature_i,global_f=AppendFromDir('GlobalFeature/i/',global_f)
Global_Feature_ui,global_f=AppendFromDirUI('GlobalFeature/ui/',global_f)
78 print 'Global_Feature_Count:%d'%global_f
#User Feature
User_Feature,global_f=AppendFromDir('UserFeature/',global_f)
print 'User_Feature_Count:%d'%global_f
#Item Feature
83 Item_Feature,global_f=AppendFromDir('ItemFeature/',global_f)
print 'Item_Feature_Count:%d'%global_f

global_f.c=0
user_f.c=1
88 item_f.c=1

global_f.list=[]
user_f.list=[]
item_f.list=[]

93 global_f.v=[]
user_f.v=[]
item_f.v=[]

98 ftrain = open('Result/'+sys.argv[1]+'/'+'global.train.libfm','w')
for line in open('Data/training_data.txt'):
    line = map(int,map(float,line.split('\t')))
    u,i,r=line
    #generate global feature
    global_f.c=len(Global_Feature_u.get(u,{}))+len(Global_Feature_i.get(i,{}))+len(Global_Feature_ui.get((u,i),{}))
    global_f.list =Global_Feature_u.get(u,{}).keys()+Global_Feature_i.get(i,{}).keys()+Global_Feature_ui.get((u,i),{}).
        keys()
    global_f.v=Global_Feature_u.get(u,{}).values()+Global_Feature_i.get(i,{}).values()+Global_Feature_ui.get((u,i),{}).
        values()
    #generate user feature
    user_f.c=1+len(User_Feature.get(u,{}))
    user_f.list =[u]+User_Feature.get(u,{}).keys()
    user_f.v=[1]+User_Feature.get(u,{}).values()
    #generate item feature
    item_f.c=len(Item_Feature.get(i,{}))
    item_f.list =Item_Feature.get(i,{}).keys()
    item_f.v=Item_Feature.get(i,{}).values()

113 ftrain.write('%d_%d'%(r))

```

```

118     for j in range(user_f.c):
        ftrain.write('%d:%.3f'%(user_f.list[j], user_f.v[j]))
    for j in range(item_f.c):
        ftrain.write('%d:%.3f'%(item_f.list[j], item_f.v[j]))
    for j in range(global_f.c):
        ftrain.write('%d:%.3f'%(global_f.list[j], global_f.v[j]))
123     ftrain.write('\n')

ftrain.close()

print 'Generate_Training_Over.'

128     ftest = open('Result/'+sys.argv[1]+'/global.test.libfm', 'w')
    for line in open('Data/testing.data.txt'):
        line = map(int, map(float, line.split('\t')))
        u, i, r = line
133         #generate global feature
        global_f.c = len(Global_Feature.u.get(u, {})) + len(Global_Feature.i.get(i, {})) + len(Global_Feature.ui.get((u, i), {}))
        global_f.list = Global_Feature.u.get(u, {}).keys() + Global_Feature.i.get(i, {}).keys() + Global_Feature.ui.get((u, i), {}).keys()
        global_f.v = Global_Feature.u.get(u, {}).values() + Global_Feature.i.get(i, {}).values() + Global_Feature.ui.get((u, i), {}).values()
        #generate user feature
138         user_f.c = 1 + len(User_Feature.get(u, {}))
        user_f.list = [u] + User_Feature.get(u, {}).keys()
        user_f.v = [1] + User_Feature.get(u, {}).values()
        #generate item feature
        item_f.c = len(Item_Feature.get(i, {}))
143         item_f.list = Item_Feature.get(i, {}).keys()
        item_f.v = Item_Feature.get(i, {}).values()

        ftest.write('%d_%d'%(r))

148     for j in range(user_f.c):
        ftest.write('%d:%.3f'%(user_f.list[j], user_f.v[j]))
    for j in range(item_f.c):
        ftest.write('%d:%.3f'%(item_f.list[j], item_f.v[j]))
    for j in range(global_f.c):
153         ftest.write('%d:%.3f'%(global_f.list[j], global_f.v[j]))

    ftest.write('\n')
    ftest.close()

158     print 'Generate_Testing_Over.'

    fmeta = open('Result/'+sys.argv[1]+'meta.txt', 'w')
    group_idx = 0

163     for i in range(user_count):
        fmeta.write('%d\n'%group_idx)
        group_idx += 1

    for i in range(item_count):
168         fmeta.write('%d\n'%group_idx)
        group_idx += 1

    for i in group_info:
        for j in range(i):
173             fmeta.write('%d\n'%group_idx)
            group_idx += 1
    fmeta.close()

    print 'Generate_Meta_Over.'

178     #Generate Run.bat
    bat = open('Result/'+sys.argv[1]+'run.bat', 'w')
    bat.write(r'''
    ..../ Lab/libfm.exe" -task r -train global.train.libfm -test global.test.libfm -dim '1,1,%d' -iter %d -method sgd -

```



```

learn_rate 0.001 --regular '0,0,5' --init_stdev 0.05 --out StepResult/pred.txt
183 ''' %(num_factor,num_round))

bat.write(r'''
python "../Lab/CatFile.py" "../Data/Local/%1/Submit.txt" "StepResult/pred.txt"
''' )

188 bat.close ()

print '-----Make_libFM_Feature-----'

```

- *makelibfm3.py*

Make LibFM formate feature with bid but without uid.

```

import sys
import os
import json

4 print '-----Make_libFM_Feature-----'

info=open('Data/info.txt')
user_count=int(info.readline())
9 item_count=int(info.readline())
info.close()

num_factor = int(sys.argv[2])
num_round = int(sys.argv[3])

14 group_info=[]

#From directories load features
def AppendFromDir(path,base_count):
19     f_map={}
    feature_count=0
    for file in os.listdir (path):
        header=True
        base_count+=feature_count
24     for line in open(path+file):
        if header:
            feature_count=int(line)
            group_info.append(feature_count)
            header=False
29     else :
        line = line.split ('\t')
        idx = int( line [0])
        flist = {}
        for f in line [1:]:
34             if f.rstrip () .lstrip ()=="": continue
            x = f.split (':')
            x = [int (x [0]) , float (x [1]) ]
            flist [x[0]+base_count]=x[1]
        if idx in f_map:
39             f_map[idx].update( flist )
        else :
            f_map[idx]= flist

    return f_map,base_count+feature_count

44 def AppendFromDirUI(path,base_count):
    f_map={}
    feature_count=0
    for file in os.listdir (path):
        header=True
        base_count+=feature_count
49     for line in open(path+file):
        if header:
            feature_count=int(line)
            group_info.append(feature_count)
54             header=False

```

```

else :
    line = line.split('\t')
    idx = (int(line[0]),int(line[1]))
    flist = {}
59     for f in line[2:]:
        if f.rstrip().lstrip()=="": continue
        x = f.split(':')
        x = [int(x[0]),float(x[1])]
        flist[x[0]+base_count]=x[1]
64     if idx in f_map:
        f_map[idx].update(flist)
    else :
        f_map[idx]=flist
    return f_map,base_count+feature_count
69
global_f=0
user_f=0
item_f=0

74 #Global Feature
Global_Feature_u,global_f=AppendFromDir('GlobalFeature/u/',item_count)
Global_Feature_i,global_f=AppendFromDir('GlobalFeature/i/',global_f)
Global_Feature_ui,global_f=AppendFromDirUI('GlobalFeature/ui/',global_f)
print 'Global_Feature_Count:%d'%global_f
79 #User Feature
User_Feature,global_f=AppendFromDir('UserFeature/',global_f)
print 'User_Feature_Count:%d'%global_f
#Item Feature
Item_Feature,global_f=AppendFromDir('ItemFeature/',global_f)
84 print 'Item_Feature_Count:%d'%global_f

global_f_c=0
user_f_c=1
item_f_c=1

89
global_f_list=[]
user_f_list=[]
item_f_list=[]

94 global_f_v=[]
user_f_v=[]
item_f_v=[]

ftrain = open('Result/'+sys.argv[1]+'/'+'global.train.libfm','w')
99 for line in open('Data/training_data.txt'):
    line = map(int,map(float,line.split('\t')))
    u,i,r=line
    #generate global feature
    global_f_c=len(Global_Feature_u.get(u,{}))+len(Global_Feature_i.get(i,{}))+len(Global_Feature_ui.get((u,i),{}))
104    global_f_list =Global_Feature_u.get(u,{}).keys()+Global_Feature_i.get(i,{}).keys()+Global_Feature_ui.get((u,i),{}).
        keys()
    global_f_v=Global_Feature_u.get(u,{}).values()+Global_Feature_i.get(i,{}).values()+Global_Feature_ui.get((u,i),{}).
        values()
    #generate user feature
    user_f_c=len(User_Feature.get(u,{}))
    user_f_list =User_Feature.get(u,{}).keys()
109    user_f_v=User_Feature.get(u,{}).values()
    #generate item feature
    item_f_c=1+len(Item_Feature.get(i,{}))
    item_f_list =[i]+Item_Feature.get(i,{}).keys()
    item_f_v=[1]+Item_Feature.get(i,{}).values()
114
    ftrain.write('%d\'%(r))

    for j in range(user_f_c):
        ftrain.write('%d:%.3f\'%(user_f_list[j], user_f_v[j]))
119    for j in range(item_f_c):
        ftrain.write('%d:%.3f\%(item_f_list[j], item_f_v[j]))

```

```

        for j in range(global_f.c):
            ftrain.write('%d:%.3f'%(global_f.list[j], global_f.v[j]))
        ftrain.write('\n')
124 ftrain.close()

print 'Generate_Training_Over.'

129 ftest = open('Result/'+sys.argv[1]+'/global.test.libfm', 'w')
for line in open('Data/testing_data.txt'):
    line = map(int, map(float, line.split('\t')))
    u, i, r = line
    #generate global feature
134 global_f.c = len(Global_Feature.u.get(u, {})) + len(Global_Feature.i.get(i, {})) + len(Global_Feature.ui.get((u, i), {}))
    global_f.list = Global_Feature.u.get(u, {}).keys() + Global_Feature.i.get(i, {}).keys() + Global_Feature.ui.get((u, i), {}).
        keys()
    global_f.v = Global_Feature.u.get(u, {}).values() + Global_Feature.i.get(i, {}).values() + Global_Feature.ui.get((u, i), {}).
        values()
    #generate user feature
    user_f.c = len(User_Feature.get(u, {}))
139 user_f.list = User_Feature.get(u, {}).keys()
    user_f.v = User_Feature.get(u, {}).values()
    #generate item feature
    item_f.c = 1 + len(Item_Feature.get(i, {}))
    item_f.list = [i] + Item_Feature.get(i, {}).keys()
144 item_f.v = [1] + Item_Feature.get(i, {}).values()

    ftest.write('%dL'%(r))

    for j in range(user_f.c):
149 ftest.write('%d:%.3f'%(user_f.list[j], user_f.v[j]))
    for j in range(item_f.c):
        ftest.write('%d:%.3f'%(item_f.list[j], item_f.v[j]))
    for j in range(global_f.c):
        ftest.write('%d:%.3f'%(global_f.list[j], global_f.v[j]))
154 ftest.write('\n')
ftest.close()

print 'Generate_Testing_Over.'

159 fmeta = open('Result/'+sys.argv[1]+'meta.txt', 'w')
group_idx = 0

for i in range(user_count):
164 fmeta.write('%d\n'%group_idx)
    group_idx += 1

for i in range(item_count):
    fmeta.write('%d\n'%group_idx)
169 group_idx += 1

for i in group_info:
    for j in range(i):
        fmeta.write('%d\n'%group_idx)
174 group_idx += 1
fmeta.close()

print 'Generate_Meta_Over.'

179 #Generate Run.bat
bat = open('Result/'+sys.argv[1]+'run.bat', 'w')
bat.write(r'''
'''../Lab/libfm.exe'' -task r -train global.train.libfm -test global.test.libfm -dim '1,1,%d' -iter %d -method sgd -
    learn_rate 0.001 -regular '0,0,5' -init_stdev 0.05 -out StepResult/pred.txt
'''%(num_factor, num_round))
184 bat.write(r'''

```

```
python "../Lab/CatFile.py" "../Data/Local/%1/Submit.txt" "StepResult/pred.txt"
'''
)
#bat.write('pause')
```

189

```
bat.close()

print '-----Make_libFM_Feature-----'
```

- *makelibfm4.py*

Make LibFM format feature with out bid and uid.

```
import sys
import os
import json

print '-----Make_libFM_Feature-----'
```

```
info=open('Data/info.txt')
8 user_count=int(info.readline())
  item_count=int(info.readline())
  info.close()

num_factor = int(sys.argv[2])
13 num_round = int(sys.argv[3])

group_info=[]
```

#From directories load features

```
18 def AppendFromDir(path,base_count):
    f_map={}
    feature_count=0
    for file in os.listdir(path):
        header=True
23         base_count+=feature_count
        for line in open(path+file):
            if header:
                feature_count=int(line)
                group_info.append(feature_count)
                header=False
28             else:
                line = line.split('\t')
                idx = int(line[0])
                flist = {}
33                 for f in line[1:]:
                    if f.rstrip().lstrip()=="": continue
                    x = f.split(':')
                    x = [int(x[0]), float(x[1])]
                    flist[x[0]+base_count]=x[1]
38                 if idx in f_map:
                    f_map[idx].update(flist)
                else:
                    f_map[idx]=flist

    return f_map,base_count+feature_count
```

43

```
def AppendFromDirUI(path,base_count):
    f_map={}
    feature_count=0
    for file in os.listdir(path):
        header=True
48         base_count+=feature_count
        for line in open(path+file):
            if header:
                feature_count=int(line)
                group_info.append(feature_count)
                header=False
53             else:
                line = line.split('\t')
                idx = (int(line[0]), int(line[1]))
```

```

58         flist = {}
        for f in line [2:]:
            if f.rstrip().lstrip()=="": continue
            x = f.split(':')
            x = [int(x[0]), float(x[1])]
63         flist [x[0]+base_count]=x[1]
        if idx in f_map:
            f_map[idx].update( flist )
        else :
            f_map[idx]= flist
68     return f_map,base_count+feature_count

    global_f=0
    user_f=0
    item_f=0

73     #Global Feature
    Global_Feature_u,global_f=AppendFromDir('GlobalFeature/u/',0)
    Global_Feature_i, global_f=AppendFromDir('GlobalFeature/i/',global_f)
    Global_Feature_ui, global_f=AppendFromDirUI('GlobalFeature/ui/',global_f)
78     print 'Global_Feature_Count:%d'%global_f
    #User Feature
    User_Feature,global_f=AppendFromDir('UserFeature/',global_f)
    print 'User_Feature_Count:%d'%global_f
    #Item Feature
83     Item_Feature,global_f=AppendFromDir('ItemFeature/',global_f)
    print 'Item_Feature_Count:%d'%global_f

    global_f_c=0
    user_f_c=1
88     item_f_c=1

    global_f_list =[]
    user_f_list =[]
    item_f_list =[]

93     global_f_v=[]
    user_f_v=[]
    item_f_v=[]

98     ftrain = open('Result/'+sys.argv[1]+'/global.train.libfm', 'w')
    for line in open('Data/training_data.txt'):
        line = map(int,map(float,line.split('\t')))
        u,i,r=line
        #generate global feature
103     global_f_c=len(Global_Feature_u.get(u,{}))+len(Global_Feature_i.get(i,{}))+len(Global_Feature_ui.get((u,i),{}))
        global_f_list =Global_Feature_u.get(u,{}).keys()+Global_Feature_i.get(i,{ }).keys()+Global_Feature_ui.get((u,i),{ }).
            keys()
        global_f_v=Global_Feature_u.get(u,{ }).values()+Global_Feature_i.get(i,{ }).values()+Global_Feature_ui.get((u,i),{ }).
            values()
        #generate user feature
        user_f_c=len(User_Feature.get(u,{ }))
108     user_f_list =User_Feature.get(u,{ }).keys()
        user_f_v=User_Feature.get(u,{ }).values()
        #generate item feature
        item_f_c=len(Item_Feature.get(i,{ }))
        item_f_list =Item_Feature.get(i,{ }).keys()
113     item_f_v=Item_Feature.get(i,{ }).values()

        ftrain.write('%d_%d'%(r))

        for j in range(user_f_c):
118             ftrain.write('%d:%.3f_%d'%(user_f_list[j], user_f_v[j]))
        for j in range(item_f_c):
            ftrain.write('%d:%.3f_%d'%(item_f_list[j], item_f_v[j]))
        for j in range(global_f_c):
            ftrain.write('%d:%.3f_%d'%(global_f_list[j], global_f_v[j]))
123     ftrain.write('\n')

```

```

ftrain . close ()

print 'Generate_Training_Over.'

128 ftest = open('Result/'+sys.argv[1]+'/global.test.libfm','w')
for line in open('Data/testing.data.txt'):
    line = map(int,map(float,line.split('\t')))
    u,i,r=line
133 #generate global feature
    global_f_c=len(Global_Feature_u.get(u,{}))+len(Global_Feature_i.get(i,{}))+len(Global_Feature_ui.get((u,i),{}))
    global_f_list =Global_Feature_u.get(u,{}).keys()+Global_Feature_i.get(i,{ }).keys()+Global_Feature_ui.get((u,i),{}).
        keys()
    global_f_v=Global_Feature_u.get(u,{ }).values()+Global_Feature_i.get(i,{ }).values()+Global_Feature_ui.get((u,i),{}).
        values()
    #generate user feature
138 user_f_c=len(User_Feature.get(u,{ }))
    user_f_list =User_Feature.get(u,{ }).keys()
    user_f_v=User_Feature.get(u,{ }).values()
    #generate item feature
    item_f_c=len(Item_Feature.get(i,{ }))
143 item_f_list =Item_Feature.get(i,{ }).keys()
    item_f_v=Item_Feature.get(i,{ }).values()

    ftest . write('%d\'%(r))

148 for j in range(user_f_c):
    ftest . write('%d:%.3f\'%(user_f_list[j], user_f_v[j]))
for j in range(item_f_c):
    ftest . write('%d:%.3f\%(item_f_list[j], item_f_v[j]))
for j in range(global_f_c):
153 ftest . write('%d:%.3f\%(global_f_list[j], global_f_v[j]))

    ftest . write('\n')
ftest . close ()

158 print 'Generate_Testing_Over.'

fmeta = open('Result/'+sys.argv[1]+'meta.txt','w')
group_idx=0

163 for i in range(user_count):
    fmeta.write('%d\n'%group_idx)
    group_idx+=1

for i in range(item_count):
168 fmeta.write('%d\n'%group_idx)
    group_idx+=1

for i in group_info:
    for j in range(i):
173 fmeta.write('%d\n'%group_idx)
        group_idx+=1
fmeta.close ()

print 'Generate_Meta_Over.'

178 #Generate Run.bat
bat = open('Result/'+sys.argv[1]+'run.bat','w')
bat.write(r'''
    ../../ Lab/libfm.exe" -task r -train global.train.libfm -test global.test.libfm -dim '1,1,%d' -iter %d -method sgd -
        learn_rate 0.001 -regular '0,0,5' -init_stdev 0.05 -out StepResult/pred.txt
183 ''' %(num_factor,num_round))

bat.write(r'''
python ../../ Lab/CatFile.py" ../../Data/Local/%1/Submit.txt" "StepResult/pred.txt"
''' )
188 #bat.write('pause')

```

```
bat.close()
```

```
print '-----Make_libFM_Feature-----'
```

- *makesvdfeature.py*

Make SVDFeature formate feature with uid and bid.

```
import sys
```

```
import os
```

```
3 import json
```

```
print '-----Make_svdFeature_Feature-----'
```

```
info=open('Data/info.txt')
```

```
8 user_count=int(info.readline())
```

```
item_count=int(info.readline())
```

```
info.close()
```

```
num_factor = int(sys.argv[2])
```

```
13 num_round = int(sys.argv[3])
```

```
def AppendFromDir(path,base_count):
```

```
    f_map={}
```

```
    feature_count=0
```

```
18 for file in os.listdir(path):
```

```
    header=True
```

```
    base_count+=feature_count
```

```
    for line in open(path+file):
```

```
        if header:
```

```
23         feature_count=int(line)
```

```
        header=False
```

```
    else:
```

```
        line = line.split('\t')
```

```
        idx = int(line[0])
```

```
28         flist = {}
```

```
        for f in line[1:]:
```

```
            if f.rstrip().lstrip()=="": continue
```

```
            x = f.split(':')
```

```
            x = [int(x[0]), float(x[1])]
```

```
33             flist[x[0]+base_count]=x[1]
```

```
        if idx in f_map:
```

```
            f_map[idx].update(flist)
```

```
        else:
```

```
            f_map[idx]=flist
```

```
38 return f_map,base_count+feature_count
```

```
def AppendFromDirUI(path,base_count):
```

```
    f_map={}
```

```
    feature_count=0
```

```
43 for file in os.listdir(path):
```

```
    header=True
```

```
    base_count+=feature_count
```

```
    for line in open(path+file):
```

```
        if header:
```

```
48         feature_count=int(line)
```

```
        header=False
```

```
    else:
```

```
        line = line.split('\t')
```

```
        idx = (int(line[0]),int(line[1]))
```

```
53         flist = {}
```

```
        for f in line[2:]:
```

```
            if f.rstrip().lstrip()=="": continue
```

```
            x = f.split(':')
```

```
            x = [int(x[0]), float(x[1])]
```

```
58             flist[x[0]+base_count]=x[1]
```

```
        if idx in f_map:
```

```
            f_map[idx].update(flist)
```

```

else :
    f_map[idx]= flist
63     return f_map,base_count+feature_count

global f=0
user_f=0
item_f=0
68
#Global Feature
Global_Feature_u,global_f=AppendFromDir('GlobalFeature/u/',0)
Global_Feature_i, global_f=AppendFromDir('GlobalFeature/i/',global_f)
Global_Feature_ui, global_f=AppendFromDirUI('GlobalFeature/ui/',global_f)
73 print 'Global_Feature_Count:%d'%global_f
#User Feature
User_Feature,user_f=AppendFromDir('UserFeature/',user_count)
print 'User_Feature_Count:%d'%user_f
#Item Feature
78 Item_Feature,item_f=AppendFromDir('ItemFeature/',item_count)
print 'Item_Feature_Count:%d'%item_f

global f_c=0
user_f_c=1
83 item_f_c=1

global f_list=[]
user_f_list=[]
item_f_list=[]
88
global f_v=[]
user_f_v=[]
item_f_v=[]

93 ftrain = open('Result/'+sys.argv[1]+'/'+'global.train.txt', 'w')
for line in open('Data/training_data.txt'):
    line = map(int,map(float,line.split(' ')))
    u,i,r=line
    #generate global feature
98     global_f_c=len(Global_Feature_u.get(u,{}))+len(Global_Feature_i.get(i,{}))+len(Global_Feature_ui.get((u,i),{}))
    global_f_list =Global_Feature_u.get(u,{}).keys()+Global_Feature_i.get(i,{}).keys()+Global_Feature_ui.get((u,i),{}).
        keys()
    global_f_v=Global_Feature_u.get(u,{}).values()+Global_Feature_i.get(i,{}).values()+Global_Feature_ui.get((u,i),{}).
        values()
    #generate user feature
    user_f_c=1+len(User_Feature.get(u,{}))
103     user_f_list =[u]+User_Feature.get(u,{}).keys()
    user_f_v=[1]+User_Feature.get(u,{}).values()
    #generate item feature
    item_f_c=1+len(Item_Feature.get(i,{}))
    item_f_list =[i]+Item_Feature.get(i,{}).keys()
108     item_f_v=[1]+Item_Feature.get(i,{}).values()

    ftrain.write('%d_%d_%d_%d'% (r,global_f_c,user_f_c,item_f_c))
    for j in range(global_f_c):
        ftrain.write('%d:%.8f'%(global_f_list[j], global_f_v[j]))
113     for j in range(user_f_c):
        ftrain.write('%d:%.8f'%(user_f_list[j], user_f_v[j]))
    for j in range(item_f_c):
        ftrain.write('%d:%.8f'%(item_f_list[j], item_f_v[j]))

118     ftrain.write('\n')

ftrain.close()

print 'Generate_Training_Over.'
123
ftest = open('Result/'+sys.argv[1]+'/'+'global.test.txt', 'w')
for line in open('Data/testing_data.txt'):
    line = map(int,map(float,line.split(' ')))

```



```

128     u,i,r=line
    #generate global feature
    global_f.c=len(Global_Feature_u.get(u,{}))+len(Global_Feature_i.get(i,{}))+len(Global_Feature_ui.get((u,i),{}))
    global_f.list =Global_Feature_u.get(u,{}).keys()+Global_Feature_i.get(i,{}).keys()+Global_Feature_ui.get((u,i),{}).
        keys()
    global_f.v=Global_Feature_u.get(u,{}).values()+Global_Feature_i.get(i,{}).values()+Global_Feature_ui.get((u,i),{}).
        values()
    #generate user feature
133     user_f.c=1+len(User_Feature.get(u,{}))
    user_f.list =[u]+User_Feature.get(u,{}).keys()
    user_f.v=[1]+User_Feature.get(u,{}).values()
    #generate item feature
    item_f.c=1+len(Item_Feature.get(i,{}))
138     item_f.list =[i]+Item_Feature.get(i,{}).keys()
    item_f.v=[1]+Item_Feature.get(i,{}).values()

    ftest.write('%d_%d_%d_%d'%(r,global_f.c,user_f.c,item_f.c))
    for j in range(global_f.c):
143         ftest.write('%d:%.8f'%(global_f.list[j], global_f.v[j]))
    for j in range(user_f.c):
        ftest.write('%d:%.8f'%(user_f.list[j], user_f.v[j]))
    for j in range(item_f.c):
        ftest.write('%d:%.8f'%(item_f.list[j], item_f.v[j]))

148     ftest.write('\n')
    ftest.close()

    print 'Generate_Testing_Over.'

153     os.mkdir('Result/'+sys.argv[1]+'/'+'model')

    #Generate Run.bat
    bat = open('Result/'+sys.argv[1]+'/'+'run.bat','w')
158     bat.write(r'''
    ../../ Lab/make_feature_buffer.exe" global.train.txt global.train.buffer
    ../../ Lab/make_feature_buffer.exe" global.test.txt global.test.buffer

    ../../ Lab/svdfeature.exe" basicMF.conf num_round=%d

163     '''%(num_round))

    bat.write(' ../../ Lab/svdfeature_infer.exe" _basicMF.conf\n')

168     bat.write('pause')

    bat.close()

    print '-----Make_svdFeature_Feature-----'

```

- *BiasMF.cpp*

Bias FM of C++ version using opencv matrix lib.

