Small_dataset_preparation

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
        import random
```

```
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:
ůůůůůůůůůůů
Mounted at /content/drive
```

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

0.0.1 This part is for creating a small dataset

The reason we want a small dataset is because of the space and running time constrain for the baseline model. We will run all our models on the small dataset. Moreover, we will also run the three more complex models, neural CF, Factorization Machine and Wide & Deep, on a large dataset.

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.

We combine the information of business and user with the rating information by joining the tables on user_id and business_id. We also renumber user_id and business_id from 0.

```
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]
```

```
/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]: review3=review3.reset_index()
```

Read the test index we got before.

```
In [0]: 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'])]
```

```
In [0]: min(test['user_id'].value_counts())
```

```
Out[0]: 3
```

Using the test index to split data into train and test

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

Join the review table with business and user information table on userID and busisnessID.

```
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]: min(test['user_id'].value_counts())
```

```
Out[0]: 3
```

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

```
In [0]: min(test['user_id'].value_counts())
```

```
Out[0]: 3
```

Check if the number of unique users in test set is same as the number of unique users in the training set.

```
In [0]: len(np.unique(test['user_id']))==len(np.unique(train['user_id']))

Out[0]: True
```

Done checking.

```
In [0]: len(train)

Out[0]: 3618100
```

Save the selected columns.

```
In [0]: train1=train[['user_id','business_id','date_review','rating_review']]
        test1=test[['user_id','business_id','date_review','rating_review']]
        train1.to_csv(path+'large_train.csv',index=False)
        test1.to_csv(path+'large_test.csv',index=False)
```

Since we are creating a smaller dataset due to RAM constrain of the labtop when running the baseline model such as item-based collaborative filtering and Non-negative matrix factorization, we randomly select 20000 unique users to create a smaller dataset.

```
In [0]: user_list=random.sample(list(np.unique(test['user_id'])), 20000)

In [0]: train_small=train.loc[train['user_id'].isin(user_list)]
        test_small=test.loc[test['user_id'].isin(user_list)]

In [0]: min(test_small['user_id'].value_counts())

Out[0]: 3

In [0]: train_small=train_small[['user_id','business_id','date_review','rating_review']]
        test_small=test_small[['user_id','business_id','date_review','rating_review']]
```

Save the smaller train and test set.

```
In [0]: train_small.to_csv('train_small.csv',index=False)
        test_small.to_csv(path+'test_small2.csv',index=False)

In [0]: len(test_small)/3

Out[0]: 20000.0
```

0.0.2 Prepare smaller dataset for Factorization Machine model.

In the smaller dataset for FM model, we still use those 20000 randomly selected unique users we selected before. And we save the feature columns we want for training the FM model. We have tried with different combination of feature columns, but for the conciseness of the code, we only show our final choices.

```
In [0]: path1="/content/drive/My Drive/yelp_small_dataset/"
        path="/content/drive/My Drive/yelp_final_data/"
```

```
In [0]: train_small=pd.read_csv(path1+'train_small.csv')

In [0]: test_small=pd.read_csv(path1+'test_small2.csv')

In [0]: min(test_small['user_id'].value_counts())

Out[0]: 3

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_small = pd.merge(train_small, user, on='user_id')
        test_small=pd.merge(test_small, user, on='user_id')

In [0]: train_small = pd.merge(train_small, business, on='business_id')
        test_small = pd.merge(test_small, business, on='business_id')

In [0]: min(test_small['user_id'].value_counts())

Out[0]: 3

In [0]: train1=train_small[['user_id','business_id','average_stars','stars','city','state','ra
        test1=test_small[['user_id','business_id','average_stars','stars','city','state','rati

In [0]: train1.head(2)

Out[0]:                  user_id             business_id  ...  state  rating_review
        0  pkq41Qh9yGOI_4pwdVmmDg  ujmEBvifdJM6h6RLv4wQIg  ...     NV            5.0
        1  ARvvk5AcVtNREPiT1ZHNcw  ujmEBvifdJM6h6RLv4wQIg  ...     NV            5.0

        [2 rows x 7 columns]
```

Renumber the user_id and business_id from 0.

```
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 else
        test1=test1.applymap(lambda s: business_id_dict.get(s) if s in business_id_dict else s

In [0]: train1.head(2)

Out[0]:    user_id  business_id  average_stars  stars       city state  rating_review
        0        0        75020           3.95    2.5  Las Vegas    NV            5.0
        1        1        75020           3.97    2.5  Las Vegas    NV            5.0

In [0]: np.savetxt(path+'test1_small.txt', test1.values, fmt=['%d','%d','%d','%d','%s','%s','%

In [0]: np.savetxt(path+'train1_small.txt', train1.values, fmt=['%d','%d','%d','%d','%s','%s',
```

0.0.3 Prepare smaller dataset for Wide and Deep model.

Same logic applies when creating smaller dataset for Wide and Deep model.

```
In [0]: path1="/content/drive/My Drive/yelp_small_dataset/"
        path="/content/drive/My Drive/yelp_final_data/"

In [0]: train_small=pd.read_csv(path1+'train_small.csv')

In [0]: test_small=pd.read_csv(path1+'test_small2.csv')

In [0]: min(test_small['user_id'].value_counts())

Out[0]: 3
```

Read user and business CSV files and join the tables on 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_small = pd.merge(train_small, user, on='user_id')
        test_small=pd.merge(test_small, user, on='user_id')

In [0]: train_small = pd.merge(train_small, business, on='business_id')
        test_small = pd.merge(test_small, business, on='business_id')

In [0]: min(test_small['user_id'].value_counts())

Out[0]: 3

In [0]: test_small.head(2)

Out[0]:                  user_id  ...                                                hours
        0  gJ1zjuo0V4YUfm3o7700Jg  ...  {'Monday': '17:30-22:0', 'Tuesday': '17:30-22:...
        1  8rqdVEvoVCgWPli5Omt1wg  ...  {'Monday': '17:30-22:0', 'Tuesday': '17:30-22:...

        [2 rows x 32 columns]
```

Select features we want. Same as in FM model, we only show our final choices of feature columns.

```
In [0]: train1=train_small[['user_id','business_id','city','state',"average_stars",'compliment_
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"
        test1=test_small[['user_id','business_id','city','state',"average_stars",'compliment_mo
        'num_friends','stars','useful','funny','cool','fans','compliment_funny','categories',"

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

```
Out[0]:                  user_id  ... rating_review
        0  pkq41Qh9yGOI_4pwdVmmDg  ...           5.0
        1  ARvvk5AcVtNREPiT1ZHNcw  ...           5.0

        [2 rows x 16 columns]
```

Save the smaller dataset for Wide and Deep model.

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