W&D_Dataset_preparation_larger

December 19, 2019

```
In [0]: import pandas as pd
        import numpy as np
        import random
        import matplotlib.pyplot as plt
        import time
        import sklearn
        from sklearn.model_selection import train_test_split
        from random import shuffle
        import seaborn as sns
```

```
In [2]: from google.colab import drive
        drive.mount('/content/drive',force_remount=True)
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6

Enter your authorization code:
ůůůůůůůůůů
Mounted at /content/drive
```

```
In [0]: path="/content/drive/My Drive/yelp_final_data/"
```

0.0.1 This notebook is for preparing train and test set for Wide and Deep model.

There are two parts: 1. Prepare the large train and test set using the test index we prepared before. 2. Segment the test set into different levels of users and business.

Read the dataset we prepared before. We only want the restaurant that has been rated more than twice and the users that rated at least 5 times.

Read the test index we saved before, so that we split the data into train and test set.

```
In [4]: #start_time=time.time()
        review=pd.read_csv(path+'review.csv')
        del review['text_review']
        review['freq_business'] = review.groupby('business_id')['business_id'].transform('count
        review2=review.loc[review['freq_business']>2]
        review2['freq_user'] = review2.groupby('user_id')['user_id'].transform('count')
        review3=review2.loc[review2['freq_user']>=5]
        review3=review3.reset_index()
```

```
        test_idx=pd.read_csv(path+'all_test_idx_df2.csv')
        test_idx=test_idx.rename({'0':'index'},axis=1)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/in
  """
```

Using the test row index we saved before to get train and test set.

```
In [0]: test=review3.loc[review3['index'].isin(test_idx['index'])]
        train=review3.loc[~review3['index'].isin(test_idx['index'])]
```

```
In [0]: train.head(3)
```

```
Out[0]:    index                 user_id  ... freq_business  freq_user
        0      0  hG7b0MtEbXx5QzbzE6C_VA  ...           183         10
        1      1  n6-Gk65cPZL6Uz8qRm3NYw  ...            20          9
        2      2  jlu4CztcSxrKx56ba1a5AQ  ...           108        336

        [3 rows x 10 columns]
```

Read the user and business dataset and join the table on the user_id and business_id.

```
In [0]: user=pd.read_csv(path+'user2.csv')
        business=pd.read_csv(path+'business.csv')
        business=business.rename(columns={"business_ids": "business_id"})
```

```
In [0]: train = pd.merge(train, user, on='user_id')
        test=pd.merge(test, user, on='user_id')
```

```
In [0]: train = pd.merge(train, business, on='business_id')
        test = pd.merge(test, business, on='business_id')
```

```
In [0]: train.head(2)
```

```
Out[0]:     index  ...                                                hours
        0       0  ...  {'Monday': '0:0-0:0', 'Tuesday': '0:0-0:0', 'W...
        1  186281  ...  {'Monday': '0:0-0:0', 'Tuesday': '0:0-0:0', 'W...

        [2 rows x 38 columns]
```

Select the features we want.

```
In [0]: train1=train[['index','user_id','business_id','city','state',"average_stars",'useful_re
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"r
        test1=test[['index','user_id','business_id','city','state',"average_stars",'useful_revi
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"r
```

```
In [0]: train1.head(2)
```

```
Out[0]:     index                 user_id  ... compliment_funny rating_review
        0       0  hG7b0MtEbXx5QzbzE6C_VA  ...                 0           1.0
        1  186281  hG7b0MtEbXx5QzbzE6C_VA  ...                 0           1.0

        [2 rows x 19 columns]
```

```
In [0]: # user_id_addresses = train1.user_id.unique()
        # user_id_dict = dict(zip(user_id_addresses, range(len(user_id_addresses))))
        # train1=train1.applymap(lambda s: user_id_dict.get(s) if s in user_id_dict else s)
        # test1=test1.applymap(lambda s: user_id_dict.get(s) if s in user_id_dict else s)
```

```
In [0]: # total_business_id=list(train1.business_id.unique())+list(test1.business_id.unique())
```

```
In [0]: # business_id_dict = dict(zip(total_business_id, range(len(total_business_id))))
```

```
In [0]: # train1=train1.applymap(lambda s: business_id_dict.get(s) if s in business_id_dict el
        # test1=test1.applymap(lambda s: business_id_dict.get(s) if s in business_id_dict else
```

```
In [0]: # train1.head(2)
```

```
Out[0]:    user_id  business_id       city  ... fans  compliment_funny  rating_review
        0        0       151026  Las Vegas  ...    0                 0            1.0
        1        0       151026  Las Vegas  ...    0                 0            1.0

        [2 rows x 18 columns]
```

```
In [0]: del train1['index']
        del test1['index']
```