```

#include <iostream>
#include <iomanip>
3  #include <fstream>
#include <sstream>
#include <vector>
#include <time.h>
#include <opencv2\opencv.hpp>

8
using namespace std;
using namespace cv;

typedef struct _TData
13 {
    int x;
    int y;
    double r;

```

```

}TData;
18
int f=1;
int dims=2;
int size[]={55965,14334};://{51296,12742};
int usize[]={size[0], f};
23 int isize[]={size[1], f};
int mu_usize[]={size[0]};
int mu_isize[]={size[1]};
vector<TData> RattingData;
vector<TData> TestData;
28 Mat U(dims,usize,CV_64F);
Mat I(dims,isize,CV_64F);
Mat U_mu(1,mu_usize,CV_64F);
Mat I_mu(1,mu_isize,CV_64F);
Mat U_c(1,mu_usize,CV_32S);
33 Mat I_c(1,mu_isize,CV_32S);
double eta=0.0008;
double lambda=0.5;
int MaxIter=200;

38 double mu=0.0;

void ImportData()
{
    ifstream trainfile ("training_data.txt");
43    TData t;
    while(! trainfile .eof())
    {
        trainfile >>t.x>>t.y>>t.r;
        RattingData.push_back(t);
48        mu+=t.r;
    }
    trainfile .close();
    cout<<"Training_Data_Size:_"<<RattingData.size()<<endl;

53    mu/=RattingData.size();

    ifstream testfile ("testing_data.txt");
    while(! testfile .eof())
    {
58        testfile >>t.x>>t.y>>t.r;
        TestData.push_back(t);
    }
    testfile .close();
    cout<<"Test_Data_Size:_"<<TestData.size()<<endl;

63    cout<<"Global_Mean:_"<<mu<<"_Lambda:_"<<lambda<<"_Eta:_"<<eta<<endl;
}

void WritePredictFile()
68 {
    ofstream f("predict");
    double rmse=0.0;
    for (vector<TData>::iterator iter=TestData.begin();iter!=TestData.end();iter++)
    {
73        double r_predict=mu+U_mu.at<double>(iter->x)+I_mu.at<double>(iter->y)+((Mat)(U.row(iter->x)*I.
            row(iter->y).t()))>(0,0);
        if (r_predict<1) r_predict=1;
        if (r_predict>5) r_predict=5;
        f<<r_predict<<endl;
        double error=r_predict-iter->r;
78        rmse+=error*error;
    }
    rmse=sqrt(rmse/TestData.size());
    f.close();
    cout<<"Predict_File_Generate_Over_Final_rmse:_"<<rmse<<endl;
83 }

```

```

double CalcTestRmse()
{
    double rmse=0.0;
88     for (vector<TData>::iterator iter=TestData.begin();iter!=TestData.end();iter++)
    {
        double r_predict=mu+U_mu.at<double>(iter->x)+I_mu.at<double>(iter->y)+((Mat)(U.row(iter->x)*I.
            row(iter->y).t()))).at<double>(0,0);
        if ( r_predict<1) r_predict=1;
        if ( r_predict>5) r_predict=5;
93     double error=r_predict-iter->r;
        rmse+=error*error;
    }
    rmse=sqrt(rmse/TestData.size());
    return rmse;
98 }

float BasicMF()
{
    ofstream rmse_file ("rmse.txt");
103

    double rmse=0.0;
    double pre_rmse=100.0;

    randu(U,Scalar(0),Scalar(1/sqrt((double)f)));
108    randu(I,Scalar(0),Scalar(1/sqrt((double)f)));
    randu(U_mu,Scalar(0),Scalar(0));
    randu(I_mu,Scalar(0),Scalar(0));

    for (int t=0;t<MaxIter;t++)
113    {
        clock_t start=clock();
        for (vector<TData>::iterator iter=RattingData.begin();iter!=RattingData.end();iter++)
        {
            double u_mu=U_mu.at<double>(iter->x);
            double i_mu=I_mu.at<double>(iter->y);
118            double r_predict=mu+u_mu+i_mu+((Mat)(U.row(iter->x)*I.row(iter->y).t()))).at<double>(0,0);
            double error=r_predict-iter->r;

            for (int k=0;k<f;k++)
123            {
                double u=U.at<double>(iter->x,k);
                double i=I.at<double>(iter->y,k);
                U.at<double>(iter->x,k)=u-eta*(error*i+lambda*u);
                I.at<double>(iter->y,k)=i-eta*(error*u+lambda*i);
128            }
            U_mu.at<double>(iter->x)=u_mu-eta*(error+lambda*u_mu);
            I_mu.at<double>(iter->y)=i_mu-eta*(error+lambda*i_mu);

            rmse+=error*error;
133        }
        rmse=sqrt(rmse/RattingData.size());
        clock_t end=clock();
        double rmse_test=CalcTestRmse();
        cout<<"Iter:_"<<std::setw(3)<<t<<"\tRMSE:_"<<rmse<<"\tTest_RMSE:_"<<rmse_test<<"\tTime:_"
            <<((double)end-start)/CLOCKS_PER_SEC <<"s"<<endl;
138

        rmse_file <<rmse<<"\t"<<rmse_test<<endl;
    }
    rmse_file . close ();
    return 0;
143 }

void WriteMatToFile(char * filename,Mat &m)
{
    ofstream of(filename);
148    for (int i=0;i<m.cols;i++)
    {

```

```

        for (int j=0;j<m.rows;j++)
        {
            of<<m.at<double>(i,j)<<"\t";
153     }
        of<<endl;
    }
    of.close();
}

158 void WriteFeatureToFile(char * filename,Mat &A,Mat &B)
{
    ofstream of(filename);
    for (int i=0;i<A.cols;i++)
163     {
        int j=0;
        for (j=0;j<A.rows;j++)
        {
            of<<A.at<double>(j,i)<<"\t";
168     }
        for (j=0;j<B.rows;j++)
        {
            of<<B.at<double>(j,i)<<"\t";
173     }
        of<<endl;
    }
    of.close();
}

178 int main(int argc,char * argv[])
{
    if (argc>1)
    {
        183     istream para(argv[1]);
        para>>f;
    }
    ImportData();

    BasicMF();

188     WriteFeatureToFile("v",U,I);
    WriteFeatureToFile("w",U_mu,I_mu);

    WritePredictFile();

193     getchar();
    return 0;
}

```

3. Lei

- *makeFeatureVector.py*

this script is created for make feature vectors based on the generated feature files.

```

# -*- coding: utf-8 -*-
"""

```

Created on Thu Jul 25 23:52:09 2013

```

4  this script is created for make feature vectors based on the generated feature files
"""

```

```

import os
9  import sys

```

```

def readUIdLidRating(strInputFile):
    listUId = list()
    listLid = list()
14    listRating = list()
    for line in file(strInputFile):

```

```

        line = line.rstrip()
        strArr = line.split('\t')
        listUid.append(strArr[0])
19      listLid.append(strArr[1])
        if len(strArr) == 3:
            listRating.append(strArr[2])
        else:
            listRating.append(4)
24      return (listUid, listLid, listRating)

def readGlobalFeatures(strFeatureFile):
    dictID2Feature = dict()
    indexOffset = 0
29      for line in file(strFeatureFile):
        line = line.rstrip()
        strArr = line.split('\t')
        if len(strArr) == 1:
            indexOffset = int(strArr[0])
34      else:
        dictID2Feature[strArr[0]+"_"+strArr[1]] = list()
        for i in xrange(2,len(strArr)):
            if strArr[i] != "":
                dictID2Feature[strArr[0]+"_"+strArr[1]].append(strArr[i])
39      return (dictID2Feature,indexOffset)

def readFeatures(strFeatureFile):
    dictID2Feature = dict()
    indexOffset = 0
44      for line in file(strFeatureFile):
        line = line.rstrip()
        strArr = line.split('\t')
        if len(strArr) == 1:
            indexOffset = int(strArr[0])
49      else:
        dictID2Feature[strArr[0]] = list()
        for i in xrange(1,len(strArr)):
            if strArr[i] != "":
                dictID2Feature[strArr[0]].append(strArr[i])
54      return (dictID2Feature,indexOffset)

def initFeatureVector(listRating):
    listVector = list()
    for i in xrange(len(listRating)):
59      listVector.append("")
    return listVector

def initFeatureVectorLen(listRating):
    listVector = list()
64      for i in xrange(len(listRating)):
        listVector.append(0)
    return listVector

69      def appendGlobalFeatures(strFeatureFile,listFeatureVectors,listFeatureVectorsLen, listEntity, currentIndex):
        dictID2Feature,indexOffset = readGlobalFeatures(strFeatureFile)
        print strFeatureFile
        for i in xrange(0,len(listEntity)):
            if dictID2Feature.has_key(listEntity[i]):
74      features = dictID2Feature[listEntity[i]]
                for j in xrange(0,len(features)):
                    strArr = features[j].split(':')
                    listFeatureVectors[i] += "\t" + str(currentIndex+int(strArr[0]))+":"+strArr[1]
                    listFeatureVectorsLen[i] += 1
79      currentIndex += indexOffset
        return (listFeatureVectors, listFeatureVectorsLen, currentIndex)

def appendFeatures(strFeatureFile,listFeatureVectors,listFeatureVectorsLen, listEntity, currentIndex):
    dictID2Feature,indexOffset = readFeatures(strFeatureFile)

```

```

84     print strFeatureFile
    for i in xrange(0,len( listEntity )):
        if dictID2Feature.has_key( listEntity [ i ] ):
            features = dictID2Feature[ listEntity [ i ] ]
            for j in xrange(0,len( features )):
89                 strArr = features[ j ]. split ( ':' )
                    listFeatureVectors [ i ] += "\t" + str( currentIndex+int(strArr[0]))+":"+strArr[1]
                    listFeatureVectorsLen[ i ] += 1
            currentIndex += indexOffset
    return ( listFeatureVectors , listFeatureVectorsLen, currentIndex)

94 def combineUidLid( listUid, listLid ):
    listUidLid = list ( )
    for i in xrange( len( listUid ) ):
        listUidLid .append( listUid[ i ]+"_" + listLid [ i ] )
99     return listUidLid

def max( int_1, int_2 ):
    if int_1 > int_2:
        return int_1
104    else :
        return int_2

if '__main__' == __name__:
    ## input
109    strTrainUidLidRating = sys.argv[1]
    strTestUidLidRating = sys.argv[2]
    strConfigFileTemplate = r"D:\Experiment\configTemplate.conf"
    strItemFeatureFileDir = r"D:\Experiment\Features\i"
    strUserFeatureFileDir = r"D:\Experiment\Features\u"
114    strGlobalFeatureFileDir = r"D:\Experiment\Features\g"

    ## output
    strTrainFeatureFile = sys.argv[3]
    strTestFeatureFile = sys.argv[4]
119    strConfigFile = sys.argv[5]

    userFeatureCnt = 0
    itemFeatureCnt = 0
    globalFeatureCnt = 0
124    listTrainUid , listTrainLid , listTrainRating = readUidLidRating( strTrainUidLidRating )
    listTestUid , listTestLid , listTestRating = readUidLidRating( strTestUidLidRating )
    listTrainUidLid = combineUidLid( listTrainUid, listTrainLid )
    listTestUidLid = combineUidLid( listTestUid, listTestLid )

129    listTrainFeatureVectors = initFeatureVector( listTrainRating )
    listTestFeatureVectors = initFeatureVector( listTestRating )
    listTrainGlobalFeatureVectorLen = initFeatureVectorLen( listTrainRating )
    listTestGlobalFeatureVectorLen = initFeatureVectorLen( listTestRating )
    listTrainUserFeatureVectorLen = initFeatureVectorLen( listTrainRating )
134    listTestUserFeatureVectorLen = initFeatureVectorLen( listTestRating )
    listTrainItemFeatureVectorLen = initFeatureVectorLen( listTrainRating )
    listTestItemFeatureVectorLen = initFeatureVectorLen( listTestRating )

    currentFeatureIndex = 0
139    trainCurrentFeatureIndex = 0
    testCurrentFeatureIndex = 0

    ## get avg rating
144    avgRating = 0
    for i in xrange( len( listTrainRating ) ):
        avgRating += float( listTrainRating [ i ] )
    avgRating = float( avgRating ) / len( listTrainRating )

    ## global feature
149    featureFileList = os. listdir ( strGlobalFeatureFileDir )
    for i in xrange( 0, len( featureFileList ) ):
        featureFilePath = os.path.join( strGlobalFeatureFileDir, featureFileList [ i ] )

```

```

        listTrainFeatureVectors, listTrainGlobalFeatureVectorLen, trainCurrentFeatureIndex = appendGlobalFeatures(
            featureFilePath, listTrainFeatureVectors, listTrainGlobalFeatureVectorLen, listTrainUidId, trainCurrentFeatureIndex)
        listTestFeatureVectors, listTestGlobalFeatureVectorLen, testCurrentFeatureIndex = appendGlobalFeatures(
            featureFilePath, listTestFeatureVectors, listTestGlobalFeatureVectorLen, listTestUidId, testCurrentFeatureIndex)
154    globalFeatureCnt = max(trainCurrentFeatureIndex, testCurrentFeatureIndex)
    ##user feature
    ##user id feature
        currentFeatureIndex = 0
    ##training set
159    dicUid2Idx = dict()
    for i in xrange(0, len(listTrainUid)):
        listTrainUserFeatureVectorLen[i] += 1
        if dicUid2Idx.has_key(listTrainUid[i]):
            listTrainFeatureVectors[i] += "\t" + str(dicUid2Idx[listTrainUid[i]]) + ":1"
164        else:
            dicUid2Idx[listTrainUid[i]] = currentFeatureIndex
            currentFeatureIndex += 1
            listTrainFeatureVectors[i] += "\t" + str(dicUid2Idx[listTrainUid[i]]) + ":1"

    ##testing set
169    for i in xrange(0, len(listTestUid)):
        listTestUserFeatureVectorLen[i] += 1
        if dicUid2Idx.has_key(listTestUid[i]):
            listTestFeatureVectors[i] += "\t" + str(dicUid2Idx[listTestUid[i]]) + ":1"
        else:
174            dicUid2Idx[listTestUid[i]] = currentFeatureIndex
            currentFeatureIndex += 1
            listTestFeatureVectors[i] += "\t" + str(dicUid2Idx[listTestUid[i]]) + ":1"

    print currentFeatureIndex
    ##user features exclude user id
179    trainCurrentFeatureIndex = currentFeatureIndex
    testCurrentFeatureIndex = currentFeatureIndex
    featureFileList = os.listdir(strUserFeatureFileDir)
    for i in xrange(len(featureFileList)):
        featureFilePath = os.path.join(strUserFeatureFileDir, featureFileList[i])
184        listTrainFeatureVectors, listTrainUserFeatureVectorLen, trainCurrentFeatureIndex = appendFeatures(featureFilePath,
            listTrainFeatureVectors, listTrainUserFeatureVectorLen, listTrainUid, trainCurrentFeatureIndex)
        listTestFeatureVectors, listTestUserFeatureVectorLen, testCurrentFeatureIndex = appendFeatures(featureFilePath,
            listTestFeatureVectors, listTestUserFeatureVectorLen, listTestUid, testCurrentFeatureIndex)

    userFeatureCnt = max(trainCurrentFeatureIndex, testCurrentFeatureIndex)
    ##item feature
    ##item id feature
189    currentFeatureIndex = 0
    ##training set
        dicId2Idx = dict()
    for i in xrange(0, len(listTrainId)):
        listTrainItemFeatureVectorLen[i] += 1
        if dicId2Idx.has_key(listTrainId[i]):
            listTrainFeatureVectors[i] += "\t" + str(dicId2Idx[listTrainId[i]]) + ":1"
        else:
194            dicId2Idx[listTrainId[i]] = currentFeatureIndex
            currentFeatureIndex += 1
            listTrainFeatureVectors[i] += "\t" + str(dicId2Idx[listTrainId[i]]) + ":1"

    ##testing set
        for i in xrange(0, len(listTestId)):
            listTestItemFeatureVectorLen[i] += 1
            if dicId2Idx.has_key(listTestId[i]):
204                listTestFeatureVectors[i] += "\t" + str(dicId2Idx[listTestId[i]]) + ":1"
            else:
                dicId2Idx[listTestId[i]] = currentFeatureIndex
                currentFeatureIndex += 1
                listTestFeatureVectors[i] += "\t" + str(dicId2Idx[listTestId[i]]) + ":1"
209

    ##item features exclude item id
        trainCurrentFeatureIndex = currentFeatureIndex
        testCurrentFeatureIndex = currentFeatureIndex
        featureFileList = os.listdir(strItemFeatureFileDir)
        for i in xrange(len(featureFileList)):

```

```

featureFilePath = os.path.join(strItemFeatureFileDir, featureFileList [i])
listTrainFeatureVectors, listTrainItemFeatureVectorLen, trainCurrentFeatureIndex = appendFeatures(featureFilePath,
listTrainFeatureVectors, listTrainItemFeatureVectorLen, listTrainId, trainCurrentFeatureIndex)
listTestFeatureVectors, listTestItemFeatureVectorLen, testCurrentFeatureIndex = appendFeatures(featureFilePath,
listTestFeatureVectors, listTestItemFeatureVectorLen, listTestId, testCurrentFeatureIndex)

219 itemFeatureCnt = max(trainCurrentFeatureIndex, testCurrentFeatureIndex)
##merge rating, feature count, features
for i in xrange(len(listTrainFeatureVectors)):
listTrainFeatureVectors[i] = listTrainRating[i]+"\\t"+str(listTrainGlobalFeatureVectorLen[i])+"\\t"+str(
listTrainUserFeatureVectorLen[i])+"\\t"+str(listTrainItemFeatureVectorLen[i])+listTrainFeatureVectors[i]
224 for i in xrange(len(listTestFeatureVectors)):
listTestFeatureVectors[i] = listTestRating[i]+"\\t"+str(listTestGlobalFeatureVectorLen[i])+"\\t"+str(
listTestUserFeatureVectorLen[i])+"\\t"+str(listTestItemFeatureVectorLen[i])+listTestFeatureVectors[i]

##restore
229 f_train = open(strTrainFeatureFile,"w")
for i in xrange(0,len(listTrainFeatureVectors)):
f_train . write(listTrainFeatureVectors[i]+"\\n")
f_train . close()
f_test = open(strTestFeatureFile,"w")
234 for i in xrange(0,len(listTestFeatureVectors)):
f_test . write(listTestFeatureVectors[i]+"\\n")
f_test . close()

239 ##generate config file
print "globalFeatureCnt=%d,userFeatureCnt=%d,itemFeatureCnt=%d" %(globalFeatureCnt,userFeatureCnt,
itemFeatureCnt)
intLineIdx = 1
f = open(strConfigFile,"w")
for line in file (strConfigFileTemplate):
244 if intLineIdx == 4:
f.write("base_score=%"+str(avgRating)+"\\n")
elif intLineIdx == 16:
f.write("num_item=%"+str(itemFeatureCnt)+"\\n")
elif intLineIdx == 17:
249 f.write("num_user=%"+str(userFeatureCnt)+"\\n")
elif intLineIdx == 19:
f.write("num_global=%"+str(globalFeatureCnt)+"\\n")
else:
f.write(line)
254 intLineIdx += 1
f.close()

```

- *svdPP.bat*

this script is used to run the svd++ model.

```

countUI.py
svdpp_randorder training_set_re.txt train_set_re.svdpporder
line_reorder training_set_re.txt train_set_re.svdpporder train_set_re.shuffle
python mkbasicfeature.py train_set_re.shuffle training_set.basicfeature
5 python mkimplicitfeedbackfeature.py ALLRecord.txt train_set_re.shuffle train_set.feedback
make_ugroup_buffer training_set.basicfeature training_set.buffer -fd train_set.feedback

python group_predict.py predict_re.txt predict_re.g.txt
10 python mkbasicfeature.py predict_re.g.txt testing_set.basicfeature
python mkimplicitfeedbackfeature.py ALLRecord.txt predict_re.g.txt test_set.feedback
make_ugroup_buffer testing_set.basicfeature testing_set.buffer -fd test_set.feedback

svdfeature implicitFeedback.conf num_round=87
15 svdfeature_infer implicitFeedback.conf pred=87
reidx.py pred.txt pred_final.txt
pause

```

- *makeCatBiasFeature.py*

script used to generate global features from business categories.


```

import json

3 def getEntity2IdMap(strMapFile):
    dicEntity2ID = dict()
    for line in file (strMapFile):
        line = line.rstrip()
        strArr = line.split('\t')
8         dicEntity2ID[strArr[0]] = strArr[1]
    return dicEntity2ID

def getItem2Feature(strTrainBusi,strTestBusi,item2ID):
13     catID = 0
    dicCat2ID = dict()
    dicBusi2Feature = dict()
    for line in file (strTrainBusi):
        line = line.rstrip()
18         js = json.loads(line)
        busi = js["business_id"]
        cats = js["categories"]
        feature = ""
        for cat in cats:
23             if not dicCat2ID.has_key(cat):
                dicCat2ID[cat] = catID
                catID += 1
                feature += "\t"+str(dicCat2ID[cat])+":1"
        dicBusi2Feature[item2ID[busi]] = feature
28     return (dicBusi2Feature,catID)

strId2IDmap = r"D:\Experiment\rawData\itemmap.final"
33 strUser2IDmap = r"D:\Experiment\rawData\usermap.final"
strTrainBusi = r"D:\Experiment\rawData\yelp_training_set\training_set_business.json"
strTestBusi = r"D:\Experiment\rawData\yelp_test_set\test_set_business.json"
strTrainReview = r"D:\Experiment\rawData\yelp_training_set\training_set_review.json"
strTestReview = r"D:\Experiment\rawData\yelp_test_set\test_set_review.json"
38 strBiasCatFeature = r"biasCatFeature.txt"

user2ID = getEntity2IdMap(strUser2IDmap)
item2ID = getEntity2IdMap(strId2IDmap)
dicBusi2Feature,featureCnt = getItem2Feature(strTrainBusi,strTestBusi,item2ID)

f = open(strBiasCatFeature,"w")
48 f.write(str(featureCnt)+"\n")
for line in file (strTrainReview):
    line = line.rstrip()
    js = json.loads(line)
    user = js["user_id"]
53     busi = js["business_id"]
    if dicBusi2Feature.has_key(item2ID[busi]):
        f.write(user2ID[user]+" \t"+item2ID[busi]+dicBusi2Feature[item2ID[busi]]+"\n")
for line in file (strTestReview):
    line = line.rstrip()
58     js = json.loads(line)
    user = js["user_id"]
    busi = js["business_id"]
    if dicBusi2Feature.has_key(item2ID[busi]):
        f.write(user2ID[user]+" \t"+item2ID[busi]+dicBusi2Feature[item2ID[busi]]+"\n")
63 f.close()

```

- *makeItemNameFeature.py*
extract the headword of item names as item feature.

```
import json
```

```

def getHeadWord(str):
5     strArr = str.split(' ')
    listWord = list()
    listWord.append(strArr[-1])
    return listWord

10 def getEntity2IdMap(strMapFile):
    dicEntity2ID = dict()
    for line in file(strMapFile):
        line = line.rstrip()
        strArr = line.split('\t')
15     dicEntity2ID[strArr[0]] = strArr[1]
    return dicEntity2ID

dicItem2HeadWord = dict()

20 strTrainBusinessPath = r"D:\Experiment\rawData\yelp_training_set\training_set_business.json"
strTestBusinessPath = r"D:\Experiment\rawData\yelp_test_set\test_set_business.json"
strId2IDmap = r"D:\Experiment\rawData\itemmap.final"
strCatFeature = r"iiWordIdfFeature.txt"

25 dicItem2ID = getEntity2IdMap(strId2IDmap)
dicWord2ID = dict()
wordID = 0

dicItem2WordList = dict()
30 dicWord2ItemList = dict()
intItemCnt = 0

for line in file(strTrainBusinessPath):
    intItemCnt += 1
35     line = line.rstrip()
    js = json.loads(line)
    bid = js["business_id"]
    busiName = js["name"]
    words = getHeadWord(busiName)
40     dicItem2WordList[bid] = list()
    for word in words:
        dicItem2WordList[bid].append(word)
        if not dicWord2ItemList.has_key(word):
            dicWord2ItemList[word] = list()
45     dicWord2ItemList[word].append(bid)

for line in file(strTestBusinessPath):
    intItemCnt += 1
    line = line.rstrip()
50     js = json.loads(line)
    bid = js["business_id"]
    busiName = js["name"]
    words = getHeadWord(busiName)
    dicItem2WordList[bid] = list()
55     for word in words:
        dicItem2WordList[bid].append(word)
        if not dicWord2ItemList.has_key(word):
            dicWord2ItemList[word] = list()
        dicWord2ItemList[word].append(bid)

60 f = open(strCatFeature,"w")
f.write(str(intItemCnt)+"\n")
for line in file(strTrainBusinessPath):
    line = line.rstrip()
65     js = json.loads(line)
    bid = js["business_id"]
    hsNBid = set()
    for word in dicItem2WordList[bid]:
        for Nbid in dicWord2ItemList[word]:

```

```

70         if Nbid != bid:
            hsNBid.add(Nbid)
        if len(hsNBid) == 0:
            continue
        f.write(dicItem2ID[bid])
75     for NBid in hsNBid:
        f.write("\t"+dicItem2ID[NBid]+":1")
        f.write("\n")
    for line in file (strTestBusinessPath):
        line = line.rstrip()
80     js = json.loads(line)
        bid = js["business_id"]
        hsNBid = set()
        for word in dicItem2WordList[bid]:
            for Nbid in dicWord2ItemList[word]:
85                 if Nbid != bid:
                    hsNBid.add(Nbid)
        if len(hsNBid) == 0:
            continue
        f.write(dicItem2ID[bid])
90     for NBid in hsNBid:
        f.write("\t"+dicItem2ID[NBid]+":1")
        f.write("\n")
    f.close()

```

4. Xudong

- *Dataset2.cs*

for loading data.

```

using System;
using System.Collections.Generic;
using System.Linq;
4  using System.Text;
    using System.IO;
    using Newtonsoft.Json;

    using yelpRS.dataset.details;
9
    namespace yelpRS.dataset
    {
        public class DataSet2
        {
14             double globelave = 3.7667;

            public Dictionary<string, int> uidindex = new Dictionary<string, int>();
            public Dictionary<string, int> bidindex = new Dictionary<string, int>();

19             public Dictionary<string, userinfo> userlist = new Dictionary<string, userinfo>();
            public Dictionary<string, businessinfo> businesslist = new Dictionary<string, businessinfo>();
            public Dictionary<string, datainfo> traindatalist = new Dictionary<string, datainfo>();
            public List<datainfo> testdatalist = new List<datainfo>();
            public List<datainfo> testdatalist1 = new List<datainfo>();
24             public List<datainfo> testdatalist2 = new List<datainfo>();
            public List<datainfo> testdatalist3 = new List<datainfo>();
            public List<datainfo> testdatalist4 = new List<datainfo>();

29             public Dictionary<string, int> userwordcount = new Dictionary<string, int>();
            public Dictionary<string, int> businesswordcount = new Dictionary<string, int>();
            public Dictionary<string, int> tagidindex = new Dictionary<string, int>();
            public Dictionary<string, int> positionidindex = new Dictionary<string, int>();
            public Dictionary<string, List<int>> userreviewlist = new Dictionary<string, List<int>>();
34             public Dictionary<string, List<int>> itemreviewlist = new Dictionary<string, List<int>>();
            public Dictionary<string, double> userave = new Dictionary<string, double>();
            public Dictionary<string, double> itemave = new Dictionary<string, double>();
            public Dictionary<string, double> userva = new Dictionary<string, double>();
            public Dictionary<string, double> itemva = new Dictionary<string, double>();

```

```

39     public Dictionary<string, int> usercount = new Dictionary<string, int>();
    public Dictionary<string, int> itemcount = new Dictionary<string, int>();
    public Dictionary<string, int> businessnameindex = new Dictionary<string, int>();
    public Dictionary<string, int> businessnameindex2 = new Dictionary<string, int>();
    public Dictionary<string, int> streetindex = new Dictionary<string, int>();
44     public Dictionary<string, List<double>> usertopic = new Dictionary<string, List<double>>();
    public Dictionary<string, List<double>> businesstopic = new Dictionary<string, List<double>>();
    public Dictionary<string, List<double>> userreviewtopic = new Dictionary<string, List<double>>();
    public Dictionary<string, List<double>> businessreviewtopic = new Dictionary<string, List<double>>();

49     //checkin
    public Dictionary<string, Dictionary<int, int>> traincheckinlist = new Dictionary<string, Dictionary<int, int>>();
    public Dictionary<string, Dictionary<int, int>> testcheckinlist = new Dictionary<string, Dictionary<int, int>>();
    public Dictionary<string, Dictionary<int, int>> checkinhourlist = new Dictionary<string, Dictionary<int, int>>();
    public Dictionary<string, Dictionary<int, int>> checkindaylist = new Dictionary<string, Dictionary<int, int>>();
54     public Dictionary<string, int> traincheckincount = new Dictionary<string, int>();
    public Dictionary<string, int> testcheckincount = new Dictionary<string, int>();

    Dictionary<int, string> map = new Dictionary<int, string>();
    Dictionary<int, string> umap = new Dictionary<int, string>();
    Dictionary<string, string> map2 = new Dictionary<string, string>();
    Dictionary<string, string> usermap2 = new Dictionary<string, string>();

    public Dictionary<string, int> jblantdic = new Dictionary<string, int>();
64     public Dictionary<string, int> jblongdic = new Dictionary<string, int>();
    public Dictionary<string, int> jblockindic = new Dictionary<string, int>();
    public Dictionary<string, int> jbstreetdic = new Dictionary<string, int>();
    public Dictionary<string, int> jbzipcodedic = new Dictionary<string, int>();
    public Dictionary<string, int> jbusertimeindic = new Dictionary<string, int>();
69     public Dictionary<string, int> jbitemtimeindic = new Dictionary<string, int>();
    public Dictionary<string, int> jbitemopencity = new Dictionary<string, int>();
    public Dictionary<string, int> jbcattypedic = new Dictionary<string, int>();
    public Dictionary<string, List<int>> jbica = new Dictionary<string, List<int>>();
    public Dictionary<string, int> jbitemcity = new Dictionary<string, int>();
74     public Dictionary<string, int> jbitemstate = new Dictionary<string, int>();
    public Dictionary<string, int> jbitemreview = new Dictionary<string, int>();
    public Dictionary<string, int> jbusergender = new Dictionary<string, int>();
    public Dictionary<string, int> jbuserreview = new Dictionary<string, int>();
    public Dictionary<string, int> jbusername = new Dictionary<string, int>();
79     public Dictionary<string, int> jbqch = new Dictionary<string, int>();
    public Dictionary<string, int> jbqbn = new Dictionary<string, int>();

    public Dictionary<string, List<double>> useremotionwords = new Dictionary<string, List<double>>();
84     public Dictionary<string, List<double>> itememotionwords = new Dictionary<string, List<double>>();

    public Dictionary<string, int> streetnameindex = new Dictionary<string, int>();

89

    void loadMap(string path)
    {
        StreamReader sr = new StreamReader(path + "itemmap.final");
        string line = "";
94         while ((line = sr.ReadLine()) != null)
        {
            string[] s = line.Split(' ');
            map.Add(int.Parse(s[1]), s[0]);
99         }
        sr.Close();
        StreamReader sr2 = new StreamReader(path + "usermap.final");
        while ((line = sr2.ReadLine()) != null)
        {
104         string[] s = line.Split(' ');
            umap.Add(int.Parse(s[1]), s[0]);
        }
    }

```

```

    sr2.Close();
    StreamReader srw = new StreamReader(path + "itemmap.final");
109     while ((line = srw.ReadLine()) != null)
    {
        string[] s = line.Split(' ');
        map2.Add(s[0], s[1]);
    }
114     srw.Close();
    StreamReader srw2 = new StreamReader(path + "usermap.final");
    while ((line = srw2.ReadLine()) != null)
    {
        string[] s = line.Split(' ');
119         usermap2.Add(s[0], s[1]);
    }
    srw2.Close();
}

124 bool loadTagId(string path)
{
    string file = path + "catid";
    StreamReader sr = new StreamReader(file);
    string line = "";
129     while ((line = sr.ReadLine()) != null)
    {
        string[] s = line.Split(' ');
        tagidindex.Add(s[0].Trim(), int.Parse(s[1]));
    }
134     sr.Close();
    return true;
}

bool loadAvaData(string path)
139 {
    StreamReader sr1 = new StreamReader(path + "useraveandva");
    string line = "";
    while ((line = sr1.ReadLine()) != null)
    {
        string[] splits = line.Split(' ');
144         userave.Add(splits[0], double.Parse(splits[1]));
        userava.Add(splits[0], double.Parse(splits[2]));
    }
    sr1.Close();

149     StreamReader sr2 = new StreamReader(path + "itemaveandva");
    while ((line = sr2.ReadLine()) != null)
    {
        string[] splits = line.Split(' ');
154         itemave.Add(splits[0], double.Parse(splits[1]));
        itemva.Add(splits[0], double.Parse(splits[2]));
    }
    sr2.Close();
    return true;
159 }

bool loadCountData(string path)
{
    StreamReader sr1 = new StreamReader(path + "userreviewcount");
164     string line = "";
    while ((line = sr1.ReadLine()) != null)
    {
        string[] splits = line.Split(' ');
        usercount.Add(splits[0], int.Parse(splits[1]));
169     }
    sr1.Close();

    StreamReader sr2 = new StreamReader(path + "itemreviewcount");
    while ((line = sr2.ReadLine()) != null)
174 {

```

```

        string[] splits = line.Split(' ');
        itemcount.Add(splits[0], int.Parse(splits[1]));
    }
    sr2.Close();
    return true;
}

bool loadBusinessNameData(string path)
{
    184    StreamReader sr = new StreamReader(path + "businessNameData");
    string line = "";
    while ((line = sr.ReadLine()) != null)
    {
        189        string[] splits = line.Split(new string[] { "||| " }, StringSplitOptions.None);
        businessnameindex.Add(splits[0], int.Parse(splits[1]));
    }
    sr.Close();

    StreamReader sr2 = new StreamReader(path + "businessNameData2");
    194    while ((line = sr2.ReadLine()) != null)
    {
        string[] splits = line.Split(new string[] { "||| " }, StringSplitOptions.None);
        businessnameindex2.Add(splits[0], int.Parse(splits[1]));
    }
    199    sr2.Close();
    return true;
}

bool loadStreetNameData(string path)
{
    204    StreamReader sr = new StreamReader(path + "streetname");
    string line = "";
    while ((line = sr.ReadLine()) != null)
    {
        209        string[] splits = line.Split(new string[] { "||| " }, StringSplitOptions.None);
        try
        {
            streetindex.Add(splits[0], int.Parse(splits[1]));
        }
        214        catch
        {
            //Console.WriteLine(splits[0]);
        }
    }
    219    sr.Close();
    return true;
}

void loadWordCountData(string path)
{
    224    string line = "";
    StreamReader sr1 = new StreamReader(path + "usertext_count");
    while ((line = sr1.ReadLine()) != null)
    {
        229        string[] splits = line.Split(new string[] { "||| " }, StringSplitOptions.None);
        userwordcount.Add(splits[0], int.Parse(splits[1]));
    }
    sr1.Close();
    StreamReader sr2 = new StreamReader(path + "businesstext_count");
    234    while ((line = sr2.ReadLine()) != null)
    {
        string[] splits = line.Split(new string[] { "||| " }, StringSplitOptions.None);
        businesswordcount.Add(splits[0], int.Parse(splits[1]));
    }
    239    sr2.Close();
}

void loadTopicData(string path)

```

```

{
244     string line = "";
    StreamReader sr1 = new StreamReader(path + "user_topic");
    while ((line = sr1.ReadLine()) != null)
    {
        line = line.Trim();
249        string[] splits = line.Split(new string[] { "|||" }, StringSplitOptions.None);
        List<double> tmp = new List<double>();
        string[] s = splits[1].Split(' ');
        foreach (var v in s)
            tmp.Add(double.Parse(v));
254        usertopic.Add(splits[0], tmp);
    }
    sr1.Close();

    StreamReader sr2 = new StreamReader(path + "business_topic");
259    while ((line = sr2.ReadLine()) != null)
    {
        line = line.Trim();
        string[] splits = line.Split(new string[] { "|||" }, StringSplitOptions.None);
        List<double> tmp = new List<double>();
264        string[] s = splits[1].Split(' ');
        foreach (var v in s)
            tmp.Add(double.Parse(v));
        businesstopic.Add(splits[0], tmp);
    }
269    sr2.Close();
}

void loadReviewTopicData(string path)
{
274    string line = "";
    StreamReader sr1 = new StreamReader(path + "user_reviewtopic");
    while ((line = sr1.ReadLine()) != null)
    {
        line = line.Trim();
279        string[] splits = line.Split(new string[] { "|||" }, StringSplitOptions.None);
        List<double> tmp = new List<double>();
        string[] s = splits[1].Split(' ');
        foreach (var v in s)
            tmp.Add(double.Parse(v));
284        userreviewtopic.Add(splits[0], tmp);
    }
    sr1.Close();

    StreamReader sr2 = new StreamReader(path + "business_reviewtopic");
289    while ((line = sr2.ReadLine()) != null)
    {
        line = line.Trim();
        string[] splits = line.Split(new string[] { "|||" }, StringSplitOptions.None);
        List<double> tmp = new List<double>();
294        string[] s = splits[1].Split(' ');
        foreach (var v in s)
            tmp.Add(double.Parse(v));
        businessreviewtopic.Add(splits[0], tmp);
    }
299    sr2.Close();
}

void loadEmotionWords(string path)
{
304    StreamReader sr1 = new StreamReader(path + "user_emotionwords");
    string line = "";
    while ((line = sr1.ReadLine()) != null)
    {
        line = line.Trim();
309        string[] splits = line.Split(new string[] { "|||" }, StringSplitOptions.None);
        string[] s = splits[1].Trim().Split(' ');

```

```

        List<double> t = new List<double>();
        foreach (var v in s)
            t.Add(double.Parse(v));
314     useremotionwords.Add(splits[0], t);
    }
    sr1.Close();

    StreamReader sr2 = new StreamReader(path + "user_emotionwords");
319     while ((line = sr2.ReadLine()) != null)
    {
        line = line.Trim();
        string[] splits = line.Split(new string[] { "|||" }, StringSplitOptions.None);
        string[] s = splits[1].Split(' ');
324     List<double> t = new List<double>();
        foreach (var v in s)
            t.Add(double.Parse(v));
        itememotionwords.Add(splits[0], t);
    }
329     sr2.Close();
}

bool loadUidIndex(string path)
334 {
    Console.WriteLine("-----loadingTrainUidIndexData-----");
    StreamReader sr = new StreamReader(path + "uid");
    string line = "";
339     while ((line = sr.ReadLine()) != null)
    {
        string[] splits = line.Split(' ');
        uidindex.Add(splits[0], int.Parse(splits[1]));
    }
344     sr.Close();
    Console.WriteLine("-----loadingTrainUidIndexDataOver-----");
    return true;
}

bool loadBidIndex(string path)
349 {
    Console.WriteLine("-----loadingTrainBidIndexData-----");
    StreamReader sr = new StreamReader(path + "bid");
    string line = "";
354     while ((line = sr.ReadLine()) != null)
    {
        string[] splits = line.Split(' ');
        bidindex.Add(splits[0], int.Parse(splits[1]));
    }
359     sr.Close();
    Console.WriteLine("-----loadingTrainBidIndexDataOver-----");
    return true;
}

bool loadTrainUserData(string path)
364 {
    Console.WriteLine("-----loadingTrainUserData-----");
    StreamReader sr = new StreamReader(path + "yelp_training_set_user.json");
    string line = "";
369     while ((line = sr.ReadLine()) != null)
    {
        JsonReader jr = new JsonTextReader(new StringReader(line));
        userinfo uif = new userinfo();
        string uid = "";
        int i = 0;
        while (jr.Read())

```



```

379         {
            i++;
            if (i == 5)
                uif.votefun = int.Parse(jr.Value.ToString());
            if (i == 7)
                uif.voteuserful = int.Parse(jr.Value.ToString());
384            if (i == 9)
                uif.votecool = int.Parse(jr.Value.ToString());
            if (i == 12)
                uid = jr.Value.ToString();
389            if (i == 14)
                uif.name = jr.Value.ToString();
            if (i == 16)
                uif.avescore = double.Parse(jr.Value.ToString());
            if (i == 18)
394            {
                uif.reviewcount = int.Parse(jr.Value.ToString());
            }
        }
        userlist [uid] = uif;
399    }
    sr.Close();
    Console.WriteLine("-----loadingTrainUserDataOver-----");
    return true;
}

404 bool loadTestUserData(string path)
{
    Console.WriteLine("-----loadingTestUserData-----");
    StreamReader sr = new StreamReader(path + "final_test_set_user.json");
409    string line = "";
    while ((line = sr.ReadLine()) != null)
    {
        JsonReader jr = new JsonTextReader(new StringReader(line));
        string lastline = "";
414        string uid = "";
        userinfo uif = new userinfo();
        while (jr.Read())
        {
            if (lastline == "review_count")
                uif.reviewcount = int.Parse(jr.Value.ToString());
            if (lastline == "name")
                uif.name = jr.Value.ToString();
            if (lastline == "average_stars")
                uif.avescore = double.Parse(jr.Value.ToString());
424            if (lastline == "user_id")
                uid = jr.Value.ToString();
            if (jr.ValueType != null)
                lastline = jr.Value.ToString();
        }
429        userlist [uid] = uif;
    }
    sr.Close();
    Console.WriteLine("-----loadingTestUserDataOver-----");
    return true;
434 }

void loadUserData(string path)
{
    foreach (var v in uidindex.Keys)
439    {
        userinfo uif = new userinfo();
        userlist .Add(v, uif);
    }
    loadTrainUserData(path + "yelp_training_set");
444    loadTestUserData(path + "final_test_set");
}

```

```

449 bool loadTrainBusinessData(string path)
{
    Console.WriteLine("-----loadingTrainBusinessData-----");
    StreamReader sr = new StreamReader(path + "yelp_training_set_business.json");
    string line = "";
454 while ((line = sr.ReadLine()) != null)
    {
        JsonReader jr = new JsonTextReader(new StringReader(line));
        businessinfo bif = new businessinfo();
        string bid = "";
459 int i = 0;
        string lastline = "";
        while (jr.Read())
        {
            i++;
464 if (i == 3)
                bid = jr.Value.ToString();
            if (i == 5)
                bif.address = jr.Value.ToString();
            if (i == 7)
                bif.isopen = bool.Parse(jr.Value.ToString());
469 if (lastline == "categories")
            {
                while (jr.ValueType != null && jr.Value.ToString() != "city")
                {
474 bif.categories.Add(jr.Value.ToString());
                    jr.Read();
                }
            }
            if (lastline == "city")
                bif.city = jr.Value.ToString();
            if (lastline == "review_count")
            {
484 bif.review_count = int.Parse(jr.Value.ToString());
            }
            if (lastline == "name")
                bif.name = jr.Value.ToString();
            if (lastline == "longitude")
                bif.longitude = double.Parse(jr.Value.ToString());
489 if (lastline == "state")
                bif.state = jr.Value.ToString();
            if (lastline == "stars")
            {
                bif.stras = double.Parse(jr.Value.ToString());
494
            }
            if (lastline == "latitude")
                bif.latitude = double.Parse(jr.Value.ToString());
            if (jr.ValueType != null)
                lastline = jr.Value.ToString();
499        }
        businesslist [bid] = bif;
    };
    sr.Close();
504
    Console.WriteLine("-----loadingTrainBusinessDataOver-----");
    return true;
}

509 bool loadTestBusinessData(string path)
{
    Console.WriteLine("-----loadingTestBusinessData-----");
    StreamReader sr = new StreamReader(path + "final_test_set_business.json");
    string line = "";
514 while ((line = sr.ReadLine()) != null)

```

```

{
    JsonReader jr = new JsonTextReader(new StringReader(line));
    businessinfo bif = new businessinfo();
    string bid = "";
519     int i = 0;
    string lastline = "";
    while (jr.Read())
    {
        i++;
524         if (lastline == "business_id")
            bid = jr.Value.ToString();
        if (lastline == "full_address")
            bif.address = jr.Value.ToString();
        if (lastline == "open")
529             bif.isopen = bool.Parse(jr.Value.ToString());
        if (lastline == "categories")
        {
            while (jr.ValueType != null && jr.Value.ToString() != "city")
            {
534                 bif.categories.Add(jr.Value.ToString());
                jr.Read();
            }
        }
        if (lastline == "city")
539             bif.city = jr.Value.ToString();
        if (lastline == "review_count")
            bif.review_count = int.Parse(jr.Value.ToString());
        if (lastline == "name")
            bif.name = jr.Value.ToString();
544         ////neighbor
        if (lastline == "longitude")
            bif.longitude = double.Parse(jr.Value.ToString());
        if (lastline == "state")
            bif.state = jr.Value.ToString();
549         if (lastline == "stars")
            bif.stras = double.Parse(jr.Value.ToString());
        if (lastline == "latitude")
            bif.latitude = double.Parse(jr.Value.ToString());
        if (jr.ValueType != null)
554             lastline = jr.Value.ToString();
    }
    businesslist [bid] = bif;
}
sr.Close();
559 path = "F:\\yelp_RS\\data\\generated";
return true;
}

void loadBusinessData(string path)
564 {
    foreach (var v in bidindex.Keys)
    {
        businessinfo bif = new businessinfo();
        businesslist.Add(v, bif);
569    }
    loadTrainBusinessData(path + "yelp_training_set");
    loadTestBusinessData(path + "final_test_set");
}

bool loadTrainReviewData(string path)
{
    Console.WriteLine("-----loadingTrainReviewData-----");
579    StreamReader sr = new StreamReader(path + "yelp_training_set_review.json");
    string line = "";
    while ((line = sr.ReadLine()) != null)
    {

```

```

584     JsonReader jr = new JsonTextReader(new StringReader(line));
    string lastline = "";
    string rid = "";
    datainfo dif = new datainfo();
    while (jr.Read())
    {
589         if (lastline == "funnny")
            dif.votefun = int.Parse(jr.Value.ToString());
        if (lastline == "useful")
            dif.voteuserful = int.Parse(jr.Value.ToString());
        if (lastline == "cool")
594            dif.votecool = int.Parse(jr.Value.ToString());
        if (lastline == "user_id")
            dif.uid = jr.Value.ToString();
        if (lastline == "review_id")
            rid = jr.Value.ToString();
599        if (lastline == "stars")
            dif.stars = int.Parse(jr.Value.ToString());
        if (lastline == "date")
            dif.date = jr.Value.ToString();
        if (lastline == "text")
604            dif.text = jr.Value.ToString();
        if (lastline == "business_id")
            dif.bid = jr.Value.ToString();

        if (jr.ValueType != null)
609            lastline = jr.Value.ToString();
    }

    try
    {
614        List<int> tmplist = new List<int>();
        tmplist.Add(bidindex[dif.bid]);
        userreviewlist.Add(dif.uid, tmplist);
    }
    catch
619    {
        userreviewlist[dif.uid].Add(bidindex[dif.bid]);
    }

    try
624    {
        List<int> tmplist = new List<int>();
        tmplist.Add(uidindex[dif.uid]);
        itemreviewlist.Add(dif.bid, tmplist);
    }
    catch
629    {
        itemreviewlist[dif.bid].Add(uidindex[dif.uid]);
    }
    ///
634    traindatalist.Add(rid, dif);
}
sr.Close();

Console.WriteLine("-----loadingTrainReviewDataOver-----");
639 return true;
}

bool loadTrainCheckinData(string path)
{
644    Console.WriteLine("-----loadingTrainCheckinData-----");
    StreamReader sr = new StreamReader(path + "yelp_training_set_checkin.json");
    string line = "";
    while ((line = sr.ReadLine()) != null)
    {
649        JsonReader jr = new JsonTextReader(new StringReader(line));
        string lastline = "";

```

```

        string vid = "";
        Dictionary<int, int> tmp = new Dictionary<int, int>();
        for (int i = 0; i < 7 * 24; i++)
        {
            tmp.Add(i, 0);
            int count = 0;
            while (jr.Read())
            {
                if (lastline == "business_id")
                {
                    vid = jr.Value.ToString();
                    if (jr.TokenType.ToString() == "Integer")
                    {
                        string[] s = lastline.Split('-');
                        int time = 24 * int.Parse(s[1]) + int.Parse(s[0]);
                        tmp[time] = int.Parse(jr.Value.ToString());
                        count += int.Parse(jr.Value.ToString());
                    }
                    if (jr.ValueType != null)
                        lastline = jr.Value.ToString();
                }
                traincheckinlist.Add(vid, tmp);
                traincheckincount.Add(vid, count);
            }
        }
        sr.Close();
        Console.WriteLine("-----loadingTrainCheckinDataOver-----");
        return true;
    }

    bool loadTestReviewData(string path)
    {
        Console.WriteLine("-----loadingTestReviewData-----");
        StreamReader sr = new StreamReader(path + "test");
        string line = "";
        while ((line = sr.ReadLine()) != null)
        {
            string[] splits = line.Split(',');
            datainfo dif = new datainfo();
            dif.uid = splits[0];
            dif.bid = splits[1];
            dif.stars = int.Parse(splits[2]);
            testdatalist.Add(dif);
        }
        sr.Close();

        StreamReader sr1 = new StreamReader(path + "knownuser_knownitem");
        while ((line = sr1.ReadLine()) != null)
        {
            string[] splits = line.Split(',');
            datainfo dif = new datainfo();
            dif.uid = splits[0];
            dif.bid = splits[1];
            dif.stars = int.Parse(splits[2]);
            testdatalist1.Add(dif);
        }
        sr1.Close();

        StreamReader sr2 = new StreamReader(path + "knownuser_newitem");
        while ((line = sr2.ReadLine()) != null)
        {
            string[] splits = line.Split(',');
            datainfo dif = new datainfo();
            dif.uid = splits[0];
            dif.bid = splits[1];
            dif.stars = int.Parse(splits[2]);
            testdatalist2.Add(dif);
        }
        sr2.Close();
    }

```

```

719     StreamReader sr3 = new StreamReader(path + "newuser_knownitem");
    while ((line = sr3.ReadLine()) != null)
    {
        string[] splits = line.Split(',');
        datainfo dif = new datainfo();
724         dif.uid = splits[0];
        dif.bid = splits[1];
        dif.stars = int.Parse(splits[2]);
        testdatalist3.Add(dif);
    }
729     sr3.Close();

    StreamReader sr4 = new StreamReader(path + "newuser_newitem");
    while ((line = sr4.ReadLine()) != null)
    {
        string[] splits = line.Split(',');
        datainfo dif = new datainfo();
734         dif.uid = splits[0];
        dif.bid = splits[1];
        dif.stars = int.Parse(splits[2]);
739         testdatalist4.Add(dif);
    }
    sr4.Close();

    Console.WriteLine("-----loadingTestReviewDataOver-----");
744     return true;
}

bool loadTestCheckinData(string path)
{
749     Console.WriteLine("-----loadingTrainBusinessData-----");
    StreamReader sr = new StreamReader(path + "final_test_set_checkin.json");
    string line = "";
    while ((line = sr.ReadLine()) != null)
    {
754         JsonReader jr = new JsonTextReader(new StringReader(line));
        string lastline = "";
        string vid = "";
        int count = 0;
        Dictionary<int, int> tmp = new Dictionary<int, int>();
759         for (int i = 0; i < 7 * 24; i++)
            tmp.Add(i, 0);
        while (jr.Read())
        {
764             if (lastline == "business_id")
                vid = jr.Value.ToString();
            if (jr.TokenType.ToString() == "Integer")
            {
                string[] s = lastline.Split('-');
                int time = 24 * int.Parse(s[1]) + int.Parse(s[0]);
769                 tmp[time] = int.Parse(jr.Value.ToString());
                count += int.Parse(jr.Value.ToString());
            }
            if (jr.ValueType != null)
                lastline = jr.Value.ToString();
774         }
        testcheckinlist.Add(vid, tmp);
        testcheckincount.Add(vid, count);
    }
    sr.Close();
779     Console.WriteLine("-----loadingTrainBusinessDataOver-----");
    return true;
}

void calCheckin()
784 {
    foreach (var v in traincheckinlist)
    {

```

```

789     string key = v.Key;
    Dictionary<int, int> hourtmp = new Dictionary<int, int>();
    for (int i = 0; i < 24; i++)
    {
        hourtmp.Add(i, 0);
    }
    Dictionary<int, int> daytmp = new Dictionary<int, int>();
794     for (int i = 0; i < 7; i++)
    {
        daytmp.Add(i, 0);
    }
    foreach (var va in v.Value)
799     {
        hourtmp[va.Key % 24] += va.Value;
        daytmp[va.Key / 24] += va.Value;
    }
    checkindaylist.Add(v.Key, daytmp);
804     checkinhourlist.Add(v.Key, hourtmp);
}
foreach (var v in testcheckinlist)
{
    string key = v.Key;
809     Dictionary<int, int> hourtmp = new Dictionary<int, int>();
    for (int i = 0; i < 24; i++)
    {
        hourtmp.Add(i, 0);
    }
814     Dictionary<int, int> daytmp = new Dictionary<int, int>();
    for (int i = 0; i < 7; i++)
    {
        daytmp.Add(i, 0);
    }
819     foreach (var va in v.Value)
    {
        hourtmp[va.Key % 24] += va.Value;
        daytmp[va.Key / 24] += va.Value;
    }
824     checkindaylist.Add(v.Key, daytmp);
    checkinhourlist.Add(v.Key, hourtmp);
}