Check if there are missing values.

```
In [10]: train1.isnull().sum()
```

```
Out[10]: index                0
         user_id              0
         business_id          0
         city                 0
         state                0
         average_stars        0
         useful_review        2
         funny_review         2
         cool_review          2
         compliment_more      0
         compliment_cute      0
         num_friends          0
         stars                0
         useful               0
         funny                0
```

```
cool                  0
fans                  0
compliment_funny      0
categories          554
rating_review         0
dtype: int64
```

Since there are not many missing values, we fill them with zero, and we fill the missing value in the 'categories' column as 'No category'

```
In [11]: train1['useful_review']=train1['useful_review'].fillna(float(0))
         train1['funny_review']=train1['funny_review'].fillna(float(0))
         train1['cool_review']=train1['cool_review'].fillna(float(0))
         train1 = train1.fillna('No_category')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/ir
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/ir

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/ir
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
In [12]: train1.isnull().sum()
```

```
Out[12]: index              0
         user_id            0
         business_id        0
         city               0
         state              0
         average_stars      0
         useful_review      0
         funny_review       0
         cool_review        0
         compliment_more    0
         compliment_cute    0
         num_friends        0
```

```
stars               0
useful              0
funny               0
cool                0
fans                0
compliment_funny    0
categories          0
rating_review       0
dtype: int64
```

Done filling missing value for train set.
Check missing value for test set.

```
In [13]: test1.isnull().sum()
```

```
Out[13]: index               0
         user_id             0
         business_id         0
         city                0
         state               0
         average_stars       0
         useful_review       0
         funny_review        0
         cool_review         0
         compliment_more     0
         compliment_cute     0
         num_friends         0
         stars               0
         useful              0
         funny               0
         cool                0
         fans                0
         compliment_funny    0
         categories        168
         rating_review       0
         dtype: int64
```

```
In [0]: test1 = test1.fillna('No_category')
```

```
In [15]: test1.isnull().sum()
```

```
Out[15]: index               0
         user_id             0
         business_id         0
         city                0
         state               0
         average_stars       0
         useful_review       0
         funny_review        0
```

```
cool_review         0
compliment_more     0
compliment_cute     0
num_friends         0
stars               0
useful              0
funny               0
cool                0
fans                0
compliment_funny    0
categories          0
rating_review       0
dtype: int64
```

Done filling missing value for test set.

```
In [0]: train1.to_csv(path+'new_train.csv',index=False)
        test1.to_csv(path+'new_test.csv',index=False)
```

0.0.2 Prepare segmented test set in user and business dimension

Same data preparation logic applies here, we read the segmented userId and businessID we prepared before to segment the test set into three parts. For user, we have: 1. Unpopular user test set 2. Midpopular user test set 3. Popular user test set