}

829

void loadGenerateData(string gepath)
{
834     loadTagId(gepath);
    loadAvaData(gepath);
    loadCountData(gepath);
    loadBusinessNameData(gepath);
    loadStreetNameData(gepath);
839     loadWordCountData(gepath);
    loadTopicData(gepath);
    loadReviewTopicData(gepath);
    loadEmotionWords(gepath);
}

844
Dictionary<string, int> nameid = new Dictionary<string, int>();
Dictionary<string, int> namecount = new Dictionary<string, int>();
bool generatBusinessNameData(string path)
{
849     StreamWriter sw = new StreamWriter(path + "businessNameData");
    int id = 0;
    foreach (var v in businesslist)
    {
        string name = v.Value.name;
854         try

```

```

    {
        string[] names = v.Value.name.Split(' ');
        name = names[0].Trim();
    }
859     catch { }
        try
        {
            nameid.Add(name, id);
            namecount.Add(name, 1);
864             id++;
        }
        catch
        {
            namecount[name]++;
869        }
    }

    foreach (var item in nameid.OrderBy(s => s.Key))
    {
874         sw.WriteLine(item.Key + "|||" + item.Value); //+ "|||" +
    }
    sw.Close();
    return true;
}

879 void loadJiaobenjunData(string path)
{
    string line = "";
    path = "F:\Dropbox\RecSysChallenge\Features";
884    StreamReader sriva = new StreamReader(path + "i//Liang_ItemCategories.txt");
    line = sriva.ReadLine();
    while ((line = sriva.ReadLine()) != null)
    {
        string[] splits = line.Split(' ');
889        List<int> tmp = new List<int>();
        for (int i = 1; i < splits.Length - 1; i++)
        {
            string[] s = splits[i].Split(':');
            tmp.Add(int.Parse(s[0]));
894        }
        tmp.Sort();
        jbica.Add(map[int.Parse(splits[0])], tmp);
    }
    sriva.Close();
899

    StreamReader sric = new StreamReader(path + "i//Liang_ItemCity.txt");
    line = sric.ReadLine();
    while ((line = sric.ReadLine()) != null)
    {
904        string[] splits = line.Split(' ');
        string[] s = splits[1].Split(':');
        jbitemcity.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
    }
    sric.Close();
909

    StreamReader srir = new StreamReader(path + "i//Liang_ItemReview.txt");
    line = srir.ReadLine();
    while ((line = srir.ReadLine()) != null)
    {
914        string[] splits = line.Split(' ');
        string[] s = splits[1].Split(':');
        jbitemreview.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
    }
    srir.Close();
919

    StreamReader sris = new StreamReader(path + "i//Liang_ItemState.txt");
    line = sris.ReadLine();
    while ((line = sris.ReadLine()) != null)

```



```

924     {
        string [] splits = line.Split('\t');
        string [] s = splits [1].Split(':');
        jbitemstate.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
    }
    sris.Close();

929

StreamReader srlat = new StreamReader(path + "i//Liang_ItemLatitude.txt");
line = srlat.ReadLine();
while ((line = srlat.ReadLine()) != null)
{
934     string [] splits = line.Split('\t');
    string [] s = splits [1].Split(':');
    jblantdic.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
}
srlat.Close();

939

StreamReader srlon = new StreamReader(path + "i//Liang_ItemLongitude.txt");
line = srlon.ReadLine();
while ((line = srlon.ReadLine()) != null)
{
944     string [] splits = line.Split('\t');
    string [] s = splits [1].Split(':');
    jblongdic.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
}
srlon.Close();

949

StreamReader srlocation = new StreamReader(path + "i//Liang_ItemLocation.txt");
line = srlocation.ReadLine();
while ((line = srlocation.ReadLine()) != null)
{
954     string [] splits = line.Split('\t');
    string [] s = splits [1].Split(':');
    jblocklocationdic.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
}
srlocation.Close();

959

StreamReader srstreet = new StreamReader(path + "i//Qiang_1264_Street.txt");
line = srstreet.ReadLine();
while ((line = srstreet.ReadLine()) != null)
{
964     string [] splits = line.Split('\t');
    string [] s = splits [1].Split(':');
    jbstreetdic.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
}
srstreet.Close();

969

StreamReader srqbn = new StreamReader(path + "i//Qiang_9998_BusinessName_Normalized.txt");
line = srqbn.ReadLine();
while ((line = srqbn.ReadLine()) != null)
{
974     string [] splits = line.Split('\t');
    string [] s = splits [1].Split(':');
    jbqbn.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
}
srqbn.Close();

979

StreamReader srzipcode = new StreamReader(path + "i//Qiang_197_zipcode.txt");
line = srzipcode.ReadLine();
while ((line = srzipcode.ReadLine()) != null)
{
984     string [] splits = line.Split('\t');
    string [] s = splits [1].Split(':');
    jbzipcodedic.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
}
srzipcode.Close();

989

StreamReader srlug = new StreamReader(path + "u//Liang_UserGender.txt");

```

```

line = srlug.ReadLine();
while ((line = srlug.ReadLine()) != null)
{
    string[] splits = line.Split('\t');
    string[] s = splits[1].Split(':');
    jbusergender.Add(umap[int.Parse(splits[0])], int.Parse(s[0]));
}
srlug.Close();

StreamReader srlunl = new StreamReader(path + "u//Liang_UserNameLen.txt");
line = srlunl.ReadLine();
while ((line = srlunl.ReadLine()) != null)
{
    string[] splits = line.Split('\t');
    string[] s = splits[1].Split(':');
    jbusernamelen.Add(umap[int.Parse(splits[0])], int.Parse(s[0]));
}
srlunl.Close();

StreamReader srlur = new StreamReader(path + "u//Liang_UserReview.txt");
line = srlur.ReadLine();
while ((line = srlur.ReadLine()) != null)
{
    string[] splits = line.Split('\t');
    string[] s = splits[1].Split(':');
    jbuserreview.Add(umap[int.Parse(splits[0])], int.Parse(s[0]));
}
srlur.Close();

StreamReader srtimebin = new StreamReader(path + "ui//Liang_UserItemTimeBin.txt");
line = srtimebin.ReadLine();
while ((line = srtimebin.ReadLine()) != null)
{
    string[] splits = line.Split('\t');
    string[] s = splits[2].Split(':');
    jbusertimebindic.Add(umap[int.Parse(splits[0])] + map[int.Parse(splits[1])], int.Parse(s[0]));
}
srtimebin.Close();

StreamReader srsname = new StreamReader(path + "i//xudong_streetname.txt");
line = srsname.ReadLine();
while ((line = srsname.ReadLine()) != null)
{
    string[] splits = line.Split('\t');
    string[] s = splits[1].Split(':');
    streetnameindex.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
}
srsname.Close();
}

////
bool calAveScore(string path)
{
    StreamWriter sw1 = new StreamWriter(path + "useraveandva");
    StreamWriter sw2 = new StreamWriter(path + "itemaveandva");
    double sum = 0;
    double sum2 = 0;
    foreach (var v in userreviewlist)
    {
        sum = 0;
        sum2 = 0;
        foreach (var va in v.Value)
        {
            sum += (double)va;
            sum2 += (double)va * va;
        }
        double ave = sum / v.Value.Count;
        double var = sum2 / v.Value.Count - ave * ave;
        sw1.WriteLine(v.Key + " " + ave + " " + var);
    }
}

```

```

1059     }
    foreach (var v in itemreviewlist)
    {
        sum = 0;
        sum2 = 0;
1064        foreach (var va in v.Value)
        {
            sum += (double)va;
            sum2 += (double)va * va;
        }
1069        double ave = sum / v.Value.Count;
        double var = sum2 / v.Value.Count - ave * ave;
        sw2.WriteLine(v.Key + " " + ave + " " + var);
    }
    sw1.Close();
1074    sw2.Close();
    return true;
}

void featureGenerator(string path)
1079 {
    StreamWriter swuserglobleave = new StreamWriter(path + "features//globle_user_ave");
    Dictionary<string, int> userglobleave = new Dictionary<string, int>();
    foreach (var v in userlist)
    {
1084        userglobleave.Add(usermap2[v.Key], (int)(v.Value.avescore / 0.2));
    }
    foreach (var v in userglobleave)
        swuserglobleave.WriteLine(v.Key + "\t" + v.Value + ":1");
    swuserglobleave.Close();

1089    StreamWriter swusercount = new StreamWriter(path + "features//globle_user_count");
    Dictionary<string, int> usercount = new Dictionary<string, int>();
    foreach (var v in userlist)
    {
1094        usercount.Add(usermap2[v.Key], (int)(Math.Log(v.Value.reviewcount + 1) / 0.1));
    }
    foreach (var v in usercount)
        swusercount.WriteLine(v.Key + "\t" + v.Value + ":1");
    swusercount.Close();

1099    StreamWriter swbusinessave = new StreamWriter(path + "features//globle_business_ave");
    Dictionary<string, int> businessave = new Dictionary<string, int>();
    foreach (var v in businesslist)
    {
1104        businessave.Add(map2[v.Key], (int)(v.Value.stras / 0.2));
    }
    foreach (var v in businessave)
        swbusinessave.WriteLine(v.Key + "\t" + v.Value + ":1");
    swbusinessave.Close();

1109    StreamWriter swbusinesscount = new StreamWriter(path + "features//globle_business_count");
    Dictionary<string, int> businesscount = new Dictionary<string, int>();
    foreach (var v in businesslist)
    {
1114        businesscount.Add(map2[v.Key], (int)(Math.Log(v.Value.review_count + 1) / 0.1));
    }
    foreach (var v in businesscount)
        swbusinesscount.WriteLine(v.Key + "\t" + v.Value + ":1");
    swbusinesscount.Close();

1119    StreamWriter sw1 = new StreamWriter("F:\yelp_RS\data\features\checkin_hour");
    StreamWriter sw2 = new StreamWriter(@"F:\yelp_RS\data\features\checkin_day");
    sw1.WriteLine(24);
    sw2.WriteLine(7);
1124    foreach (var v in checkinhourlist)
    {
        StringBuilder sb = new StringBuilder();

```

```

1129     sb.Append(map2[v.Key] + "\t");
        for (int i = 0; i < 24; i++)
        {
            if (v.Value[i] > 5)
            {
                double value = Math.Min(1, (double)(v.Value[i] / 200));
                sb.Append(i).Append(":").Append(value).Append("\t");
            }
        }
        sw1.WriteLine(sb.ToString());
    }
    foreach (var v in checkindaylist)
    {
1139        StringBuilder sb = new StringBuilder();
        sb.Append(map2[v.Key] + "\t");
        for (int i = 0; i < 7; i++)
        {
1144            if (v.Value[i] > 5)
            {
                double value = Math.Min(1, (double)(v.Value[i] / 200));
                sb.Append(i).Append(":").Append(value).Append("\t");
            }
        }
1149        sw2.WriteLine(sb.ToString());
    }
    sw1.Close();
    sw2.Close();
1154    StreamWriter sw3 = new StreamWriter(@"F:\yelp_RS\data\features\checkin_count");
    sw3.WriteLine(301);
    foreach (var v in traincheckincount)
    {
1159        StringBuilder sb = new StringBuilder();
        sb.Append(map2[v.Key] + "\t");
        int reviewcount = traincheckincount[v.Key];
        reviewcount = Math.Min(reviewcount, 3000);
        reviewcount /= 10;
        sb.Append(reviewcount).Append(":1.");
1164        sw3.WriteLine(sb.ToString());
    }
    foreach (var v in testcheckincount)
    {
1169        StringBuilder sb = new StringBuilder();
        sb.Append(map2[v.Key] + "\t");
        int reviewcount = testcheckincount[v.Key];
        reviewcount = Math.Min(reviewcount, 3000);
        reviewcount /= 10;
        sb.Append(reviewcount).Append(":1.");
1174        sw3.WriteLine(sb.ToString());
    }
    sw3.Close();

    StreamWriter sw4 = new StreamWriter(@"F:\yelp_RS\data\features\streetname");
    StreamWriter sw5 = new StreamWriter(@"F:\yelp_RS\data\features\businessname");
    StreamWriter sw6 = new StreamWriter(@"F:\yelp_RS\data\features\otecool");
    StreamWriter sw7 = new StreamWriter(@"F:\yelp_RS\data\features\otecofun");
    StreamWriter sw8 = new StreamWriter(@"F:\yelp_RS\data\features\otecuseful");
    foreach (var v in businesslist)
1184    {
        StringBuilder sb = new StringBuilder();
        sb.Append(map2[v.Key] + "\t");
        string fulladdress = v.Value.address;
        int start = fulladdress.LastIndexOf("\n");
1189        if (start == -1)
            start = 0;
        int end = fulladdress.IndexOf(",AZ");
        if (end == -1)
            end = fulladdress.IndexOf(",");
1194        string streetname = fulladdress.Substring(start, end - start).Trim();

```

```

        sb.Append(streetindex[streetname]).Append(":1_");
        sw4.WriteLine(sb.ToString());
    }
    foreach (var v in businesslist )
1199    {
        StringBuilder sb = new StringBuilder();
        sb.Append(map2[v.Key] + "\t");
        try
        {
1204            string name = v.Value.name;
            string [] names = v.Value.name.Split('_');
            name = names[0].Trim();
            sb.Append(businessnameindex[name] + ":1_");
            sw5.WriteLine(sb.ToString());
1209        }
        catch
        {
        }
    }
1214    foreach (var v in userlist )
    {
        StringBuilder sb = new StringBuilder();
        sb.Append(usermap2[v.Key] + "\t");
        int fun = (int)(100 * Math.Log((v.Value.votecool + 1)));
1219        sb.Append(fun).Append(":1_");
        sw6.WriteLine(sb.ToString());
    }
    foreach (var v in userlist )
    {
1224        StringBuilder sb = new StringBuilder();
        sb.Append(usermap2[v.Key] + "\t");
        int fun = (int)(100 * Math.Log((v.Value.votefun + 1)));
        sb.Append(fun).Append(":1_");
        sw7.WriteLine(sb.ToString());
1229    }
    foreach (var v in userlist )
    {
        StringBuilder sb = new StringBuilder();
        sb.Append(usermap2[v.Key] + "\t");
1234        int fun = (int)(100 * Math.Log((v.Value.voteuserful + 1)));
        sb.Append(fun).Append(":1_");
        sw8.WriteLine(sb.ToString());
    }
    sw4.Close();
1239    sw5.Close();
    sw6.Close();
    sw7.Close();
    sw8.Close();
}

1244 void generateTextData()
{
    Dictionary<string, List<string>> usertextdic = new Dictionary<string, List<string>>();
    Dictionary<string, List<string>> businesstextdic = new Dictionary<string, List<string>>();
1249    foreach (var v in traindatalist )
    {
        string uid = v.Value.uid;
        string bid = v.Value.bid;
        string text = v.Value.text;
1254
        string [] splits = text.Split(new char[3] { '_', '\n', '\r' });
        for (int i = 0; i < splits.Length; i++)
        {
            try
            {
1259                List<string> t = new List<string>();
                t.Add(splits[i]);
                usertextdic.Add(uid, t);
            }
            catch
            {
            }
        }
    }
}

```

```

    }
1264     catch
    {
        usertextdic[uid].Add(splits[i]);
    }
    try
1269     {
        List<string> t = new List<string>();
        t.Add(splits[i]);
        businesstextdic.Add(bid, t);
    }
1274     catch
    {
        businesstextdic[bid].Add(splits[i]);
    }
}
1279 StreamWriter sw1 = new StreamWriter(@"F:\yelp_RS\data\exp\usertext");
foreach (var v in usertextdic)
{
    List<string> t = v.Value;
1284     t.Sort();
    StringBuilder sb = new StringBuilder();
    sb.Append(v.Key + "||||");
    foreach (var va in t)
        sb.Append(va).Append(" ");
1289     sw1.WriteLine(sb.ToString());
}
sw1.Close();
StreamWriter sw2 = new StreamWriter(@"F:\yelp_RS\data\exp\businesstext");
foreach (var v in businesstextdic)
1294 {
    List<string> t = v.Value;
    t.Sort();
    StringBuilder sb = new StringBuilder();
    sb.Append(v.Key + "||||");
1299     foreach (var va in t)
        sb.Append(va).Append(" ");
    sw2.WriteLine(sb.ToString());
}
sw2.Close();
1304 }

```

```

public Dictionary<string, int> biabusinessname = new Dictionary<string, int>();
public Dictionary<string, int> bialocation = new Dictionary<string, int>();
1309 public Dictionary<string, int> bialat = new Dictionary<string, int>();
public Dictionary<string, int> bialon = new Dictionary<string, int>();
public Dictionary<string, int> biazipcode = new Dictionary<string, int>();
void loadBiasData()
{
1314     string path = @"F:\Dropbox\RecSysChallenge\Features\i";
    string line = "";
    StreamReader srbyname = new StreamReader(path + "Qiang_9998_BusinessName_Normalized.txt");
    line = srbyname.ReadLine();
    while ((line = srbyname.ReadLine()) != null)
1319     {
        string[] splits = line.Split('\t');
        string[] s = splits[1].Split(':');
        biabusinessname.Add(map[int.Parse(splits[0])], int.Parse(s[0]));
    }
1324     srbyname.Close();

    StreamReader srzipcode = new StreamReader(path + "Qiang_197_zipcode.txt");
    line = srzipcode.ReadLine();
    while ((line = srzipcode.ReadLine()) != null)
1329     {
        string[] splits = line.Split('\t');

```

```

        string [] s = splits [1]. Split (':');
        biazipcode.Add(map[int.Parse(splits[0]) ], int.Parse(s[0]) );
    }
1334 srzipcode.Close();

    StreamReader srlocation = new StreamReader(path + "Liang_ItemLocation.txt");
    line = srlocation.ReadLine();
    while ((line = srlocation.ReadLine()) != null)
1339 {
        string [] splits = line.Split ('\t');
        string [] s = splits [1]. Split (':');
        bialocation.Add(map[int.Parse(splits[0]) ], int.Parse(s[0]) );
    }
1344 srlocation.Close();

    StreamReader srlat = new StreamReader(path + "Liang_ItemLatitude.txt");
    line = srlat.ReadLine();
    while ((line = srlat.ReadLine()) != null)
1349 {
        string [] splits = line.Split ('\t');
        string [] s = splits [1]. Split (':');
        bialat.Add(map[int.Parse(splits[0]) ], int.Parse(s[0]) );
    }
1354 srlat.Close();
    StreamReader srlon = new StreamReader(path + "Liang_ItemLocation.txt");
    line = srlon.ReadLine();
    while ((line = srlon.ReadLine()) != null)
1359 {
        string [] splits = line.Split ('\t');
        string [] s = splits [1]. Split (':');
        bialon.Add(map[int.Parse(splits[0]) ], int.Parse(s[0]) );
    }
    srlon.Close();
1364 }

    public void run(string path)
    {
        string gepath = path + "generated";
1369 string trainpath = path + "yelp_training_set";
        string testpath = path + "final_test_set";

        loadMap(path + "features");

1374 loadUidIndex(trainpath);
        loadBidIndex(trainpath);

        loadUserData(path);
        loadBusinessData(path);
1379 loadTrainReviewData(trainpath);
        loadTrainCheckinData(trainpath);

        loadTestReviewData(testpath);
        loadTestCheckinData(testpath);
1384 calCheckin();

        loadGenerateData(gepath);

        ////from
1389 loadJiaobenjunData(path + "jbfeature");

        loadBiasData();
    }
1394 }
}

```

- *FeatureGenerator2.cs*

generate files for models from raw data.

```

    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
5    using System.IO;
    using Newtonsoft.Json;

    using yelpRS.dataset.details;
    using yelp_RS.dataset;
10   using yelpRS.dataset;

    namespace yelp_RS.dataset
    {
        public class FeatureGenerator2
15         {
            double globelave = 3.7667;
            double uservaave = 0.5091;
            double itemvaave = 1.1485;
            double maxlon = -111.2635082;
            double minlon = -112.875482;
20         double maxlat = 34.002867;
            double minlat = 32.876848;
            int maxfun = 24519;
            int maxuseful = 24293;
            int maxcool = 122410;
25         int maxreviewcount = 22977;
            int maxuserreviewcount = 5807;

            string generatOnline(DataSet2 ds, datainfo dif, LoadTopicData ltd)
30         {
                string uid = dif.uid;
                string bid = dif.bid;
                int featuresum = 1;
                StringBuilder sb = new StringBuilder();
35                //user id
                sb.Append(ds.uidindex[uid] + featuresum).Append(":1_");
                featuresum += ds.uidindex.Count + 1;
                ///business id
                sb.Append(ds.bidindex[bid] + featuresum).Append(":1_");
40                featuresum += ds.bidindex.Count + 1;

                string line = sb.ToString();
                return line;
            }
45        }

        string generatOnline2(DataSet2 ds, datainfo dif, LoadTopicData ltd)
        {
            string uid = dif.uid;
            string bid = dif.bid;
50            int featuresum = 1;
            StringBuilder sb = new StringBuilder();
            //user id
            sb.Append(ds.uidindex[uid]).Append("_");
            ///business id
            sb.Append(ds.bidindex[bid]).Append("_");
55            //userfeature
            //globlereviewcount
            int ucount = 0;
            ucount = (int)(Math.Log(ds.userlist[uid].reviewcount + 1) / 0.1);
60            sb.Append(ucount + featuresum).Append(":1_");
            featuresum += 120;
            ///businessfeature
            if (ds.businesslist[bid].isopen)
                sb.Append(featuresum + 1).Append(":1_");
65            else
                sb.Append(featuresum + 2).Append(":1_");
            featuresum += 3;
            ///globlereviewcount

```



```

70     int bcount = 0;
    bcount = (int)(Math.Log(ds.businesslist[bid].review_count + 1) / 0.1);
    sb.Append(bcount + featuresum).Append(":1_");
    featuresum += 170;
    ///tagid
    List<int> tagidlist = new List<int>();
75     int tagcount = 0;
    try
    {
        foreach (var v in ds.businesslist[bid].categories)
        {
80             tagidlist.Add(ds.tagidindex[v]);
            try
            {
                if (ds.jbcattypedic[v] == 1)
                {
85                     tagcount++;
                }
            }
            catch { }
90        }
        tagidlist.Sort();
        foreach (var v in tagidlist)
            sb.Append(featuresum + v).Append(":1_");
    }
95     catch { }
    featuresum += 1000;
    ///timebin
    int timebin = ds.jbusertimebindic[uid + bid];
    sb.Append(featuresum + timebin).Append(":1_");
100    featuresum += 22;

    ///checkincount
    int reviewcount = 0;
    try
105    {
        reviewcount = ds.traincheckincount[bid];
    }
    catch
    {
110        try
        {
            reviewcount = ds.testcheckincount[bid];
        }
        catch { }
115    }
    reviewcount = Math.Min(reviewcount, 3000);
    reviewcount /= 10;
    sb.Append(featuresum + reviewcount).Append(":1_");
    featuresum += 305;

120    ///ckinhour&day
    try
    {
125        for (int i = 0; i < 24; i++)
        {
            if (ds.checkinhourlist[bid][i] > 10)
            {
                double value = Math.Min(1, (double)ds.checkinhourlist[bid][i] / 200);
                sb.Append(featuresum + i).Append(":").Append(value).Append("_");
130            }
        }
    }
    catch { }
    featuresum += 26;
135    try
    {

```

```

    for (int i = 0; i < 7; i++)
    {
        if (ds.checkindaylist [bid][i] > 10)
        {
            double value = Math.Min(1, (double)ds.checkindaylist[bid][i] / 200);
            sb.Append(featuresum + i).Append(":").Append(value).Append(" ");
        }
    }
}
catch { }
featuresum += 10;
///vote
try
{
    int fun = (int)(100 * Math.Log((ds.userlist [uid]. votefun + 1)));
    //int
    sb.Append(featuresum + fun).Append(":1");
}
catch
{ }
featuresum += 1015;
try
{
    int fun = (int)(100 * Math.Log((ds.userlist [uid]. voteuserful + 1)));
    sb.Append(featuresum + fun).Append(":1");
}
catch
{ }
featuresum += 1015;
try
{
    int fun = (int)(100 * Math.Log((ds.userlist [uid]. votecool + 1)));
    sb.Append(featuresum + fun).Append(":1");
}
catch
{ }
featuresum += 1005;

//text&businessnam&streetname
//businessname
string fullname = "";
string name = "";
fullname = ds.businesslist [bid].name.Trim();
string [] names = fullname.Split(' ');
name = names[0].Trim();
try
{
    sb.Append(featuresum + ds.businessnameindex[name] + ":1");
}
catch
{
    Console.WriteLine(name);
}
featuresum += 5314;

sb.Append(featuresum + ds.streetnameindex[bid]).Append(":1");
featuresum += 88;

sb.Append(featuresum + ds.jblantdic[bid]).Append(":1");
featuresum += 52;
sb.Append(featuresum + ds.jblongdic[bid]).Append(":1");
featuresum += 52;
sb.Append(featuresum + ds.jblocationdic[bid]).Append(":1");
featuresum += 52;

sb.Append(featuresum + ds.jbzipcodedic[bid]).Append(":1");
featuresum += 200;

```

```

205 //topic
    try
    {
        List<double> usertopic = ds.userreviewtopic[uid];
        for (int i = 0; i < usertopic.Count; i++)
210     {

            sb.Append(featuresum + i).Append(":").Append(usertopic[i]).Append(" ");
        }
    }
215 catch
    {
    }
    featuresum += 6;

220 //businessstopic
    try
    {
        List<double> businesstopic = ds.businessreviewtopic[bid];
        for (int i = 0; i < businesstopic.Count; i++)
225     {
            sb.Append(featuresum + i).Append(":").Append(businesstopic[i]).Append(" ");
        }
    }
    catch { }
230 featuresum += 6;
    string line = sb.ToString();
    return line;
}

235 string generatOnline3(DataSet2 ds, datainfo dif, LoadTopicData ltd)
{
    string uid = dif.uid;
    string bid = dif.bid;
    StringBuilder sb = new StringBuilder();
240 //userfeature

    double uscore = 0;
    try
    {
245         uscore = ((double)ds.userlist [uid].avescore / 5);
    }
    catch
    {
        uscore = ((double)globelave / 5);
250     }
    sb.Append(uscore + 0.0001).Append(" ");
    ///businessfeature

    if (ds.businesslist [bid].isopen)
255         sb.Append("0.5");
    else
        sb.Append("1");

    ///timebin
260 double timebin = (double)ds.jbusertimebindic[uid + bid] / 20;
    sb.Append(timebin + 0.0001).Append(" ");
    //ckinhour&day
    try
    {
265         for (int i = 0; i < 24; i++)
        {
            double value = Math.Min(1, (double)ds.checkinhourlist[bid][i] / 200);
            sb.Append(value + 0.0001).Append(" ");
        }
270     }
    catch
    {

```

```

        for (int i = 0; i < 24; i++)
        {
275             sb.Append("0.0001");
        }
    }
    //text&businessnam&streetname
    //streetname
280    sb.Append((double)ds.streetnameindex[uid] / 86).Append(" ");

    sb.Append((double)ds.jblantdic[uid] / 50 + 0.0001).Append(" ");
    sb.Append((double)ds.jblongdic[uid] / 50 + 0.0001).Append(" ");
285    sb.Append((double)ds.jblocationdic[uid] / 50 + 0.0001).Append(" ");
    sb.Append((double)ds.jbzipcodedic[uid] / 197 + 0.0001).Append(" ");
    sb.Append((double)ds.jbitemcity[uid] / 69 + 0.0001).Append(" ");
    sb.Append((double)ds.jbitemreview[uid] / 51 + 0.0001).Append(" ");
    sb.Append((double)ds.jbitemstate[uid] / 4 + 0.0001).Append(" ");
290    sb.Append((double)ds.jbusergender[uid] / 3 + 0.0001).Append(" ");
    try
    {
        sb.Append(ds.jbusenamelen[uid] / 26 + 0.0001).Append(" ");
    }
295    catch
    {
        sb.Append("0.5");
    }
    try
300    {
        sb.Append((double)ds.jbuserreview[uid] / 51 + 0.0001).Append(" ");
    }
    catch
    {
305        sb.Append("0.00001");
    }
    //topic
    try
    {
310        List<double> usertopic = ds.userreviewtopic[uid];
        for (int i = 0; i < usertopic.Count; i++)
        {
            sb.Append(usertopic[i] + 0.0001).Append(" ");
        }
315    }
    catch
    {
        sb.Append("0.2_0.2_0.2_0.2_0.2");
    }
320
    //businesstopic
    try
    {
        List<double> businesstopic = ds.businessreviewtopic[uid];
325        for (int i = 0; i < businesstopic.Count; i++)
        {
            sb.Append(businesstopic[i] + 0.0001).Append(" ");
        }
    }
330    catch
    {
        sb.Append("0.2_0.2_0.2_0.2_0.2");
    }
    string line = sb.ToString();
335    return line;
}

string generatOnline4(DataSet2 ds, datainfo dif, LoadTopicData ltd)
{
340    string uid = dif.uid;

```

```

string bid = dif.bid;
int featuresum = 1;
StringBuilder sb = new StringBuilder();

345 sb.Append(ds.uidindex[uid]).Append(" ");
    ///business id
sb.Append(ds.bidindex[bid]).Append(" ");
    ///bias
sb.Append(ds.biabusinessname[bid]).Append(" ");
350 sb.Append(ds.bialat[bid]).Append(" ");
sb.Append(ds.bialon[bid]).Append(" ");
sb.Append(ds.bialocation[bid]).Append(" ");
sb.Append(ds.biazipcode[bid]);
    ///user id
355 sb.Append("||");
sb.Append(ds.uidindex[uid] + featuresum).Append(":1");
featuresum += ds.uidindex.Count + 1;
    ///business id
sb.Append(ds.bidindex[bid] + featuresum).Append(":1");
360 featuresum += ds.bidindex.Count + 1;
    ///userfeature
int uscore = 0;
try
{
365     uscore = (int)(ds.userlist[uid].avescore / 0.2);
}
catch
{
    uscore = (int)(globelave / 0.2);
370 }
sb.Append(uscore + featuresum).Append(":1");
featuresum += 27;
    ///businessfeature

375 if (ds.businesslist[bid].isopen)
    sb.Append(featuresum + 1).Append(":1");
else
    sb.Append(featuresum + 2).Append(":1");
featuresum += 3;
    ///tagid
List<int> tagidlist = new List<int>();
int tagcount = 0;
try
{
385     foreach (var v in ds.businesslist[bid].categories)
    {
        tagidlist.Add(ds.tagidindex[v]);
        try
        {
390             if (ds.jbcattypedic[v] == 1)
            {

                tagcount++;
            }
        }
        catch { }
    }
    tagidlist.Sort();
    foreach (var v in tagidlist)
400         sb.Append(featuresum + v).Append(":1");
}
catch { }
featuresum += 1000;

405 ///timebin
int timebin = ds.jbusertimebindic[uid + bid];
sb.Append(featuresum + timebin).Append(":1");
featuresum += 22;

```

```

410 //ckinhour&day
    try
    {
        for (int i = 0; i < 24; i++)
        {
            if (ds.checkinhourlist [bid][i] > 10)
            {
                double value = Math.Min(1, (double)ds.checkinhourlist[bid][i] / 200);
                sb.Append(featuresum + i).Append(":").Append(value).Append("\n");
            }
        }
    }
420 catch { }
    featuresum += 26;
    try
    {
        for (int i = 0; i < 7; i++)
        {
            if (ds.checkindaylist [bid][i] > 10)
            {
                double value = Math.Min(1, (double)ds.checkindaylist[bid][i] / 200);
                sb.Append(featuresum + i).Append(":").Append(value).Append("\n");
            }
        }
    }
435 catch { }
    featuresum += 10;
    ///vote
    try
    {
        int fun = (int)(100 * Math.Log((ds.userlist [uid]. vote fun + 1)));
        //int
        sb.Append(featuresum + fun).Append(":1\n");
    }
    catch
    { }
445 featuresum += 1200;
    try
    {
        int fun = (int)(100 * Math.Log((ds.userlist [uid]. voteuserful + 1)));
        sb.Append(featuresum + fun).Append(":1\n");
    }
    catch
    { }
450 featuresum += 1200;
    try
    {
        int fun = (int)(100 * Math.Log((ds.userlist [uid]. votecool + 1)));
        sb.Append(featuresum + fun).Append(":1\n");
    }
    catch
    { }
460 featuresum += 1200;

    //text&businessnam&streetname
    //businessname
465 string fullname = "";
    string name = "";
    fullname = ds.businesslist [bid]. name.Trim();
    string [] names = fullname.Split(' ');
    name = names[0].Trim();
470 try
    {
        sb.Append(featuresum + ds.businessnameindex[name] + ":1\n");
    }
    catch
    {
475 Console.WriteLine(name);

```

```

    }
    featuresum += 5314;
    //streetname
480 sb.Append(featuresum + ds.streetnameindex[bid]).Append(":1_");
    featuresum += 88;

    sb.Append(featuresum + ds.jblantdic[bid]).Append(":1_");
    featuresum += 52;
485 sb.Append(featuresum + ds.jblongdic[bid]).Append(":1_");
    featuresum += 52;
    sb.Append(featuresum + ds.jblocationdic[bid]).Append(":1_");
    featuresum += 52;
    sb.Append(featuresum + ds.jbzipcodedic[bid]).Append(":1_");
490 featuresum += 200;
    sb.Append(featuresum + ds.jbitemcity[bid]).Append(":1_");
    featuresum += 71;
    sb.Append(featuresum + ds.jbitemreview[bid]).Append(":1_");
    featuresum += 54;
495 sb.Append(featuresum + ds.jbitemstate[bid]).Append(":1_");
    featuresum += 6;
    //sb.Append(featuresum + ds.jbqch[bid]).Append(":1 ");
    //featuresum += 23;
    sb.Append(featuresum + ds.jbqbn[bid]).Append(":1_");
500 featuresum += 10000;
    sb.Append(featuresum + ds.jbusergender[uid]).Append(":1_");
    featuresum += 4;
    try
    {
505 sb.Append(featuresum + ds.jbusernamelen[uid]).Append(":1_");
    }
    catch { }
    featuresum += 28;
    try
510 {
        sb.Append(featuresum + ds.jbuserreview[uid]).Append(":1_");
    }
    catch { }
    featuresum += 54;
515 //topic
    try
    {
        List<double> usertopic = ds.userreviewtopic[uid];
        for (int i = 0; i < usertopic.Count; i++)
520 {
            //if (usertopic[i] > 0.3)
            sb.Append(featuresum + i).Append(":").Append(usertopic[i]).Append("_");
        }
    }
525 catch
    {
    }
    featuresum += 6;
    //businesstopic
530 try
    {
        List<double> businesstopic = ds.businessreviewtopic[bid];
        for (int i = 0; i < businesstopic.Count; i++)
        {
535 //f (businesstopic[i] > 0.3)
            sb.Append(featuresum + i).Append(":").Append(businesstopic[i]).Append("_");
        }
    }
    catch { }
540 featuresum += 6;

    string line = sb.ToString();
    return line;
}

```

```

545 string generatOnlinebac(DataSet2 ds, datainfo dif, LoadTopicData ltd)
{
    string uid = dif.uid;
    string bid = dif.bid;
550     int featuresum = 1;
    StringBuilder sb = new StringBuilder();
    //user id
    sb.Append(ds.uidindex[uid] + featuresum).Append(":1_");
    featuresum += ds.uidindex.Count + 1;
555     ///business id
    sb.Append(ds.bidindex[bid] + featuresum).Append(":1_");
    featuresum += ds.bidindex.Count + 1;
    ///globlereviewcount
    int ucount = 0;
560     ucount = (int)(Math.Log(ds.userlist[uid].reviewcount + 1) / 0.1);
    sb.Append(ucount + featuresum).Append(":1_");
    featuresum += 120;
    int bcount = 0;
    bcount = (int)(Math.Log(ds.businesslist[bid].review_count + 1) / 0.1);
565     sb.Append(bcount + featuresum).Append(":1_");
    featuresum += 170;
    ///tagid
    List<int> tagidlist = new List<int>();
    int tagcount = 0;
570     try
    {
        foreach (var v in ds.businesslist[bid].categories)
        {
            tagidlist.Add(ds.tagidindex[v]);
575             try
            {
                if (ds.jbcattypedic[v] == 1)
                {
                    tagcount++;
580                 }
            }
            catch { }
        }
        tagidlist.Sort();
        foreach (var v in tagidlist)
        {
            sb.Append(featuresum + v).Append(":1_");
        }
        catch { }
590     featuresum += 1000;

    ///checkincount
    int reviewcount = 0;
    try
595     {
        reviewcount = ds.traincheckincount[bid];
    }
    catch
    {
        try
600        {
            reviewcount = ds.testcheckincount[bid];
        }
        catch { }
605    }
    reviewcount = Math.Min(reviewcount, 3000);
    reviewcount /= 10;
    sb.Append(featuresum + reviewcount).Append(":1_");
    featuresum += 305;

610     //ckinhour&day
    try

```



```

{
    for (int i = 0; i < 24; i++)
    {
        if (ds.checkinhourlist [bid][i] > 10)
        {
            double value = Math.Min(1, (double)ds.checkinhourlist[bid][i] / 200);
            sb.Append(featuresum + i).Append(":").Append(value).Append(" ");
        }
    }
}
catch { }
featuresum += 26;
try
{
    for (int i = 0; i < 7; i++)
    {
        if (ds.checkindaylist [bid][i] > 10)
        {
            double value = Math.Min(1, (double)ds.checkindaylist[bid][i] / 200);
            sb.Append(featuresum + i).Append(":").Append(value).Append(" ");
        }
    }
}
catch { }
featuresum += 10;

///vote
try
{
    int fun = (int)(100 * Math.Log((ds.userlist [uid]. votefun + 1)));
    //int
    sb.Append(featuresum + fun).Append(":1");
}
catch
{ }
featuresum += 1015;
try
{
    int fun = (int)(100 * Math.Log((ds.userlist [uid]. voteuserful + 1)));
    sb.Append(featuresum + fun).Append(":1");
}
catch
{ }
featuresum += 1015;
try
{
    int fun = (int)(100 * Math.Log((ds.userlist [uid]. votecool + 1)));
    sb.Append(featuresum + fun).Append(":1");
}
catch
{ }
featuresum += 1005;

//text&businessnam&streetname
//businessname
string fullname = "";
string name = "";
fullname = ds.businesslist [bid].name.Trim();
string [] names = fullname.Split(' ');
name = names[0].Trim();
try
{
    sb.Append(featuresum + ds.businessnameindex[name] + ":1");
}
catch
{
    Console.WriteLine(name);
}

```

```

        featuresum += 5314;

        sb.Append(featuresum + ds.streetnameindex[bid]).Append(":1_");
        featuresum += 88;

685         sb.Append(featuresum + ds.jblantdic[bid]).Append(":1_");
        featuresum += 52;
        sb.Append(featuresum + ds.jblongdic[bid]).Append(":1_");
        featuresum += 52;
690         sb.Append(featuresum + ds.jblocationdic[bid]).Append(":1_");
        featuresum += 52;

        sb.Append(featuresum + ds.jbzipcodedic[bid]).Append(":1_");
        featuresum += 200;
695         string line = sb.ToString();
        return line;
    }

    void trainDataGenerator(DataSet2 ds, string path, LoadTopicData ltd)
700    {
        Console.WriteLine("-----TrainDataGenerating-----");
        StreamWriter sw = new StreamWriter(path + "train");
        foreach (var v in ds.traindatalist)
        {
705             string line = v.Value.stars + "_";
            line += generatOnline4(ds, v.Value, ltd);

            sw.WriteLine(line);
        }
710        sw.Close();
        Console.WriteLine("-----TrainDataGeneratingOver-----");
    }

    void testDataGenerator(DataSet2 ds, string path, LoadTopicData ltd)
715    {
        Console.WriteLine("-----TestDataGenerating-----");
        StreamWriter sw = new StreamWriter(path + "test");
        foreach (var v in ds.testdatalist)
        {
720             string line = v.stars + "_";
            line += generatOnline4(ds, v, ltd);

            sw.WriteLine(line);
        }
725        sw.Close();

        Console.WriteLine("-----TestDataGeneratingOver-----");
    }

730    void blendingDataGenerator(DataSet2 ds, string path, LoadTopicData ltd)
    {
        string rpath = @"F:\    \yelp_RS\data\yelp_training_set";
        Console.WriteLine("-----BlendingTrainDataGenerating-----");
        for (int i = 1; i <= 7; i++)
735        {
            string filename = rpath + "blendingfile\\ local_final_train_set_review_ " + i + ".json";
            string testfilename = rpath + "blendingfile\\ local_final_test_set_review_ " + i + ".json";
            string outfilename = path + "feature\\blending2\\train" + i;
            string testoutfilename = path + "feature\\blending2\\test" + i;
740            StreamReader sr = new StreamReader(filename);
            StreamReader sr2 = new StreamReader(testfilename);
            StreamWriter sw = new StreamWriter(outfilename);
            StreamWriter sw2 = new StreamWriter(testoutfilename);
            string line = "";
745            while ((line = sr.ReadLine()) != null)
            {
                JsonReader jr = new JsonTextReader(new StringReader(line));
                string lastline = "";

```

```

750     string rid = "";
    datainfo dif = new datainfo();
    while (jr.Read())
    {
        if (lastline == "review_id")
            rid = jr.Value.ToString();
755         if (jr.ValueType != null)
            lastline = jr.Value.ToString();
    }
    string newline = ds.traindatalist[rid].stars + "␣";
    newline += generatOnline4(ds, ds.traindatalist[rid], ltd);
760    sw.WriteLine(newline);
}
while ((line = sr2.ReadLine()) != null)
{
    JsonReader jr = new JsonTextReader(new StringReader(line));
765    string lastline = "";
    string rid = "";
    datainfo dif = new datainfo();
    while (jr.Read())
    {
770        if (lastline == "review_id")
            rid = jr.Value.ToString();
        if (jr.ValueType != null)
            lastline = jr.Value.ToString();
    }
775    string newline = ds.traindatalist[rid].stars + "␣";
    newline += generatOnline4(ds, ds.traindatalist[rid], ltd);
    sw2.WriteLine(newline);
}
sr.Close();
780 sr2.Close();
sw.Close();
sw2.Close();
}
Console.WriteLine("-----BlendingTrainDataGeneratingOver-----");
785 }

public void run(DataSet2 ds, string path, LoadTopicData ltd)
{
    trainDataGenerator(ds, path + "feature", ltd);
790    testDataGenerator(ds, path + "feature", ltd);
    blendingDataGenerator(ds, path, ltd);
}
}
}

```

- *LinearSVDModel.cs*

a linear svd model.

```

1  using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.IO;
6
    namespace LinearSVD
    {
        public class Record
        {
11         public int UserId { set; get; }
            public int ItemId { set; get; }
            public double Rating { set; get; }
            public double GEPredict { set; get; }

16         public List<int> BiasID { set; get; }

            public double Predict { set; get; }
        }
    }
}

```

```

public Dictionary<int, double> feature { set; get; }

21 public Record(int u, int m, double r, Dictionary<int, double> f)
    {
        this.UserId = u;
        this.ItemId = m;
        this.Rating = r;
26     this.feature = f;
    }

public Record(int u, int m, double r, List<int> bias, Dictionary<int, double> f)
    {
31     this.UserId = u;
        this.ItemId = m;
        this.Rating = r;
        this.BiasID = bias;
        this.feature = f;
36     }
}

public class LinearSVDModel
{
41     private const int STEP = 300;

    private double weight = 0;
    private const int K = 2;
    private double lr = 0.016;
46     private const double reg = 0.004;

    private double alpha = 1;
    private const double beta = 0.003;

51     private List<Record> train = new List<Record>();
    private List<Record> test = new List<Record>();

    private int USER_CNT = 0;
    private int ITEM_CNT = 0;
56     private int FEATURE_CNT = 0;
    private double MAX_RATING = 0;
    private double MIN_RATING = 0;

    private double mean = 0;
61     private double[] bu ;
    private double[] bi ;
    private double[,] p ;
    private double[,] q ;

66     private double[] b;
    private double bm;

    Random _rand = new Random();

71     private double _rand_init ()
    {
        return (_rand.NextDouble() - 0.5) / 10000;
    }

76     private int LoadRatingData(string trainfile, string testfile )
    {
        int maxucnt = 0;
        int maxicnt = 0;
        int maxfcnt = 0;
81     double maxrating = 0;
        double minrating = 100000;
        try
        {
            using (StreamReader sr = new StreamReader(trainfile))
86             {

```

```

Console.WriteLine("=====loading_Train_Data=====");
string line = "";
while ((line = sr.ReadLine()) != null)
{
    line = line.Trim();
    string[] splits = line.Split(' ');
    int uid = int.Parse(splits[1]);
    if (uid > maxucnt) maxucnt = uid;
    int iid = int.Parse(splits[2]);
    if (iid > maxicnt) maxicnt = iid;
    double rating = double.Parse(splits[0]);
    if (rating > maxrating) maxrating = rating;
    if (rating < minrating) minrating = rating;
    Dictionary<int, double> tmpdic = new Dictionary<int, double>();
    for (int i = 3; i < splits.Length; i++)
    {
        string[] segs = splits[i].Split(':');
        tmpdic.Add(int.Parse(segs[0]), double.Parse(segs[1]));
        if ((i == splits.Length - 1) && (int.Parse(segs[0]) > maxfcnt))
            maxfcnt = int.Parse(segs[0]);
    }
    train.Add(new Record(uid, iid, rating, tmpdic));
}
Console.WriteLine("=====loading_Train_Data_Over=====");
}
}
catch { return 1; }
try
{
    using (StreamReader sr = new StreamReader(testfile))
    {
        Console.WriteLine("=====loading_Test_Data=====");
        string line = "";
        while ((line = sr.ReadLine()) != null)
        {
            line = line.Trim();
            string[] splits = line.Split(' ');
            int uid = int.Parse(splits[1]);
            if (uid > maxucnt) maxucnt = uid;
            int iid = int.Parse(splits[2]);
            if (iid > maxicnt) maxicnt = iid;
            double rating = double.Parse(splits[0]);
            if (rating > maxrating) maxrating = rating;
            if (rating < minrating) minrating = rating;
            Dictionary<int, double> tmpdic = new Dictionary<int, double>();
            for (int i = 3; i < splits.Length; i++)
            {
                string[] segs = splits[i].Split(':');
                tmpdic.Add(int.Parse(segs[0]), double.Parse(segs[1]));
                if ((i == splits.Length - 1) && (int.Parse(segs[0]) > maxfcnt))
                    maxfcnt = int.Parse(segs[0]);
            }
            test.Add(new Record(uid, iid, rating, tmpdic));
        }
    }
}
catch { return 2; }
Console.WriteLine("=====loading_Test_Data_Over=====");
Console.WriteLine("max_UID:" + (maxucnt));
Console.WriteLine("max_IID:" + (maxicnt));
Console.WriteLine("FEATURE_CONUT:" + (maxfcnt + 1).ToString());
Console.WriteLine("max_Rating:" + maxrating);
Console.WriteLine("min_Rating:" + minrating);
USER_CNT = maxucnt;
ITEM_CNT = maxicnt;
FEATURE_CNT = maxfcnt + 1;
MAX_RATING = maxrating;
MIN_RATING = minrating;

```

```

156         return 0;
    }

    private void Initialize ()
    {
        bu = new double[USER_CNT + 1];
161        bi = new double[ITEM_CNT + 1];
        p = new double[USER_CNT + 1, K];
        q = new double[ITEM_CNT + 1, K];
        b = new double[FEATURE_CNT + 1];
        mean = train.Sum(x => x.Rating) / train.Count;
166        for (int u = 0; u <= USER_CNT; u++)
        {
            bu[u] = _rand_init();
            for (int k = 0; k < K; k++)
            {
171                p[u, k] = _rand_init();
            }
        }
        for (int i = 0; i <= ITEM_CNT; i++)
        {
176            bi[i] = _rand_init();
            for (int k = 0; k < K; k++)
            {
                q[i, k] = _rand_init();
            }
181        }
        bm = _rand_init();
        for (int i = 0; i <= FEATURE_CNT; i++)
        {
186            b[i] = 0;
        }
    }

    private double Predict(Record r)
    {
191        int u = r.UserId;
        int i = r.ItemId;
        double svdpred = mean + bu[u] + bi[i];
        for (int k = 0; k < K; k++)
        {
196            svdpred += p[u, k] * q[i, k];
        }
        svdpred = Math.Max(svdpred, MIN_RATING);
        svdpred = Math.Min(svdpred, MAX_RATING);
        double logit = 0;
201        foreach (var v in r.feature)
        {
            logit += b[v.Key] * v.Value;
        }
        double linearpred = mean + bm + MAX_RATING * (1 / (1 + Math.Exp(-1 * logit)));
206        linearpred = Math.Max(linearpred, MIN_RATING);
        linearpred = Math.Min(linearpred, MAX_RATING);

        return (weight * svdpred + (1.0 - weight) * linearpred);
    }

211    private void Train()
    {
        Console.WriteLine("=====Training_Model=====");
        for (int s = 0; s < STEP; s++)
        {
216            double rn = 1 / (Math.Sqrt((double)(1 + FEATURE_CNT)));
            double train_rmse = 0;
            double test_rmse = 0;
            foreach (Record r in train.OrderBy(x => Guid.NewGuid()))
            {
221                int user = r.UserId;

```

```

226         int item = r.ItemId;
        double rui = r.Rating;
        Dictionary<int, double> f = r.feature;
        double pui = Predict(r);
        double eui = rui - pui;
        bu[user] += lr * (eui - reg * bu[user]);
        bi[item] += lr * (eui - reg * bi[item]);
        for (int k = 0; k < K; k++)
231     {
            p[user, k] += lr * (eui * q[item, k] - reg * p[user, k]);
            q[item, k] += lr * (eui * p[user, k] - reg * q[item, k]);
        }

236         bm += alpha * (eui - beta * bm);
        foreach (var v in f)
        {
            b[v.Key] += alpha * (eui * v.Value * rn - beta * b[v.Key]);
        }
    }
241     lr *= 0.9;
    alpha *= 0.95;

    foreach (Record r in train)
246     {
        double pui = Predict(r);
        double eui = r.Rating - pui;
        train_rmse += eui * eui;
    }
    foreach (Record r in test)
251     {
        double pui = Predict(r);
        double eui = r.Rating - pui;
        test_rmse += eui * eui;
    }
256     Console.WriteLine("{0}\t{1}\t{2}", s, Math.Sqrt(train_rmse / train.Count), Math.Sqrt(test_rmse / test.Count)
    );
}

}

261 private void WriteToFile(string outfile)
{
    try
    {
        StreamWriter sw = new StreamWriter(outfile);
266         foreach (var v in test)
        {
            sw.WriteLine(Predict(v));
        }
        sw.Close();
    }
271     catch { Console.WriteLine("Write_to_File_Error"); }
}

public void run(string trainfile, string testfile, string outfile)
276 {
    int loaddatainfo = LoadRatingData(trainfile, testfile);
    if (loaddatainfo == 1)
    {
        Console.WriteLine("Loading_TrainData_Error:");
281     }
    else if (loaddatainfo == 2)
    {
        Console.WriteLine("Loading_TestData_Error:");
    }
286     else
    {
        Initialize ();
        Train();
    }
}

```

```

291         if ( outfile != null )
        {
            WriteToFile(outfile);
        }
    }
}
296 }
}

```

- *LinearBiasSVDModel.cs*

a linear svd model adding biases.