For business, we have: 1. Unpopular business test set 2. Midpopular business test set 3. Popular business test set

```
In [5]: review=pd.read_csv(path+'review.csv')
        del review['text_review']
        review['freq_business'] = review.groupby('business_id')['business_id'].transform('coun
        review2=review.loc[review['freq_business']>2]
        review2['freq_user'] = review2.groupby('user_id')['user_id'].transform('count')
        review3=review2.loc[review2['freq_user']>=5]
        review3=review3.reset_index()
        test_idx=pd.read_csv(path+'all_test_idx_df2.csv')
        test_idx=test_idx.rename({'0':'index'},axis=1)
        test=review3.loc[review3['index'].isin(test_idx['index'])]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/i
  """
```

```
In [0]: path1="/content/drive/My Drive/yelp_segmented_data/"
```

```
In [0]: unpopular_userid=pd.read_csv(path1+'unpopular_user_ID.csv')
        midpopular_userid=pd.read_csv(path1+'midpopular_user_ID.csv')
        popular_userid=pd.read_csv(path1+'popular_user_ID.csv')
```

```
In [0]: unpopular_user=test.loc[test['user_id'].isin(unpopular_userid['userId'])]
        midpopular_user=test.loc[test['user_id'].isin(midpopular_userid['userId'])]
        popular_user=test.loc[test['user_id'].isin(popular_userid['userId'])]

In [0]: user=pd.read_csv(path+'user2.csv')
        business=pd.read_csv(path+'business.csv')
        business=business.rename(columns={"business_ids": "business_id"})

In [0]: unpopular_user=pd.merge(unpopular_user, user, on='user_id')
        midpopular_user=pd.merge(midpopular_user, user, on='user_id')
        popular_user=pd.merge(popular_user, user, on='user_id')

In [0]: unpopular_user = pd.merge(unpopular_user, business, on='business_id')
        midpopular_user = pd.merge(midpopular_user, business, on='business_id')
        popular_user = pd.merge(popular_user, business, on='business_id')

In [0]: unpopular_user=unpopular_user[['index','user_id','business_id','city','state',"average_
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"
        midpopular_user=midpopular_user[['index','user_id','business_id','city','state',"averag
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"
        popular_user=popular_user[['index','user_id','business_id','city','state',"average_sta
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"

In [0]: unpopular_user['useful_review']=unpopular_user['useful_review'].fillna(float(0))
        unpopular_user['funny_review']=unpopular_user['funny_review'].fillna(float(0))
        unpopular_user['cool_review']=unpopular_user['cool_review'].fillna(float(0))
        unpopular_user = unpopular_user.fillna('No_category')

        midpopular_user['useful_review']=midpopular_user['useful_review'].fillna(float(0))
        midpopular_user['funny_review']=midpopular_user['funny_review'].fillna(float(0))
        midpopular_user['cool_review']=midpopular_user['cool_review'].fillna(float(0))
        midpopular_user = midpopular_user.fillna('No_category')

        popular_user['useful_review']=popular_user['useful_review'].fillna(float(0))
        popular_user['funny_review']=popular_user['funny_review'].fillna(float(0))
        popular_user['cool_review']=popular_user['cool_review'].fillna(float(0))
        popular_user = popular_user.fillna('No_category')

In [0]: unpopular_user.to_csv(path+'unpopular_user.csv',index=False)
        midpopular_user.to_csv(path+'midpopular_user.csv',index=False)
        popular_user.to_csv(path+'popular_user.csv',index=False)

In [0]: unpopular_businessid=pd.read_csv(path1+'unpopular_business_ID.csv')
        midpopular_businessid=pd.read_csv(path1+'midpopular_business_ID.csv')
        popular_businessid=pd.read_csv(path1+'popular_business_ID.csv')

In [0]: unpopular_business=test.loc[test['business_id'].isin(unpopular_businessid['businessId']
        midpopular_business=test.loc[test['business_id'].isin(midpopular_businessid['businessI
        popular_business=test.loc[test['business_id'].isin(popular_businessid['businessId'])]
```

```
In [0]: unpopular_business=pd.merge(unpopular_business, user, on='user_id')
        midpopular_business=pd.merge(midpopular_business, user, on='user_id')
        popular_business=pd.merge(popular_business, user, on='user_id')

In [0]: unpopular_business = pd.merge(unpopular_business, business, on='business_id')
        midpopular_business = pd.merge(midpopular_business, business, on='business_id')
        popular_business= pd.merge(popular_business, business, on='business_id')

In [0]: unpopular_business=unpopular_business[['index','user_id','business_id','city','state',
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"
        midpopular_business=midpopular_business[['index','user_id','business_id','city','state
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"
        popular_business=popular_business[['index','user_id','business_id','city','state',"ave
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"

In [0]: unpopular_business['useful_review']=unpopular_business['useful_review'].fillna(float(0
        unpopular_business['funny_review']=unpopular_business['funny_review'].fillna(float(0))
        unpopular_business['cool_review']=unpopular_business['cool_review'].fillna(float(0))
        unpopular_business = unpopular_business.fillna('No_category')

        midpopular_business['useful_review']=midpopular_business['useful_review'].fillna(float
        midpopular_business['funny_review']=midpopular_business['funny_review'].fillna(float(0
        midpopular_business['cool_review']=midpopular_business['cool_review'].fillna(float(0))
        midpopular_business = midpopular_business.fillna('No_category')

        popular_business['useful_review']=popular_business['useful_review'].fillna(float(0))
        popular_business['funny_review']=popular_business['funny_review'].fillna(float(0))
        popular_business['cool_review']=popular_business['cool_review'].fillna(float(0))
        popular_business = popular_business.fillna('No_category')

In [0]: unpopular_business.to_csv(path+'unpopular_business.csv',index=False)
        midpopular_business.to_csv(path+'midpopular_business.csv',index=False)
        popular_business.to_csv(path+'popular_business.csv',index=False)
```