```

using System;
using System.Collections.Generic;
3 using System.Linq;
using System.Text;
using System.IO;

namespace LinearSVD
8 {

    public class LinearBiasSVDModel
    {
        private const int STEP = 300;
13         private const int BIAS_CNT = 10;

        private double weight = 0;
        private const int K = 1;
        private double lr = 0.016;
18         private const double reg = 0.004;

        private double alpha = 1;
        private const double beta = 0.003;

23         private List<Record> train = new List<Record>();
        private List<Record> test = new List<Record>();

        private int USER_CNT = 0;
        private int ITEM_CNT = 0;
28         private int FEATURE_CNT = 0;
        private double MAX_RATING = 0;
        private double MIN_RATING = 0;
        private int[] MAX_BIAS;
        private int BIAS_COUNT = 0;

33         private double mean = 0;
        private double[] bu;
        private double[] bi;
        private double[,] p;
38         private double[,] q;
        private List<double[]> bias = new List<double[]>();

        private double[] b;
        private double bm;
43         private double bs;

        Random _rand = new Random();

        private double _rand_init()
48         {
            return (_rand.NextDouble() - 0.5) / 100000;
        }

        private int LoadRatingData(string trainfile, string testfile )
53         {
            int maxucnt = 0;
            int maxicnt = 0;
            int maxfent = 0;

```



```

double maxrating = 0;
double minrating = 100000;
58 int [] maxbias = new int[BIAS.CNT];
for (int i = 0; i < BIAS.CNT; i++)
    maxbias[i] = 0;
int featurekind = 0;
63 try
{
    using (StreamReader sr = new StreamReader(trainfile))
    {
        Console.WriteLine("====loading_Train_Data====");
        string line = "";
        68 while ((line = sr.ReadLine()) != null)
        {
            line = line.Trim();
            string [] parts = line.Split(new string [] { "||| " }, StringSplitOptions.None);
            73 string [] splits = parts[0].Trim().Split('_');
            int uid = int.Parse(splits[1]);
            if (uid > maxucnt) maxucnt = uid;
            int iid = int.Parse(splits[2]);
            if (iid > maxicnt) maxicnt = iid;
            78 double rating = double.Parse(splits[0]);
            if (rating > maxrating) maxrating = rating;
            if (rating < minrating) minrating = rating;
            List<int> tmplist = new List<int>();
            featurekind = splits.Length - 3;
            83 if (splits.Length != 3)
            {
                for (int i = 3; i < splits.Length; i++)
                {
                    if (int.Parse(splits[i]) > maxbias[i - 3])
                        88 maxbias[i - 3] = int.Parse(splits[i]);
                    tmplist.Add(int.Parse(splits[i]));
                }
            }

            string [] splits2 = parts[1].Trim().Split('_');
            Dictionary<int, double> tmpdic = new Dictionary<int, double>();
            for (int i = 0; i < splits2.Length; i++)
            {
                98 string [] segs = splits2[i].Split(':');
                tmpdic.Add(int.Parse(segs[0]), double.Parse(segs[1]));
                if ((i == splits2.Length - 1) && (int.Parse(segs[0]) > maxfcnt))
                    maxfcnt = int.Parse(segs[0]);
            }
            103 train.Add(new Record(uid, iid, rating, tmplist, tmpdic));
        }
        Console.WriteLine("====loading_Train_Data_Over====");
    }
}
108 catch { return 1; }
try
{
    using (StreamReader sr = new StreamReader(testfile))
    {
        113 Console.WriteLine("====loading_Test_Data====");
        string line = "";
        while ((line = sr.ReadLine()) != null)
        {
            line = line.Trim();
            118 string [] parts = line.Split(new string [] { "||| " }, StringSplitOptions.None);
            string [] splits = parts[0].Trim().Split('_');
            int uid = int.Parse(splits[1]);
            if (uid > maxucnt) maxucnt = uid;
            int iid = int.Parse(splits[2]);
            if (iid > maxicnt) maxicnt = iid;
            123 double rating = double.Parse(splits[0]);

```

```

128         if (rating > maxrating) maxrating = rating;
        if (rating < minrating) minrating = rating;
        List<int> tmpelist = new List<int>();
        if ( splits.Length != 3)
        {
            for (int i = 3; i < splits.Length; i++)
            {
                if (int.Parse(splits[i]) > maxbias[i - 3])
133                 maxbias[i - 3] = int.Parse(splits[i]);
                tmpelist.Add(int.Parse(splits[i]));
            }
        }

138         string[] splits2 = parts[1].Trim().Split(' ');
        Dictionary<int, double> tmpdic = new Dictionary<int, double>();
        for (int i = 0; i < splits2.Length; i++)
        {
            string[] segs = splits2[i].Split(':');
            tmpdic.Add(int.Parse(segs[0]), double.Parse(segs[1]));
            if ((i == splits2.Length - 1) && (int.Parse(segs[0]) > maxfcnt))
                maxfcnt = int.Parse(segs[0]);
        }
148         test.Add(new Record(uid, iid, rating, tmpelist, tmpdic));
    }
}

catch { return 2; }
153 Console.WriteLine("=====loading_Test_Data_Over=====");
Console.WriteLine("max_UID:" + (maxucnt));
Console.WriteLine("max_IID:" + (maxicnt));
Console.WriteLine("FEATURE_CONUT:" + (maxfcnt + 1).ToString());
Console.WriteLine("max_Rating:" + maxrating);
158 Console.WriteLine("min_Rating:" + minrating);
Console.WriteLine("Bias_CNT:" + featurekind);
USER_CNT = maxucnt;
ITEM_CNT = maxicnt;
FEATURE_CNT = maxfcnt + 1;
163 MAX_RATING = maxrating;
MIN_RATING = minrating;
if (featurekind != 0)
{
    BIAS_COUNT = featurekind;
    MAX_BIAS = new int[featurekind];
    for (int i = 0; i < featurekind; i++)
    {
        MAX_BIAS[i] = maxbias[i];
    }
173 }
return 0;
}

private void Initialize ()
178 {
    bu = new double[USER_CNT + 1];
    bi = new double[ITEM_CNT + 1];
    p = new double[USER_CNT + 1, K];
    q = new double[ITEM_CNT + 1, K];
183 b = new double[FEATURE_CNT + 1];
    if (BIAS_COUNT != 0)
    {
        for (int i = 0; i < BIAS_COUNT; i++)
        {
            double[] t = new double[MAX_BIAS[i] + 1];
            for (int bia = 0; bia < MAX_BIAS[i] + 1; bia++)
                t[bia] = rand.init();
            bias.Add(t);
        }
    }
}

```

```

193     }

    mean = train.Sum(x => x.Rating) / train.Count;
    for (int u = 0; u <= USER_CNT; u++)
198     {
        bu[u] = _rand_init();
        for (int k = 0; k < K; k++)
        {
            p[u, k] = _rand_init();
203        }
    }
    for (int i = 0; i <= ITEM_CNT; i++)
    {
        bi[i] = _rand_init();
208        for (int k = 0; k < K; k++)
        {
            q[i, k] = _rand_init();
        }
    }
213    bm = _rand_init();
    bs = _rand_init();
    for (int i = 0; i <= FEATURE_CNT; i++)
    {
        b[i] = 0;
218    }
}

private double Predict(Record r)
223 {
    int u = r.UserId;
    int i = r.ItemId;
    List<int> bia = r.BiasID;
    double svdpred = mean + bs + bu[u] + bi[i];
228    for (int b = 0; b < BIAS_COUNT; b++)
    {
        svdpred += bias[b][bia[b]];
    }
    for (int k = 0; k < K; k++)
233    {
        svdpred += p[u, k] * q[i, k];
    }

    double logit = 0;
238    foreach (var v in r.feature)
    {
        logit += b[v.Key] * v.Value;
    }
    double linearpred = logit;

243    double re = (weight * svdpred + (1.0 - weight) * linearpred);

    re = Math.Max(re, MIN_RATING);
    re = Math.Min(re, MAX_RATING);
248    return re;
}

private void Train()
{
253    Console.WriteLine("=====Training_Model=====");
    for (int s = 0; s < STEP; s++)
    {
        double rn = 1 / (Math.Sqrt((double)(1 + FEATURE_CNT)));
        double train_rmse = 0;
        double test_rmse = 0;
258        foreach (Record r in train.OrderBy(x => Guid.NewGuid()))
        {

```

```

263     int user = r.UserId;
        int item = r.ItemId;
        double rui = r.Rating;
        List<int> bia = r.BiasID;
        Dictionary<int, double> f = r.feature;
        double pui = Predict(r);
        double eui = rui - pui;
268     bu[user] += lr * (eui - reg * bu[user]);
        bi[item] += lr * (eui - reg * bi[item]);
        for (int i = 0; i < BIAS_COUNT; i++)
        {
            bias[i][bia[i]] += lr * (eui - reg * bias[i][bia[i]]);
273     }
        for (int k = 0; k < K; k++)
        {
            p[user, k] += lr * (eui * q[item, k] - reg * p[user, k]);
            q[item, k] += lr * (eui * p[user, k] - reg * q[item, k]);
278     }

        bs += alpha * (eui - beta * bs);
        bm += alpha * (eui - beta * bm);
        foreach (var v in f)
283     {
            b[v.Key] += alpha * (eui * v.Value * rn - beta * b[v.Key]);
        }
        lr *= 0.95;
288     alpha *= 0.95;

    foreach (Record r in train)
    {
        double pui = Predict(r);
        double eui = r.Rating - pui;
        train_rmse += eui * eui;
    }
    foreach (Record r in test)
    {
298         double pui = Predict(r);
        double eui = r.Rating - pui;
        test_rmse += eui * eui;
    }
    Console.WriteLine("{0}\t{1}\t{2}", s, Math.Sqrt(train_rmse / train.Count), Math.Sqrt(test_rmse / test.Count)
        );
303 }
}

private void WriteToFile(string outfile)
{
308     try
    {
        StreamWriter sw = new StreamWriter(outfile);
        foreach (var v in test)
        {
            sw.WriteLine(Predict(v));
        }
        sw.Close();
    }
    catch { Console.WriteLine("Write_to_File_Error"); }
318 }

public void run(string trainfile, string testfile, string outfile)
{
    int loaddatainfo = LoadRatingData(trainfile, testfile);
323     if (loaddatainfo == 1)
    {
        Console.WriteLine("Loading_TrainData_Error");
    }
    else if (loaddatainfo == 2)

```

```

328         {
            Console.WriteLine("Loading_TestData_Error:");
        }
        else
        {
333             Initialize ();
            Train();
            if ( outfile != null)
            {
338                 WriteToFile(outfile);
            }
        }
    }
}

```

- *test_ensemble.py*

for test ensemble.

```

import numpy as np
import os

```

```

3  e0 = 3.99428
   X_pred = []
   y_rmse = []

8  list_dirs = os.walk("SUBMITTED")
   for root, dirs, files in list_dirs :
       for f in files :
           preds = [float (r.replace(",","'\t').split ('\t')[1]) for r in file ("SUBMITTED/" + f)]
           rmse = float(f)
13      X_pred.append(preds)
           y_rmse.append(rmse)

```

```

X_pred = np.matrix(X_pred).T

```

```

print X_pred[0]

```

```

18 print X_pred.shape

```

```

Z = np.matrix(X_pred)

```

```

N,M = Z.shape

```

```

23 p1 = (Z.T * Z + 0.000006 * np.eye(M)).I

```

```

zmr = []

```

```

for m in range(M):

```

```

28     zm = np.array(X_pred[:,m].T)[0]
        zm_square = np.dot(zm, zm)
        r_square = N * e0 * e0
        nem = N * y_rmse[m] * y_rmse[m]
        a = (r_square + zm_square - nem) / 2
33     zmr.append(a)

```

```

zmr = np.matrix(zmr).T

```

```

w = p1 * zmr

```

```

38 for i in zip(w,y_rmse):

```

```

    print i

```

```

ensemble_pred = np.array((Z * w).T)[0]

```

```

fp = open("out",'w')

```

```

for p in ensemble_pred:

```

```

43     p = max(p,1)

```

```

        p = min(p,5)

```

```

        print >> fp , p

```

```

fp.close()

```

- *crossJoin.py*

for cross join features.

```

import os

def cross(filename1, filename2, outfilename):
4   fin1 = open(filename1)
   fin2 = open(filename2)
   featurecount1 = int(fin1.readline().strip())
   featurecount2 = int(fin2.readline().strip())
   rawfeature1 = dict()
9   rawfeature2 = dict()
   while 1:
       line1 = fin1.readline().strip()
       line2 = fin2.readline().strip()
       if not line1:
14          break
       parts1 = line1.split('\t')
       f = parts1[1].strip().split(':')[0]
       rawfeature1[int(parts1[0])] = float(f)
       parts2 = line2.split('\t')
19          f = parts2[1].strip().split(':')[0]
       rawfeature2[int(parts2[0])] = float(f)
       generatefeatue = dict()
       featuremap = dict()
       for k, v1 in rawfeature1.items():
24          v2 = rawfeature2[k]
           generatefeatue[k] = v1 * featurecount2 + v2
           featuremap[v1 * featurecount2 + v2] = 1
       i = 0
       indexmap = dict()
29          for fk, fv in sorted(featuremap.items(), lambda x, y: cmp(x[1], y[1])):
               indexmap[fk] = i
               #print str(fk) + '\t' + str(i)
               i += 1
       fout = open(outfilename, 'w')
34          fout.write(str(i) + '\n')
       for k, v in generatefeatue.items():
           fout.write(str(k) + '\t' + str(indexmap[v]) + ':1\n')
       fout.close()
       fin1.close()
39          fin2.close()

if __name__ == '__main__':
    path = 'u/'
44    list_dirs = os.walk(path)
    files = []
    for root, dirs, f in list_dirs:
        files += f
    print len(files)
49    cnt = 0
    for i, f in enumerate(files):
        for j, g in enumerate(files):
            if i < j:
                cnt += 1
54          cross(path + f, path + g, f+g)
    print cnt

```

5. Qiang

- *NNTrain.cpp*

Training NNBlending Model.

```

#include "../NNBlending/floatfann.h"
#include <iostream>
#include <fstream>
#include <sstream>
5  #include <stdlib.h>
using namespace std;

```

```

// NNTrain train.data predict.data result.csv save.net input_nodes_num learn_rate max_epochs layers_num
// hidden_layer1_node_num ...
int main(int argc, char* argv[])
10 {
    if (argc < 10)
    {
        cout << "Usage:" << endl;
        cout << "-----" << endl;
15        cout << argv[0] << " _train.data_predict.data_result.csv_save.net_input_nodes_num_layers_num_hidden_layer_node_num
            _..." << endl;
        return -1;
    }
    const unsigned int num_input = atoi(argv[5]);
    const unsigned int num_layers = atoi(argv[8]);
20    const float learn_rate = atof(argv[6]);
    const unsigned int num_output = 1;
    unsigned int* num_nodes_layers = (unsigned int*)malloc(num_layers * sizeof(unsigned int)); //
    const float desired_error = (const float) 0.001;
    const unsigned int max_epochs = atoi(argv[7]);
25    const unsigned int epochs_between_reports = 100;

    char* train_file = argv[1];
    char* predict_file = argv[2];
    char* result_file = argv[3];
30    char* network_file = argv[4];

    num_nodes_layers[0] = num_input;
    for (int i = 1; i < num_layers-1; i++)
    {
35        num_nodes_layers[i] = atoi(argv[8+i]);
    }
    num_nodes_layers[num_layers-1] = 1;

    struct fann *ann = fann_create_standard_array(num_layers, num_nodes_layers);
40

    fann_set_activation_function_hidden (ann, FANN_SIGMOID_SYMMETRIC);
    fann_set_activation_function_output (ann, FANN_SIGMOID_SYMMETRIC);

    fann_set_learning_rate (ann, learn_rate);
45

    /* fann_set_bit_fail_limit (ann, 5.0); */
    cout << "bit_fail_limit:_" << fann_get_bit_fail_limit (ann) << endl;
    cout << "training_algorithm:_" << fann_get_training_algorithm(ann) << endl;
    cout << "learning_rate:_" << fann_get_learning_rate(ann) << endl;
50

    fann_train_on_file (ann, train_file , max_epochs, epochs_between_reports, desired_error);

    fann_type *calc_out;
    fann_type *input = (fann_type*)malloc(num_input * sizeof(fann_type));
55    ifstream ifsPreFile;
    ifsPreFile.open( predict_file );
    ofstream ofsRetFile;
    ofsRetFile.open( result_file );
    float fScore = 0.0;
60    string uid("");
    string bid("");
    if ( ifsPreFile.is_open() )
    {
65        ofsRetFile << "review_id,stars" << endl;
        while ( ifsPreFile.good() )
        {
            ifsPreFile >> uid;

70            ofsRetFile << uid;
            ofsRetFile << ",";

            for (int i = 0; i < num_input; i++)

```

```

75         {
            ifsPreFile >> fScore;
            input[i] = fScore;
        }
        calc_out = fann_run(ann, input);
        ofsRetFile << calc_out[0]*4.0+1.0 << endl;
80     }
    }
    else
    {
        cout << "Fail_to_open_the_predict_file!" << endl;
85         return -1;
    }

    fann_save(ann, network_file);
    fann_destroy(ann);
90    free(input);
    free(num_nodes_layers);

    return 0;
}

```

- *MergeNNBlendingResult.py*
Merge NNBlending Model Result.

```

1  # -*- coding: utf-8 -*-

from time import time
from glob import glob
import os
6  import cPickle
import pylab as pl
import numpy as np
from math import sqrt
import json
11 from collections import defaultdict

def outputFinalPredict(pre, outfile):
    schema = []
    for line in file(final_predict_shcema):
        parts = line.strip().split(',')
16        schema.append([parts[0], parts[1]])
    f = file(outfile, 'w')
    f.write('user_id,business_id,stars\n')
    for i, parts in enumerate(schema):
        parts.append(str(pre[i]))
21    f.write(','.join(parts) + '\n')
    f.close()

if '__main__' == __name__:
26    begin = time()

    final_predict_shcema = 'predict.schema'

    try:
31        folder_id = open('folder_id').read()
        folder_id = int(folder_id)
    except:
        folder_id = 0

36    folder_id = folder_id + 1
    folder_f = open('folder_id', 'w')
    folder_f.write(str(folder_id))
    folder_f.close()
    os.mkdir('%04d' % folder_id)

41    predict_set = []
    outfiles = glob('*.csv')

```



```

print "\nEnsemble_Outfile_List_(Count:_%d)" % len(outfiles)
for outfile in outfiles :
    print outfile
    predict = []
    for line in file ( outfile ):
        parts = line.strip().split(',')
        if parts[2] != 'stars':
51         predict.append(float(parts[2]))
    predict_set.append(predict)
ensemble_predicts = zip(*predict_set)
final_predict = []
for ep in ensemble_predicts:
56     final_predict.append(np.average(ep))
    final_outfile = '%04d/' % folder_id + '_NNBlending' + '.csv'
    outputFinalPredict(final_predict, final_outfile)
    print '\n\nFinal_ensemble_output_file:', final_outfile

61     print '\n\nTotal_Execution_Time:_%3fs' % (time() - begin)

```

- *Ensemble.py*

Using GradientBoostingRegressor, LinearRegression, RandomForestRegressor, Ridge Model to blending our output models (learning by 8 separated local set).

```
# -*- coding: utf-8 -*-
```

```

from time import time
4 from glob import glob
import os
import cPickle
import pylab as pl
import numpy as np
9 from math import sqrt
import json
from collections import defaultdict
from sklearn.linear_model import LinearRegression, Ridge, LogisticRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
14

def getLocalTestData():
    if (os.path.exists ( local_testing_obj )):
        target = cPickle.load( file ( local_testing_obj ))
19    else :
        target = []
        for line in file ( local_testing ):
            js = json.loads( line )
            target.append(js['stars'])
24    cPickle.dump(target, file ( local_testing_obj , 'wb'))
    return target

def getEnsembleTrainData():
    local_predict_files = glob(local_predict_path)
29    print "\nLocal_Predict_File_List_(Count:_%d)" % len(local_predict_files)
    for v in local_predict_files :
        print v
        predict_set = []
        for f in local_predict_files :
34            predict = []
            for line in file (f):
                parts = line.strip().split(',')
                if parts[2] != 'stars':
                    predict.append(float(parts[2]))
39            predict_set.append(predict)
        local_predict = zip(*predict_set)
    return local_predict, predict_set

def getEnsemblePredictData():
44    online_predict_files = glob(online_predict_path)
    print "\nOnline_Predict_File_List_(Count:_%d)" % len(online_predict_files)

```

```

for v in online_predict_files :
    print v
predict_set = []
49 for f in online_predict_files :
    predict = []
    for line in file(f):
        parts = line.strip().split(',')
        if parts[2] != 'stars':
54         predict.append(float(parts[2]))
    predict_set.append(predict)
online_predict = zip(*predict_set)
return online_predict

59 def outputFinalPredict(pre, outfile):
    schema = []
    for line in file(final_predict_shcema):
        parts = line.strip().split(',')
        schema.append([parts[0], parts[1]])
64 f = file(outfile, 'w')
f.write('user_id,business_id,stars\n')
for i, parts in enumerate(schema):
    parts.append(str(pre[i]))
    f.write(','.join(parts) + '\n')
69 f.close()

if '__main__' == __name__:
    begin = time()

74 final_predict_shcema = 'predict.schema'

#Params of GBRT
learning_rate = 0.07
n_estimators = 55 #60
79 max_depth = 2#2
min_samples_split = 5#5
params = []
params.append(str(learning_rate))
params.append(str(n_estimators))
84 params.append(str(max_depth))
params.append(str(min_samples_split))

try:
    folder_id = open('folder_id').read()
89 folder_id = int(folder_id)
except:
    folder_id = '0'

folder_id = folder_id + 1
94 folder_f = open('folder_id', 'w')
folder_f.write(str(folder_id))
folder_f.close()
os.mkdir('%04d' % folder_id)

99 # 0 for gbrt, 1 for lr, 2 for RandomForestRegressor, 3 for Ridge
# blending_choise = raw_input('0 for gbrt, 1 for lr, 2 for RandomForestRegressor, 3 for Ridge\n:')
blending_choise = 0
blending_choise = int(blending_choise)

104 for GROUP in range(1, 8):
    print '\n' + '='*20 + '_GROUP_%d_' % GROUP + '='*20,

    local_testing = '..\Data\ local_final_test_set_review_ %01d.json' % GROUP
    local_testing_obj = '..\Local\%02d\local_testing_%02d' % (GROUP, GROUP)
    local_predict_path = '..\Local\%02d\*local*.csv' % GROUP
    online_predict_path = '..\Online\*csv'

    target = getLocalTestData()
    local_predict, unzip_predict = getEnsembleTrainData()

```

```

114 online_predict = getEnsemblePredictData()
print '\nReading_data_done_in_%.3fs' % (time() - begin)
t = time()

119 if 0 == blending_choise:
    gbrt = GradientBoostingRegressor(learning_rate=learning_rate, n_estimators=n_estimators, max_depth=max_depth,
                                     min_samples_split=min_samples_split)
    gbrt.fit(local_predict, target)
    print '\nGBRT_regression_done_in_%.3fs' % (time() - t)
    t = time()
124    blending paras = 'GBRT:\n'
    blending paras += '\tScore:' + str(gbrt.score(local_predict, target))
    blending paras += '\n'
    blending paras += '\tparams:' + str(gbrt.get_params())
    blending paras += '\n'
129    ensemble_predict = gbrt.predict(online_predict)
    blending_method = 'GBRT'
elif 1 == blending_choise:
    linr = LinearRegression()
    linr.fit(local_predict, target)
134    print '\nLinear_regression_done_in_%.3fs' % (time() - t)
    t = time()
    blending paras = 'LinearRegression:\n'
    blending paras += '\tScore:' + str(linr.score(local_predict, target))
    blending paras += '\n'
139    blending paras += '\tcoef:.' + str(linr.coef_)
    blending paras += '\n'
    ensemble_predict = linr.predict(online_predict)
    blending_method = 'LinearRegression'
elif 2 == blending_choise:
144    rf = RandomForestRegressor()
    rf.fit(local_predict, target)
    print '\nRandom_forest_regression_done_in_%.3fs' % (time() - t)
    t = time()
    blending paras = 'RandomForestRegressor:\n'
149    blending paras += '\tScore:' + str(rf.score(local_predict, target))
    blending paras += '\n'
    ensemble_predict = rf.predict(online_predict)
    blending_method = 'RandomForestRegressor'
elif 3 == blending_choise:
154    ridgr = Ridge(alpha=0.1)
    ridgr.fit(local_predict, target)
    print '\nRidge_regression_done_in_%.3fs' % (time() - t)
    t = time()
    blending paras = 'Ridge(alpha=0.1):\n'
159    blending paras += '\tScore:' + str(ridgr.score(local_predict, target))
    blending paras += '\n'
    blending paras += '\tcoef:.' + str(ridgr.coef_)
    blending paras += '\n'
    ensemble_predict = ridgr.predict(online_predict)
164    blending_method = 'Ridge'

    outfile = '%04d/' % folder_id + '%02d_' % GROUP + '_EnsemblePredict_' + '_'.join(params) + '.csv'
    outputFinalPredict(ensemble_predict, outfile)
    print '\nPredicting_done_in_%.3fs' % (time() - t)
169    print '\nEnsemble_output_file:', outfile
    t = time()

predict_set = []
outfiles = glob('%04d/*.csv' % folder_id)
174 print "\nEnsemble_Outfile_List_(Count:_%d)" % len(outfiles)
for outfile in outfiles:
    print outfile
    predict = []
    for line in file(outfile):
179        parts = line.strip().split(',')
        if parts[2] != 'stars':

```

```

        predict.append(float(parts[2]))
        predict_set.append(predict)
    ensemble_predicts = zip(*predict_set)
184    final_predict = []
    for ep in ensemble_predicts:
        final_predict.append(np.average(ep))
        final_outfile = '%04d/' % folder_id + '_FINAL_' + '_' .join(params) + '.csv'
    outputFinalPredict(final_predict, final_outfile)
189    print '\n\nFinal_ensemble_output_file:', final_outfile

    print '\n\nFinal_ensemble_done_in_%.3fs' % (time() - t)

194    print '\n\nTotal_Execution_Time:_.3fs' % (time() - begin)

```

- *BusinessNameAnalysis.py*

Generate BusinessNameAnalysis feature.

```

1  # -*- coding: cp936 -*-
    import re
    from time import time
    import json

6  business_name = {}
    business_name_src = {}
    bid_map = {}
    attribte_words = ['']

11 def normalizeBusinessName(name_src):
    words = re.split("_|,|&|-|'|/|#|\.", name_src)
    for word in words:
        if 1 == len(word):
            if not word.isupper():
16                words.remove(word)
            elif 'co' == word.lower():
                words.remove(word)
            elif word.isdigit():
                words.remove(word)
21            else:
                new_word = word.strip()
                new_word = new_word.rstrip('s')
                words[words.index(word)] = new_word
    if '_' in words:
26        words.remove('_')
    if '' in words:
        words.remove('')
    if '\t' in words:
        words.remove('\t')
31    norm_name = '_' .join(words).lower()
    return norm_name

def GetBusinessIdMap():
    tb = time()
36    print 'Begin_loading_the_item_id_map_data'
    content = open('./itemmap.final').read()
    lines = content.split('\n')
    for line in lines:
        if not line:
41            continue
        parts = line.split('\t')
        bid_map[parts[0]] = parts[1]
    print 'Finish_loadint_the_item_id_map_data_in_', time() - tb, 's'

46 def GetBusinessNamesFromFile(filename):
    for line in file(filename):
        js = json.loads(line)
        bname = js['name']
        bid = js['business_id']

```

```

51     bname = bname.replace(u'\xe9', u'xe9')
        bname = bname.replace(u'\xe0', u'xe0')
        bname = bname.replace(u'\xea', u'xea')
        bname = bname.replace(u'\xf1', u'xf1')
        bname = bname.replace(u'\xfc', u'xfc')
56     bname = bname.replace(u'\xeb', u'xeb')
        business_name_src[bid] = bname
        bname = normalizeBusinessName(bname)
        if bname in business_name:
            business_name[bname].append(bid)
61     else:
        business_name[bname] = [bid]

def GetBusinessNames():
    tb = time()
66     print 'Begin loading and normalize the business name'
    GetBusinessNamesFromFile('./yelp_training_set/yelp_training_set_business.json')
    GetBusinessNamesFromFile('./final_test_set/final_test_set_business.json')
    print 'Got and normalize the business name in ', time() - tb

71 def SaveNormalizedBusinessName():
    print 'Save normalized business name'
    bnames = business_name.keys()
    print 'business_name_count_after_normalize:', len(bnames)
    bnames.sort()
76     featureFile = open('./test/Qiang_%d.BusinessName.Normalized.Multi.txt' % len(bnames), 'w')
    featureFile.write('%d\n' % len(bnames))
    for bname in bnames:
        name_list = business_name[bname]
        weight = 1.0/float(len(name_list))
81         for bid in name_list:
            featureFile.write('%s' % bid_map[bid])
            for bid in name_list:
                featureFile.write('\t%s:%f' % (bid_map[bid], weight))
            featureFile.write('\n')
86     featureFile.close()
    print 'Save success!'

if __name__ == '__main__':
    GetBusinessIdMap()
91    GetBusinessNames()
    SaveNormalizedBusinessName()

```

- *BusinessNameWordAnalysis.py*

Generate BusinessNameWordAnalysis feature.

```

# -*- coding: cp936 -*-
import re
3 from time import time
import json

business_name = {}
business_name_src = {}
8 bid_map = {}
attribte_words = ['']
bname_dict = []

def normalizeBusinessName(name_src):
13     words = re.split("_|&|'|/|#|\.", name_src)
    for word in words:
        if 1 == len(word):
            if not word.isupper():
                words.remove(word)
18         elif 'co' == word.lower():
            words.remove(word)
        elif word.isdigit():
            words.remove(word)
        else:

```

```

23         new_word = word.strip()
           new_word = new_word.rstrip('s')
           words[words.index(word)] = new_word
if ' ' in words:
    words.remove(' ')
28 if ' ' in words:
    words.remove(' ')
if '\t' in words:
    words.remove('\t')

33 for word in words:
    if word.lower() not in bname_dict:
        bname_dict.append(word.lower())
norm_name = ' '.join(words).lower()
return norm_name

38 def GetBusinessIdMap():
    tb = time()
    print 'Begin loading the item id map data'
    content = open('./itemmap.final').read()
43    lines = content.split('\n')
    for line in lines:
        if not line:
            continue
        parts = line.split('\t')
48    bid_map[parts[0]] = parts[1]
    print 'Finish loading the item id map data in ', time() - tb, 's'

def GetBusinessNamesFromFile(filename):
    for line in file(filename):
53        js = json.loads(line)
        bname = js['name']
        bid = js['business_id']
        bname = bname.replace(u'\xe9', u'xe9')
        bname = bname.replace(u'\xe0', u'xe0')
58        bname = bname.replace(u'\xea', u'xea')
        bname = bname.replace(u'\xf1', u'xf1')
        bname = bname.replace(u'\xfc', u'xfc')
        bname = bname.replace(u'\xeb', u'xeb')
        business_name_src[bid] = bname
63        bname = normalizeBusinessName(bname)
        if bname in business_name:
            business_name[bname].append(bid)
        else:
            business_name[bname] = [bid]

68 def GetBusinessNames():
    tb = time()
    print 'Begin loading and normalize the business name'
    GetBusinessNamesFromFile('./yelp_training_set/yelp_training_set_business.json')
73    GetBusinessNamesFromFile('./final_test_set/final_test_set_business.json')
    print 'Got and normalize the business name in ', time() - tb

def SaveNormalizedBusinessName():
    print 'Save normalized business name'
78    bnames = business_name.keys()
    print 'business name count after normalize:', len(bnames)
    bnames.sort()
    featureFile = open('./test/Qiang-%d.BusinessName.WordIndex.txt' % len(bnames), 'w')
    featureFile.write('%d\n' % len(bnames))
83    for bname in bnames:
        bname_word_id = []
        for word in bname.split(' '):
            index = bname_dict.index(word)
            index = str(index)
88            if index not in bname_word_id:
                bname_word_id.append(index)
        for bid in business_name[bname]:

```

```

        featureFile.write('%s\t%s:1\n' % (bid_map[bid], ':1\t'.join(bname_word_id)))
    featureFile.close()
93     print 'Save_success!'

    if __name__ == '__main__':
        GetBusinessIdMap()
        GetBusinessNames()
98     SaveNormalizedBusinessName()

• BusinessStreetCluste.py
    Generate BusinessStreetCluste feature.

    # -*- coding: cp936 -*-
2
    import json
    import math
    import re

7     business_info = {}
    address_info = {}
    addr_detail_info = []
    cluster_info = {}

12     def GetBusinessAddress():
        print 'Get_business_address_begin...'
        for line in open('./yelp_training_set / yelp_training_set_business .json'):
            js = json.loads(line)
            bid = js['business_id']
            add = js['full_address']
17             # add = add.replace('\n', ' ')
            business_info[bid] = [add]
            if add not in address_info:
                address_info[add] = [bid]
22             else:
                address_info[add].append(bid)

        for line in open('./yelp_test_set / yelp_test_set_business .json'):
            js = json.loads(line)
            bid = js['business_id']
            add = js['full_address']
27             business_info[bid] = [add]
            if add not in address_info:
                address_info[add] = [bid]
            else:
32                 address_info[add].append(bid)

        print 'Got_it!_Business_count:%d,address_count:%d' % (len(business_info), len(address_info))

37     def ClusteBusinessAddress(threshold):
        """
        Cluster
        @shreshold: the similiraty threshold, between 0.0-1.0
        """
42         print 'Cluster_begin ... '
        #TODO: Cluster
        for one in address_info:
            addr = [one, address_info[one]]
            detail = re.split('\n|_|_', one)
47             addr.append(detail)
            addr.append(-1)
            addr_detail_info.append(addr)
        cnt = len(addr_detail_info)
        f = open('./test/cluster.txt', 'w')
        cluster_cnt = 0
52         for i in range(0, cnt):
            addr1 = addr_detail_info[i]
            if addr1[len(addr1)-1] > -1:
                continue

```

```

57     addr1 = addr1[len(addr1)-2]
        cluster_cnt += 1
        f.write('\n-----\n')
        f.write(str( addr_detail_info [ i ] ) + '\n')
        addr_detail_info [ i ][ len( addr_detail_info [ i ] ) - 1 ] = cluster_cnt
62     cluster_info [ cluster_cnt ] = addr_detail_info [ i ][ 0 ]
        for j in range(i+1, cnt):
            addr2 = addr_detail_info [ j ]
            if addr2[len(addr2)-1] > -1:
                continue
67         addr2 = addr2[len(addr2)-2]
            sim = AddressSimilarity(addr1, addr2)
            if sim > threshold:
                addr_detail_info [ j ][ len( addr_detail_info [ j ] ) - 1 ] = cluster_cnt
                f.write(str( addr_detail_info [ j ] ) + '\n')
72     f.close()
        print 'Cluster_finish !_Cluster_count:%d' % (cluster_cnt)

def AddressSimilarity(addr1, addr2):
    same_cnt = 0
77     for one in addr1:
        if one in addr2:
            same_cnt += 1
    return float ((2*same_cnt))/float((len(addr1) + len(addr2)))

82 def GenerateClusterFeature():
    print 'Generate_cluster_feature_begin ... '
    featureFile = open('./test/ cluster_result .txt', 'w')
    featureFile.write(str(len( cluster_info )) + '\n')
    for one in addr_detail_info :
87         cluster_id = one[len(one)-1]
            for id in one[1]:
                try:
                    src_addr = one[0].replace('\n', '\\n')
                    src_addr = src_addr.replace(u'\xed', '\\xed') # fix the bug: '\x' is not asicii code
92                 cluster_addr = cluster_info [ cluster_id ].replace('\n', '\\n')
                    cluster_addr = cluster_addr.replace(u'\xed', '\\xed')
                    featureFile.write(id + '\t' + str( cluster_id ) + '\t' + src_addr + '\t' + cluster_addr + '\n')
                except Exception, e:
                    print e
                    print one
97     featureFile.close()
    print 'Generate_cluster_feature_finish !'

if __name__ == '__main__':
102     GetBusinessAddress()
        ClusteBusinessAddress(0.5)
        GenerateClusterFeature()

```

- *CategoryAnalysis.py*

Generate CategoryAnalysis feature.

```

1  import json

    category_rating = {}
    review_rating = {}

6  # import reveiw rating data
    for line in file ( './ yelp_training_set / yelp_training_set_review .json ' ):
        js = json.loads( line )
        bid = js [ ' business_id ' ]
        rate = js [ ' stars ' ]
11     if bid in review_rating:
        review_rating [ bid ].append(rate)
    else :
        review_rating [ bid ] = [ rate ]

```



```

# import category data
for line in file ('./ yelp_training_set / yelp_training_set_business .json'):
    js = json.loads(line)
    cates = js['categories']
21    bid = js['business_id']
    rate = review_rating[bid]
    for cate in cates:
        if cate in category_rating:
            category_rating[cate] += rate
26    else:
        category_rating[cate] = rate

print 'category_count:', len(category_rating)

31 # save the analysis
import numpy as np
cates = category_rating.keys()
cates_sort_by_std = {}
stds = []
36 cates.sort()
fout = open('./test/CategoryRateAnalysis.txt', 'w')
fout.write('category_count:_%d\n' % len(cates))
for cate in cates:
    rates = category_rating[cate]
41    avg = np.average(rates)
    std = np.std(rates)
    if std in cates_sort_by_std:
        cates_sort_by_std[std].append(cate)
    else:
46        cates_sort_by_std[std] = [cate]
        stds.append(std)
    rates.sort()
    fout.write('-' * 15 + cate + '-' * 15 + '\n')
    fout.write('rate_count:_%d\n' % len(rates))
51    fout.write('average_rate:_%f\n' % avg)
    fout.write('rate_std:_%f\n' % std)
    fout.write(str(rates) + '\n')
fout.close()

56 stds.sort()
fout = open('./test/CategorySortedByStd.txt', 'w')
for std in stds:
    for cate in cates_sort_by_std[std]:
        fout.write('-'*15 + cate + '-'*15 + '\n')
61    fout.write('std:_%f\n' % std)
    rates = category_rating[cate]
    for i in range(1,6):
        fout.write('%d_stars_count:_%d\n' % (i, rates.count(i)))
fout.close()

```

- *CategoryWordIndex.py*

Generate CategoryWordIndex feature.

```

import json
import re
import GetIdMap

5 category_words = []
def GetCategoryWord(category):
    words = []
    for cat in category:
        src_words = re.split ('_|&|', cat)
10        for one in src_words:
            if not one:
                continue
            if one[0] == '(':
                continue
15        new_word = one

```

```

        if one.endswith('ies'):
            new_word = one[:-3] + 'y'
        elif one.endswith('s'):
            new_word = one[:-1]
20     new_word = new_word.lower()
        if new_word not in words:
            words.append(new_word)
        if new_word not in category_words:
            category_words.append(new_word)
25     return words

business_cat = {}
for line in file('./yelp_training_set/yelp_training_set_business.json'):
    js = json.loads(line)
30     bid = js['business_id']
    category = js['categories']
    business_cat[bid] = GetCategoryWord(category)
for line in file('./final_test_set/final_test_set_business.json'):
    js = json.loads(line)
35     bid = js['business_id']
    category = js['categories']
    business_cat[bid] = GetCategoryWord(category)

word_cnt = len(category_words)
40 word_cnt += 1
categoryFile = open('./test/Qiang-%d.Business.CategoryWordIndex.csv' % word_cnt, 'w')
categoryFile.write('%d\n' % word_cnt)
id_map = GetIdMap.GetItemMap()
for bid in business_cat:
45     categoryFile.write('%s\t' % id_map[bid])
    for word in business_cat[bid]:
        categoryFile.write('\t%d:1' % category_words.index(word))
        if 0 == len(business_cat[bid]):
            categoryFile.write('\t%d:1' % (word_cnt-1))
50     categoryFile.write('\n')
categoryFile.close()

```

- *CategorySelectionFeature.py*

Generate CategorySelectionFeature feature.

```

import json

bid_cates = {}
4 cate_var = {}

# import the data
for line in file('./test/CateVar.txt'):
    parts = line.strip().split('\t')
9     if len(parts) < 4:
        continue
    cate = parts[0]
    var = float(parts[3])
    cate_var[cate] = var
14

import GetIdMap
id_map = GetIdMap.GetItemMap()
useful_cates = []

19 threshold = 1.6500

for line in file('./yelp_training_set/yelp_training_set_business.json'):
    js = json.loads(line)
    bid = js['business_id']
24     cates = js['categories']
    bid_map = id_map[bid]
    clean_cate = []
    for cate in cates:
        if cate_var[cate] < threshold:

```

```

29         clean_cate.append(cate)
           if cate not in useful_cates:
               useful_cates.append(cate)
           if len(clean_cate) > 0:
               bid_cates[bid_map] = clean_cate
34
fFile = open('./test/Qiang_%d.CategoryVarUnder%.04f.txt' % (len(useful_cates), threshold), 'w')
fFile.write('%d\n' % len(useful_cates))
for bid in bid_cates:
    fFile.write('%s' % bid)
39     for cate in bid_cates[bid]:
        fFile.write('\t%d:1' % useful_cates.index(cate))
    fFile.write('\n')
fFile.close()

```

- *CheckInAnalysis.py*

Generate CheckInAnalysis feature.

```

import json

3  bid_checkin_info = {}

for line in file('./yelp_training_set/yelp_training_set_checkin.json'):
    js = json.loads(line)
    checkin_info = js['checkin_info']
8    bid = js['business_id']
    if bid not in bid_checkin_info:
        bid_checkin_info[bid] = [0]*4
    for key in checkin_info.keys():
        cnt = checkin_info[key]
13        hour = int(key.split('-')[0])
        bid_checkin_info[bid][hour/6] += cnt

for line in file('./final_test_set/final_test_set_checkin.json'):
    js = json.loads(line)
18    checkin_info = js['checkin_info']
    bid = js['business_id']
    if bid not in bid_checkin_info:
        bid_checkin_info[bid] = [0]*4
    for key in checkin_info.keys():
        cnt = checkin_info[key]
23        hour = int(key.split('-')[0])
        bid_checkin_info[bid][hour/6] += cnt

import GetReviewIdMap
28 import numpy as np
id_map = GetReviewIdMap.GetIdPairByRid()
user_checkin_info = {}
for rid in id_map:
    uid, bid = id_map[rid]
33     if bid not in bid_checkin_info:
        continue
    if uid not in user_checkin_info:
        user_checkin_info[uid] = np.array(bid_checkin_info[bid])
    else:
38        user_checkin_info[uid] += np.array(bid_checkin_info[bid])

ucheckFile = open('./test/UserReviewBusinessCheckInTime.txt', 'w')
mostCheckInTime = []
for uid in user_checkin_info:
    ucheckFile.write('-' * 30 + '\n' + uid + '\n')
43     for cnt in user_checkin_info[uid]:
        ucheckFile.write('%d_' % cnt)
    mostCheckIn = user_checkin_info[uid].max()
    index = user_checkin_info[uid].tolist().index(mostCheckIn)
48     if index not in mostCheckInTime:
        mostCheckInTime.append(index)
    ucheckFile.write('\n')

```

```

ucheckFile.close()

53 import GetIdMap
uid_map = GetIdMap.GetUserMap()
f_cnt = len(mostCheckInTime)
mostCheckInFeature = open('./test/Qiang_%d_UserCheckInMost_4_TimeBin.txt' % f_cnt, 'w')
58 mostCheckInFeature.write('%d\n' % f_cnt)
for uid in user_checkin_info:
    mostCheckIn = user_checkin_info[uid].max()
    index = user_checkin_info[uid].tolist().index(mostCheckIn)
    mostCheckInFeature.write('%s\t%d:1\n' % (uid_map[uid], index))

63 mostCheckInFeature.close()

```

- *CoordinateCluster.py*

Generate CoordinateCluster feature.

```

1 import time as time
import numpy as np
import pylab as pl
import mpl_toolkits.mplot3d.axes3d as p3
from sklearn.cluster import Ward
6 from sklearn.datasets.samples_generator import make_swiss_roll
import json

def GetItemIdMap():
    bid_map = {}
11 for line in open('./itemmap.final'):
    parts = line.split('\t')
    bid_map[parts[0]] = int(parts[1])
    return bid_map

16 def GetCoordinateData():
    print 'Get_business_coordinate_begin...'
    tb = time.time()
    coordinates = []
    business = {}
    bid_map = GetItemIdMap()
    for line in open('./yelp_training_set / yelp_training_set.business.json'):
        js = json.loads(line)
        bid = js['business_id']
26 lon = float(js['longitude'])
        lat = float(js['latitude'])
        coordinates.append([lon, lat])
        id = '%f.%f' % (lon, lat)
        if id in business:
31 business[id].append(bid_map[bid])
        else:
            business[id] = [bid_map[bid]]

    for line in open('./final_test_set / final_test_set.business.json'):
36 js = json.loads(line)
        bid = js['business_id']
        lon = float(js['longitude'])
        lat = float(js['latitude'])
        coordinates.append([lon, lat])
41 id = '%f.%f' % (lon, lat)
        if id in business:
            business[id].append(bid_map[bid])
        else:
            business[id] = [bid_map[bid]]

46 print 'Loading_data_done_in_%s_seconds' % str(time.time()-tb)

return coordinates, business

```

```

51  #
    #####

# Generate data (swiss roll dataset)
coordinates, business = GetCoordinateData()

#
    #####

56 # Compute clustering
    chose = 0
    clusters_num = 50
    if 0 == chose:
        print("Compute_unstructured_hierarchical_clustering...")
61     st = time.time()
        ward = Ward(n_clusters=clusters_num).fit(coordinates)
        label = ward.labels_
        print("Cluster_done!_Elapsed_time:", time.time() - st)
        print("Number_of_points:", label.size)
66     elif 1 == chose:

# Define the structure A of the data. Here a 10 nearest neighbors
    from sklearn.neighbors import kneighbors_graph
    connectivity = kneighbors_graph(coordinates, n_neighbors=10)

71
    print("Compute_structured_hierarchical_clustering...")
    st = time.time()
    ward = Ward(n_clusters=clusters_num, connectivity=connectivity).fit(coordinates)
    label = ward.labels_
76    print("Elapsed_time:", time.time() - st)
    print("Number_of_points:", label.size)

#
    #####

# Generate result
81 lid = 0
    cFile = open('./test/Qiang_%d.Coordinate.HierarCluster_%d.txt' % (clusters_num, chose), 'w')
    cFile.write('%d\n' % clusters_num)
    for point in coordinates:
        pid = '%f%f' % (point[0], point[1])
86     for bid in business[pid]:
        cFile.write('%d\t%d:1\n' % (bid, label[lid]))
        lid += 1
    cFile.close()
    print 'Finish_all!_Elapsed_time:', time.time() - st

• CoordinatesAnalyse.py
Generate CoordinatesAnalyse feature.

# -*- coding: cp936 -*-

import json
4 import math
import re

coordinate = {}
lons = []

9
def GetCoordinates():
    print 'Get_business_coordinate_begin...'
    for line in open('./yelp_training_set / yelp_training_set_business.json'):
        js = json.loads(line)
14         bid = js['business_id']
        lon = float(js['longitude'])
        lat = float(js['latitude'])
        if lon in coordinate:
            coordinate[lon].append({lat:bid})

```

```

19         else:
            coordinate[lon] = [{lat:bid}]
            lons.append(lon)

24     for line in open('./ yelp_test_set / yelp_test_set_business .json'):
        js = json.loads( line )
        bid = js['business_id']
        lon = float( js['longitude'] )
        lat = float( js['latitude'] )
29     if lon in coordinate:
        coordinate[lon].append({lat:bid})
        else:
            coordinate[lon] = [{lat:bid}]
            lons.append(lon)

34

def SaveOrderedCoordinates():
    sFile = open('coordinate_sort.txt', 'w')
    lons.sort()
39    for lon in lons:
        coordinate[lon].sort()
        for pair in coordinate[lon]:
            sFile.write('%%.6f\t%.6f\t%s\n' % (lon, pair.keys()[0], pair.values()[0]))
            sFile.write('\n')
44    sFile.close()

if __name__ == '__main__':
    GetCoordinates()
    SaveOrderedCoordinates()

```

- *GenerateUserRegisterDate.py*

Generate GenerateUserRegisterDate feature.

```

import os
2 import time
import numpy
import json
import math
from scipy.cluster.vq import vq, kmeans, whiten, kmeans2
7 import matplotlib.pyplot as plt
import random

UserCount = 55965
ItemCount = 14334

12 user_time={}
item_time={}

ui_time={}

17 user_f={}
item_f={}

maxtime=0
22 mintime=1e15

for line in open('./ yelp_training_set / yelp_training_set_review .json'):
    js = json.loads( line )
    uid = js['user_id']
27    bid = js['business_id']
    timestamp = math.log(time.mktime(time.strptime(js['date'],'%Y-%m-%d'))),2)

    if maxtime<timestamp:
        maxtime=timestamp
32    if mintime>timestamp:
        mintime=timestamp

```

```

        ui_time[(uid,bid)]=timestamp

37     if uid in user_time:
        user_time[uid].append(timestamp)
    else:
        user_time[uid]=[timestamp]
    if bid in item_time:
42         item_time[bid].append(timestamp)
    else:
        item_time[bid]=[timestamp]

    print 'Read_Data_Over!'

47 for line in open('./ final_test_set / final_test_set_review .json'):
    js = json.loads(line)
    uid = js['user_id']
    bid = js['business_id']

52     if bid in item_time:
        ui_time[(uid,bid)]=min(item_time[bid])
    elif uid in user_time:
        ui_time[(uid,bid)]=min(user_time[uid])
57     else:
        ui_time[(uid,bid)]=random.random()*(maxtime-mintime)+mintime

    print len(ui_time)

62 # generate feature

    from GetIdMap import GetUserMap
    user_map = GetUserMap()
    size = 100
67 fout = open('./test/Qiang_%d-UserFirstReviewTimeBin.txt' % (size + 1), 'w')
    fout.write('%d\n'%(size+1))
    for uid in user_map:
        if uid in user_time:
            u_time = min(user_time[uid])
72         cid = int(float(u_time-mintime-1e-8)/(maxtime-mintime)*size)
            fout.write('%s\t%d:1\n' % (user_map[uid], cid))
        else:
            fout.write('%s\t%d:1\n' % (user_map[uid], size))
    fout.close()
77 exit(0)

```

- *GenerateWeekdayOrWeekendDayCheckin.py*
Generate GenerateWeekdayOrWeekendDayCheckin feature.

```

import json
from GetIdMap import GetItemMap

3
def GetCheckInInfo(checkin_file):
    for line in file(checkin_file):
        js = json.loads(line)
        bid = js['business_id']
8        check_in = js['checkin_info']
        weekend_cnt = 0
        weekday_cnt = 0
        for keys in check_in.keys():
            parts = keys.split('-')
13            if 1 < int(parts[1]):
                weekday_cnt += check_in[keys]
            else:
                weekend_cnt += check_in[keys]
            check_in.info[bid] = 1 if weekday_cnt > weekend_cnt else 0

18
check_in_info = {}
item_map = GetItemMap()
GetCheckInInfo('./yelp_training_set / yelp_training_set_checkin .json')

```

```

GetCheckInInfo('./final_test_set / final_test_set_checkin .json')
23 fFile = open('./test/Qiang_3_CheckInWeekdayOrWeekendDay.txt', 'w')
fFile.write('3\n')
for bid in item_map:
    if bid in check_in_info:
28 fFile.write('%s\t%d:1\n' % (item_map[bid], check_in_info[bid]))
    else:
        fFile.write('%s\t2:1\n' % item_map[bid])
fFile.close()

33 fFile = open('./test/Qiang_2_HaveCheckInOrNot.txt', 'w')
fFile.write('2\n')
for bid in item_map:
    if bid in check_in_info:
        fFile.write('%s\t1:1\n' % (item_map[bid]))
38 else:
        fFile.write('%s\t0:1\n' % item_map[bid])
fFile.close()

```

- *Street.py*

Generate Street feature.

```
# -*- coding: cp936 -*-
```

```

import json
import math
5 import re

business_info = {}
address_info = {}
addr_detail_info = []
10 cluster_info = {}
item_map = {}
streetFlags = ['rd', 'road', 'ave', 'av', 'avenue', 'blvd', 'boulevard', 'pkwy', 'parkway', 'dr', 'drive', 'st', 'street', '
way', 'fwy', 'freeway', 'ln', 'la', 'lane', 'ct', 'court', 'sq', 'square', 'cir', 'circle']
streetAddr = {'rd': ['rd', 'road'],
               'ave': ['ave', 'av', 'avenue'],
15 'blvd': ['blvd', 'boulevard'],
               'pkwy': ['pkwy', 'parkway'],
               'dr': ['dr', 'drive'],
               'st': ['st', 'street'],
               'way': ['way'],
20 'fwy': ['fwy', 'freeway'],
               'ln': ['ln', 'la', 'lane'],
               'ct': ['ct', 'cour'],
               'sq': ['sq', 'square'],
               'cir': ['cir', 'circle']}

25 def CleanStreet(street):
    if '#' in street:
        street = street[:street.index('#')]
    addrs = re.split(r'_|\\.|', street)
30 try:
        int(addrs[0])
        addrs.remove(addrs[0])
    except:
        print 'No_street_number'

35 if 'n' in addrs:
    addrs.remove('n')
elif 's' in addrs:
    addrs.remove('s')
40 elif 'e' in addrs:
    addrs.remove('e')
elif 'w' in addrs:
    addrs.remove('w')
elif 'north' in addrs:

```



```

45     addrs.remove('north')
elif 'south' in addrs:
    addrs.remove('south')
elif 'east' in addrs:
    addrs.remove('east')
50 elif 'west' in addrs:
    addrs.remove('west')
elif 'nw' in addrs:
    addrs.remove('nw')
elif 'nw' in addrs:
    addrs.remove('nw')
55 if '' in addrs:
    addrs.remove('')
if '_' in addrs:
    addrs.remove('_')

60 newAddr = []
for addr in addrs:
    if addr.lower() in streetFlags:
        newAddr.append(addr)
65     break
    newAddr.append(addr)
addrs = newAddr

for addr in streetAddr:
70     if addrs[len(addrs)-1].lower() in streetAddr[addr]:
        addrs[len(addrs)-1] = addr

return '_'.join(addrs)

75 def GetStreetInfo( full_address ):
    full_address = full_address.replace(u'\xed', u'xed')
    full_address = full_address.lower()
    addrs = full_address.split('\n')
    pat = r'\d+?\s+[nsw]{1,2}\.{0,1}\_+?'
80 for addr in addrs:
    tmp = addr.replace('south', 's')
    tmp = tmp.replace('north', 'n')
    tmp = tmp.replace('west', 'w')
    tmp = tmp.replace('east', 'e')
85 if re.match(pat, tmp, re.S):
    return CleanStreet(addr)
words = re.split('_|\.', addr)
if words[len(words)-1].lower() in streetFlags:
    return CleanStreet(addr)
90 for word in words:
    if word.lower() in streetFlags:
        return CleanStreet(addr)
return ''

95 def GetBusinessAddress():
    print 'Get_business_address_begin...'
    streetFile = open('./test/streets_precise.txt', 'w')
    for line in open('./yelp_training_set/yelp_training_set_business.json'):
100         js = json.loads(line)
        bid = js['business_id']
        add = js['full_address']
        add = add.replace(u'\xed', u'xed')
        business_info[bid] = add.replace('\n', '\\n')
105         # add = add.replace('\n', ' ')
        street = GetStreetInfo(add)
        if street not in address_info:
            address_info[street] = [bid]
        else:
110             address_info[street].append(bid)
    streetFile.write(bid + '\t' + street + '\n')

```

```

115     for line in open('./ final_test_set / final_test_set_business .json'):
        js = json.loads( line )
        bid = js[' business.id ' ]
        add = js[' full_address ' ]
        add = add.replace(u'\xed', u'xed')
        business_info [bid] = add.replace('\n', '\n')
        street = GetStreetInfo(add)
120     if street not in address_info:
        address_info [street] = [bid]
    else:
        address_info [street].append(bid)
        streetFile.write(bid + '\t' + street + '\n')
125     streetFile.close()

def GetItemMap():
    for line in file ("itemmap.final"):
        if not line:
130             continue
        parts = line.strip().split('\t')
        item_map[parts[0]] = parts[1]

def SaveStreetClusterFileAndFeatureFile():
135     streetCFile = open('./test/ streetClusters.precise.txt', 'w')
        streetMissFile = open('./test/businessMissStreet.txt', 'w')
        featureFile = open('./test/Qiang_%d.Street.txt' % len(address_info), 'w')
        featureFile.write('%d\n' % len(address_info))
        cid = 0
140     for one in address_info:
        if '' == one:
            for bid in address_info[one]:
                streetMissFile.write(bid + '\t' + business_info[bid] + '\n')
            for bid in address_info[one]:
145                streetCFile.write(bid + '\t' + str(cid) + '\t' + one + '\n')
                featureFile.write('%s\t%d:1\n' % (item_map[bid], cid))
            cid += 1
        streetCFile.close()
        streetMissFile.close()
150
    print 'Got_it!'

155 if __name__ == '__main__':
    GetBusinessAddress()
    GetItemMap()
    SaveStreetClusterFileAndFeatureFile()

```

- *StreetAnalyse.py*

Generate StreetAnalyse feature.

```

# -*- coding: cp936 -*-
2
import json
import math
import re

7     business_info = {}
        address_info = {}
        full_address = {}
        addr_detail_info = []
        cluster_info = {}
12
def GetStreetInfo( full_address ):
    full_address = full_address.replace(u'\xed', u'xed')
    addrs = full_address.split('\n')
    addr = addrs[0]
17     addrs = addr.split(' ')
    try:

```

```

        int (addrs[0])
        addrs.remove(addrs[0])
    except:
22         print 'No_street_number'
    if 'N' in addrs:
        addrs.remove('N')
    elif 'S' in addrs:
        addrs.remove('S')
27     elif 'E' in addrs:
        addrs.remove('E')
    elif 'W' in addrs:
        addrs.remove('W')
    return ' '.join(addrs)
32
def GetBusinessAddress():
    print 'Get_business_address_begin...'

    for line in open('./yelp_training_set / yelp_training_set_business .json'):
37         js = json.loads( line )
        bid = js['business_id']
        add = js['full_address']
        add = add.replace(u'\xed', u'xed')
        full_address [bid] = add
42
    for line in open('./yelp_test_set / yelp_test_set_business .json'):
        js = json.loads( line )
        bid = js['business_id']
        add = js['full_address']
47         add = add.replace(u'\xed', u'xed')
        full_address [bid] = add

def SaveStreetByEnter():
    sFile = {}
52     for bid in full_address :
        addrs = full_address [bid]. split ( '\n' )
        if str (len (addrs)) not in sFile :
            sFile [str (len (addrs))] = open ( './test/StreetWith' + str (len (addrs)) + 'Enter.txt', 'w' )
            sFile [str (len (addrs))]. write (bid + '\t' + ' '.join (addrs) + '\n')
57     for one in sFile :
        sFile [one]. close ()

if __name__ == '__main__':
    GetBusinessAddress()
62     SaveStreetByEnter()

```

- *StreetDirection.py*

Generate StreetDirection feature.

```
# -*- coding: cp936 -*-
```

```

3  import json
    import math
    import re

    business_info = {}
8  address_info = {}
    addr_detail_info = []
    cluster_info = {}
    item_map = {}
    streetFlags = ['rd', 'road', 'ave', 'av', 'avenue', 'blvd', 'boulevard', 'pkwy', 'parkway', 'dr', 'drive', 'st', 'street', '
        way', 'fwy', 'freeway', 'ln', 'la', 'lane', 'ct', 'court', 'sq', 'square', 'cir', 'circle']
13 streetAddr = {'rd' : ['rd', 'road'],
                'ave' : ['ave', 'av', 'avenue'],
                'blvd' : ['blvd', 'boulevard'],
                'pkwy' : ['pkwy', 'parkway'],
                'dr' : ['dr', 'drive'],
18                'st' : ['st', 'street'],
                'way' : ['way'],

```

```

        'fwy' : ['fwy', 'freeway'],
        'ln' : ['ln', 'la', 'lane'],
        'ct' : ['ct', 'cour'],
23      'sq' : ['sq', 'square'],
        'cir' : ['cir', 'circle']]

def CleanStreet(street):
    if '#' in street:
28      street = street[:street.index('#')]
    addrs = re.split(r'_|\\.|', street)
    try:
        int(addrs[0])
        addrs.remove(addrs[0])
33    except:
        print 'No_street_number'

    if 'n' in addrs:
        return 'n'
38    elif 's' in addrs:
        return 's'
    elif 'e' in addrs:
        return 'e'
    elif 'w' in addrs:
43    return 'w'
    elif 'north' in addrs:
        return 'n'
    elif 'south' in addrs:
        return 's'
48    elif 'east' in addrs:
        return 'e'
    elif 'west' in addrs:
        return 'w'
    elif 'nw' in addrs:
53    return 'nw'
    elif 'ne' in addrs:
        return 'ne'
    elif 'sw' in addrs:
        return 'sw'
58    elif 'se' in addrs:
        return 'se'
    else:
        return ''

63
def GetStreetInfo(full_address):
    full_address = full_address.replace(u'\xed', u'xed')
    full_address = full_address.lower()
    addrs = full_address.split('\n')
68    pat = r'\d+?\s+?[nswe]{1,2}\.{0,1}_\.+?'
    for addr in addrs:
        tmp = addr.replace('south', 's')
        tmp = tmp.replace('north', 'n')
        tmp = tmp.replace('west', 'w')
73      tmp = tmp.replace('east', 'e')
        if re.match(pat, tmp, re.S):
            return CleanStreet(addr)
        words = re.split('_|\\.|', addr)
        if words[len(words)-1].lower() in streetFlags:
78      return CleanStreet(addr)
        for word in words:
            if word.lower() in streetFlags:
                return CleanStreet(addr)
    return ''

83
def GetBusinessAddress():
    print 'Get_business_address_begin...'
    streetFile = open('./test/streets_precise.txt', 'w')

```

```

88     for line in open('./ yelp_training_set / yelp_training_set_business .json'):
        js = json.loads( line )
        bid = js['business.id']
        add = js['full_address']
        add = add.replace(u'\xed', u'xed')
93     business_info[bid] = add.replace('\n', '\\n')

        street = GetStreetInfo(add)
        if street not in address_info:
            address_info[street] = [bid]
98     else:
        address_info[street].append(bid)
        streetFile.write(bid + '\t' + street + '\n')

    for line in open('./ final_test_set / final_test_set_business .json'):
103     js = json.loads( line )
        bid = js['business.id']
        add = js['full_address']
        add = add.replace(u'\xed', u'xed')
        business_info[bid] = add.replace('\n', '\\n')

108     street = GetStreetInfo(add)
        if street not in address_info:
            address_info[street] = [bid]
        else:
            address_info[street].append(bid)
113     streetFile.write(bid + '\t' + street + '\n')
    streetFile.close()

def GetItemMap():
118     for line in file("itemmap.final"):
        if not line:
            continue
        parts = line.strip().split('\t')
        item_map[parts[0]] = parts[1]
123

def SaveStreetClusterFileAndFeatureFile():
    featureFile = open('./test/Qiang_%d.StreetDirection.txt' % len(address_info), 'w')
    featureFile.write('%d\n' % len(address_info))
    cid = 0
128     for one in address_info:
        for bid in address_info[one]:
            featureFile.write('%s\t%d\n' % (item_map[bid], cid))
            cid += 1
        featureFile.close()
133

    print 'Got_it!'

138 if __name__ == '__main__':
    GetBusinessAddress()
    GetItemMap()
    SaveStreetClusterFileAndFeatureFile()

```

- *StreetWithDirection.py*

Generate StreetWithDirection feature.

```
# -*- coding: cp936 -*-
```

```

3 import json
import math
import re

business_info = {}
8 address_info = {}
addr_detail_info = []
cluster_info = {}

```

```

item_map = {}
streetFlags = ['rd', 'road', 'ave', 'av', 'avenue', 'blvd', 'boulevard', 'pkwy', 'parkway', 'dr', 'drive', 'st', 'street', '
way', 'fwy', 'freeway', 'ln', 'la', 'lane', 'ct', 'court', 'sq', 'square', 'cir', 'circle']
13 streetAddr = {'rd': ['rd', 'road'],
                'ave': ['ave', 'av', 'avenue'],
                'blvd': ['blvd', 'boulevard'],
                'pkwy': ['pkwy', 'parkway'],
                'dr': ['dr', 'drive'],
18 'st': ['st', 'street'],
                'way': ['way'],
                'fwy': ['fwy', 'freeway'],
                'ln': ['ln', 'la', 'lane'],
                'ct': ['ct', 'cour'],
23 'sq': ['sq', 'square'],
                'cir': ['cir', 'circle']}

def CleanStreet(street):
    if '#' in street:
28 street = street[:street.index('#')]
    addrs = re.split(r'_|\\.|', street)
    try:
        int(addrs[0])
        addrs.remove(addrs[0])
33 except:
        print 'No_street_number'

    if '' in addrs:
        addrs.remove('')
38 if '_' in addrs:
        addrs.remove('_')

    newAddr = []
    for addr in addrs:
43 if addr.lower() in streetFlags:
        newAddr.append(addr)
        break
    newAddr.append(addr)
    addrs = newAddr

48 for addr in streetAddr:
    if addrs[len(addrs)-1].lower() in streetAddr[addr]:
        addrs[len(addrs)-1] = addr

53 return '_'.join(addrs)

def GetStreetInfo(full_address):
    full_address = full_address.replace(u'\xed', u'xed')
    full_address = full_address.lower()
58 addrs = full_address.split('\n')
    pat = r'\d+?\s+?[nswe]{1,2}\.{0,1}_+?'
    for addr in addrs:
        tmp = addr.replace('south', 's')
        tmp = tmp.replace('north', 'n')
63 tmp = tmp.replace('west', 'w')
        tmp = tmp.replace('east', 'e')
        if re.match(pat, tmp, re.S):
            return CleanStreet(addr)
    words = re.split('_|\\.|', addr)
68 if words[len(words)-1].lower() in streetFlags:
        return CleanStreet(addr)
    for word in words:
        if word.lower() in streetFlags:
            return CleanStreet(addr)
73 return ''

def GetBusinessAddress():
    print 'Get_business_address_begin...'

```

```

78     streetFile = open('./test/ streets_precise .txt', 'w')
    for line in open('./ yelp_training_set / yelp_training_set_business .json'):
        js = json.loads( line )
        bid = js['business.id']
        add = js['full_address']
83         add = add.replace(u'\xed', u'xed')
        business_info [bid] = add.replace('\n', '\\n')
        # add = add.replace('\n', ' ')
        street = GetStreetInfo(add)
        if street not in address_info:
88             address_info [street] = [bid]
        else:
            address_info [street].append(bid)
            streetFile .write(bid + '\t' + street + '\n')

93     for line in open('./ final_test_set / final_test_set_business .json'):
        js = json.loads( line )
        bid = js['business.id']
        add = js['full_address']
        add = add.replace(u'\xed', u'xed')
98         business_info [bid] = add.replace('\n', '\\n')
        street = GetStreetInfo(add)
        if street not in address_info:
            address_info [street] = [bid]
        else:
103             address_info [street].append(bid)
            streetFile .write(bid + '\t' + street + '\n')
    streetFile .close()

    def GetItemMap():
108         for line in file ("itemmap.final"):
            if not line:
                continue
            parts = line.strip().split ('\t')
            item_map[parts[0]] = parts[1]

113     def SaveStreetClusterFileAndFeatureFile():
        featureFile = open('./test/Qiang_%d.StreetWithDirection.txt' % len(address_info), 'w')
        featureFile .write('%d\n' % len(address_info))
        cid = 0
118         for one in address_info:
            for bid in address_info [one]:
                featureFile .write('%s\t%d:1\n' % (item_map[bid], cid))
            cid += 1
        featureFile .close()

123     print 'Got_it!'

128     if __name__ == '__main__':
        GetBusinessAddress()
        GetItemMap()
        SaveStreetClusterFileAndFeatureFile()

```

- *UserBusinessName.py*

Generate UserBusinessName feature.

```

import GetReviewIdMap
import GetIdMap

3     bid_sname_map = {}

    for line in file ('./ test /Qiang_9998_BusinessName_Normalized.txt'):
        parts = line.strip().split ('\t')
8         if len(parts) < 2:
            continue
        sname = parts[1].split (':')[0]

```

```

        bid_sname_map[parts[0]] = sname

13 id_map = GetReviewIdMap.GetIdPairByRid()
    uid_map = GetIdMap.GetUserMap()
    bid_map = GetIdMap.GetItemMap()

    uid_sname_map = {}
18 all_sname = []
    for rid in id_map:
        uid, bid = id_map[rid]
        uid = uid_map[uid]
        bid = bid_map[bid]
23 sname = bid_sname_map[bid]
        if sname not in all_sname:
            all_sname.append(sname)
        if uid in uid_sname_map:
            if sname not in uid_sname_map[uid]:
28 uid_sname_map[uid].append(sname)
        else:
            uid_sname_map[uid] = [sname]

    f_cnt = len(all_sname)
33 fFile = open('./test/Qiang_%d_UserBusinessName.txt' % f_cnt, 'w')
    fFile.write('%d\n' % f_cnt)
    for uid in uid_sname_map:
        fFile.write('%s\t%s' % (uid, ':1\t'.join(uid_sname_map[uid])) + ':1\n')
    fFile.close()

```

- *UserBusinessNameTail.py*

Generate UserBusinessNameTail feature.

```

import GetReviewIdMap
2 import GetIdMap
import json

bid_city_map = {}
all_cname = []

7
for line in file('./yelp_training_set / yelp_training_set_business .json'):
    js = json.loads(line)
    bid = js['business_id']
    city = js['name']
12 city = city.lower()
    city = city.split('_')[-1]
    bid_city_map[bid] = city
    if city not in all_cname:
        all_cname.append(city)

17
for line in file('./final_test_set / final_test_set_business .json'):
    js = json.loads(line)
    bid = js['business_id']
    city = js['name']
22 city = city.lower()
    city = city.split('_')[-1]
    bid_city_map[bid] = city
    if city not in all_cname:
        all_cname.append(city)

27
print len(all_cname)
id_map = GetReviewIdMap.GetIdPairByRid()
uid_map = GetIdMap.GetUserMap()
bid_map = GetIdMap.GetItemMap()

32
uid_city_map = {}
for rid in id_map:
    uid, bid = id_map[rid]
    cname = bid_city_map[bid]
37 uid = uid_map[uid]

```



```

        bid = bid_map[bid]
        cname = all_cname.index(cname)
        cname = str(cname)
        if uid in uid_city_map:
42             if cname not in uid_city_map[uid]:
                    uid_city_map[uid].append(cname)
            else:
                    uid_city_map[uid] = [cname]

47 f_cnt = len(all_cname)
fFile = open('./test/Qiang_%d_User_BusinessNameTail.txt' % f_cnt, 'w')
fFile.write('%d\n' % f_cnt)
for uid in uid_city_map:
    fFile.write('%s\t%s' % (uid, ':1\t'.join(uid_city_map[uid])) + ':1\n')
52 fFile.close()

```

- *UserCategoryHistory.py*

Generate UserCategoryHistory feature.

```

import json

2 user_cate_his = {}
item_cates = {}
useful_cates = []

7 # import data
for line in file('./yelp_training_set/yelp_training_set_business.json'):
    js = json.loads(line)
    item_cates[js['business_id']] = js['categories']

12 for line in file('./yelp_training_set/yelp_training_set_review.json'):
    js = json.loads(line)
    uid = js['user_id']
    bid = js['business_id']
    if uid not in user_cate_his:
17         user_cate_his[uid] = []
    for cate in item_cates[bid]:
        if cate not in user_cate_his[uid]:
            user_cate_his[uid].append(cate)
            if cate not in useful_cates:
22                 useful_cates.append(cate)

import GetIdMap
id_map = GetIdMap.GetUserMap()
fFile = open('./test/Qiang_%d_UserCategoryHistory.txt' % len(useful_cates), 'w')
27 fFile.write('%d\n' % len(useful_cates))
for uid in user_cate_his:
    if len(user_cate_his[uid]) == 0:
        continue
    fFile.write(id_map[uid])
32 for cate in user_cate_his[uid]:
    fFile.write('\t%d:1' % useful_cates.index(cate))
    fFile.write('\n')

fFile.close()

```

- *UserCityFeature.py*

Generate UserCityFeature feature.

```

import GetReviewIdMap
2 import GetIdMap
import json

bid_city_map = {}
all_cname = []

7 for line in file('./yelp_training_set/yelp_training_set_business.json'):
    js = json.loads(line)
    bid = js['business_id']

```

```

    cite = js['city']
12    cite = cite.lower()
    bid_city_map[bid] = cite
    if cite not in all_cname:
        all_cname.append(cite)

17 for line in file('./final_test_set / final_test_set_business .json'):
    js = json.loads(line)
    bid = js['business_id']
    cite = js['city']
    cite = cite.lower()
22    bid_city_map[bid] = cite
    if cite not in all_cname:
        all_cname.append(cite)

id_map = GetReviewIdMap.GetIdPairByRid()
27 uid_map = GetIdMap.GetUserMap()
    bid_map = GetIdMap.GetItemMap()

uid_city_map = {}
for rid in id_map:
32     uid, bid = id_map[rid]
    cname = bid_city_map[bid]
    uid = uid_map[uid]
    bid = bid_map[bid]
    cname = all_cname.index(cname)
37    cname = str(cname)
    if uid in uid_city_map:
        if cname not in uid_city_map[uid]:
            uid_city_map[uid].append(cname)
    else:
42     uid_city_map[uid] = [cname]

f_cnt = len(all_cname)
fFile = open('./test/Qiang_%d_UserCity.txt' % f_cnt, 'w')
fFile.write('%d\n' % f_cnt)
47 for uid in uid_city_map:
    fFile.write('%s\t%s' % (uid, ':1\t'.join(uid_city_map[uid])) + ':1\n')
fFile.close()

```

- *UserStreetName.py*

Generate UserStreetName feature.

```

import GetReviewIdMap
import GetIdMap

bid_sname_map = {}
5
for line in file('./test/Qiang_1264_Street.txt'):
    parts = line.strip().split('\t')
    if len(parts) < 2:
        continue
10    sname = parts[1].split(':')[0]
    bid_sname_map[parts[0]] = sname

id_map = GetReviewIdMap.GetIdPairByRid()
uid_map = GetIdMap.GetUserMap()
15 bid_map = GetIdMap.GetItemMap()

uid_sname_map = {}
all_sname = []
for rid in id_map:
20     uid, bid = id_map[rid]
    uid = uid_map[uid]
    bid = bid_map[bid]
    sname = bid_sname_map[bid]
    if '0' == sname:
25         continue

```

```

sname = int(sname)
sname -= 1
sname = str(sname)
if sname not in all_sname:
    all_sname.append(sname)
30 if uid in uid_sname_map:
    if sname not in uid_sname_map[uid]:
        uid_sname_map[uid].append(sname)
    else:
35     uid_sname_map[uid] = [sname]

f_cnt = len(all_sname)
fFile = open('./test/Qiang-%d_UserStreetName.txt' % f_cnt, 'w')
fFile.write('%d\n' % f_cnt)
40 for uid in uid_sname_map:
    fFile.write('%s\t%s' % (uid, ':1\t'.join(uid_sname_map[uid])) + ':1\n')
fFile.close()

```

- *Zipcode.py*

Generate Zipcode feature.

```

import numpy
2 import json
import math
import re

business_info = {}
7 address_info = {}
addr_detail_info = []
item_map = {}

def GetBusinessAddress():
12     print 'Get_business_address_begin...'
    for line in open('./ yelp_training_set / yelp_training_set.business.json'):
        js = json.loads(line)
        bid = js['business_id']
        add = js['full_address']
17         # add = add.replace('\n', ' ')
        business_info[bid] = [add]
        if add not in address_info:
            address_info[add] = [bid]
        else:
22         address_info[add].append(bid)

    for line in open('./ final_test_set / final_test_set.business.json'):
        js = json.loads(line)
        bid = js['business_id']
        add = js['full_address']
27         # add = add.replace('\n', ' ')
        business_info[bid] = [add]
        if add not in address_info:
            address_info[add] = [bid]
        else:
32         address_info[add].append(bid)

    print 'Got it!_Business_count:%d,_address_count:%d' % (len(business_info), len(address_info))

37 def ClusteBusinessAddress():
    print 'Cluster_begin...'
    #TODO: Cluster
    for one in address_info:
        addr = [one, address_info[one]]
42         detail = re.split('\n|,', one)
        addr.append(detail)
        addr_detail_info.append(addr)
    print 'Cluster_finish!'

47 def GetItemMap():

```

```

for line in file("itemmap.final"):
    if not line:
        continue
    parts = line.strip().split('\t')
52     item_map[parts[0]] = parts[1]

def GenerateClusterFeature():
    print 'Generate_cluster_feature_begin ...'
    zipcodes = {}
57     # TODO: Generate
    wrongs = []
    stat = {}
    zipcodeset = {}
    maxId = 1
62     for one in addr_detail_info:
        pat = r'\w+?(\d+)'
        cnt = len(one[len(one)-1])
        if cnt not in stat:
            stat[cnt] = 1
67         else:
            stat[cnt] += 1
        ret = re.search(pat, one[len(one)-1][len(one[len(one)-1])-1])
        if not ret:
            wrongs.append(one)
72         else:
            zipcode = str(ret.group(1))
            if zipcode in zipcodeset:
                zipcodeid = zipcodeset[zipcode]
            else:
77                 zipcodeid = maxId
                zipcodeset[zipcode] = maxId
                maxId += 1
            for id in one[1]:
                zipcodes[id] = zipcodeid
82     featureFile = open('./test/Qiang_%d.zipcode.txt' % maxId, 'w')
    featureFile.write('%d\n' % maxId)
    for bid in zipcodes:
        featureFile.write('%s\t%d:1\n' % (item_map[bid], zipcodes[bid]))
    for one in wrongs:
87         for bid in one[1]:
            featureFile.write('%s\t0:1\n' % (item_map[bid]))
    featureFile.close()

    print 'zip_code_count:%d' % maxId
92
    print 'Generate_cluster_feature_finish!'

if __name__ == '__main__':
    GetBusinessAddress()
97    GetItemMap()
    ClusteBusinessAddress()
    GenerateClusterFeature()

```