Вебинар 4. Домашнее задание

Само домашнее задание нахожится в конце ноутбука

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

# Для работы с матрицами
from scipy.sparse import csr_matrix

# Матричная факторизация
from implicit.als import AlternatingLeastSquares
from implicit.nearest_neighbours import ItemItemRecommender  # нужен для одного трюка
from implicit.nearest_neighbours import bm25_weight, tfidf_weight

# Функции из 1-ого вебинара
import os, sys

module_path = os.path.abspath(os.path.join(os.pardir))
if module_path not in sys.path:
    sys.path.append(module_path)

from src.metrics import precision_at_k, recall_at_k
from src.utils import prefilter_items
```

In [2]:
```
data = pd.read_csv('../raw_data/transaction_data.csv')

data.columns = [col.lower() for col in data.columns]
data.rename(columns={'household_key': 'user_id',
                     'product_id': 'item_id'},
            inplace=True)


test_size_weeks = 3

data_train = data[data['week_no'] < data['week_no'].max() - test_size_weeks]
data_test = data[data['week_no'] >= data['week_no'].max() - test_size_weeks]

data_train.head(2)
```

Out[2]:

	user_id	basket_id	day	item_id	quantity	sales_value	store_id	retail_disc	trans_time	week_no	coupon_disc	coupon_match_disc
0	2375	26984851472	1	1004906	1	1.39	364	-0.6	1631	1	0.0	0.0
1	2375	26984851472	1	1033142	1	0.82	364	0.0	1631	1	0.0	0.0

In [3]:
```
# data[(data['week_no'] < 48) & (data['sales_value'] == 0)].count(), data['sales_value'].isna().sum()
```

In [4]:
```
item_features = pd.read_csv('../raw_data/product.csv')
item_features.columns = [col.lower() for col in item_features.columns]
item_features.rename(columns={'product_id': 'item_id'}, inplace=True)

item_features.head(2)
```

Out[4]:

	item_id	manufacturer	department	brand	commodity_desc	sub_commodity_desc	curr_size_of_product
0	25671	2	GROCERY	National	FRZN ICE	ICE - CRUSHED/CUBED	22 LB
1	26081	2	MISC. TRANS.	National	NO COMMODITY DESCRIPTION	NO SUBCOMMODITY DESCRIPTION	

In [5]:
```
result = data_test.groupby('user_id')['item_id'].unique().reset_index()
result.columns=['user_id', 'actual']
result.head(2)
```

Out[5]:

	user_id	actual
0	1	[879517, 934369, 1115576, 1124029, 5572301, 65...
1	3	[823704, 834117, 840244, 913785, 917816, 93870...

In [6]:
```
# Возбмем топ по популярности
popularity = data.groupby('item_id')['quantity'].sum().reset_index()
popularity.rename(columns={'quantity': 'n_sold'}, inplace=True)

take_n_popular=6000
# Уберутся 1000 самых популярных товаров (их и так купят)
# Останется 5000
```

```
top = popularity.sort_values('n_sold', ascending=False).head(take_n_popular).item_id.tolist()
print(data.shape)
#top
```

```
(2595732, 12)
```

In [7]:
```
n_items_before = data_train['item_id'].nunique()

# data_train = prefilter_items(data_train, item_features, take_n_popular=5000)
# data_train = prefilter_items(data_train, take_n_popular=5000, item_features=None)
data_train = prefilter_items(data_train, take_n_popular=5000, item_features=item_features)

n_items_after = data_train['item_id'].nunique()
print('Decreased # items from {} to {}'.format(n_items_before, n_items_after))
```

```
Decreased # items from 90386 to 4001
```

In [8]:
```
user_item_matrix = pd.pivot_table(data_train,
                                  index='user_id', columns='item_id',
                                  values='quantity', # Можно пробоват ьдругие варианты
                                  aggfunc='count',
                                  fill_value=0
                                 )

user_item_matrix = user_item_matrix.astype(float) # необходимый тип матрицы для implicit

user_item_matrix.head(3)
```

Out[8]:

item_id	42346	42521	78986	108646	244960	818981	819048	819400	819590	819840	...	12812474	12946257	12949590	13002975	13003092
user_id																
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

3 rows × 4001 columns

In [9]:
```
print(type(user_item_matrix))
user_item_matrix.shape
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[9]:
```
(2469, 4001)
```

In [10]:
```
user = 2
user_items = user_item_matrix.reset_index()
user_items[user_items.user_id == user]
```

Out[10]:

item_id	user_id	42346	42521	78986	108646	244960	818981	819048	819400	819590	...	12812474	12946257	12949590	13002975	13003092
1	2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0

1 rows × 4002 columns

In [11]:
```
user_item_matrix.shape
```

Out[11]:
```
(2469, 4001)
```

In [12]:
```
column_list = []  # Список из 5 товаров приобретённых пользователем
k=0
for col in list(user_items.columns[1:]):
    if user_items[col][0] != 0.0:
        column_list.append(col)
        k += 1
        if k == 5:
            break

column_list
```

Out[12]:
```
[825123, 831447, 835796, 849066, 852014]
```

In [13]:
```
#
# Функция реализована также в классе MainRecommender (см. файл recommenders.py)
#

def get_similar_items_recommendation(user, N = 5):
    """Рекомендуем товары, похожие на топ-N купленных юзером товаров"""
    # # См. лекция 3
    # your_code
    # Практически полностью реализовали на прошлом вебинаре

    user_items = user_item_matrix.getrow(user-1).todense()
    user_items = user_items.tolist()

    user_item_list = []  # список из N item_id приобретённых пользователем user
    for row in user_items:
        i = 0
```

```
        k = 0
        for element in row:
            if element != 0:
                user_item_list.append(i)
                # print(i, id_to_itemid[i], element)
                k += 1
            i += 1
            if k == N:
                break
    # Находим товары похожие на товары приобретённые пользователем

    closest_items = []  # Список рекомендуемых товаров

    for item_row_id in user_item_list:
        print(item_row_id)
        closest_items.append([id_to_itemid[row_id] for row_id, score in model.similar_items(item_row_id, N=
    res = closest_items

    assert len(res) == N, 'Количество рекомендаций != {}'.format(N)
    return res
```

In [14]:
```
userids = user_item_matrix.index.values
itemids = user_item_matrix.columns.values

matrix_userids = np.arange(len(userids))
matrix_itemids = np.arange(len(itemids))

id_to_itemid = dict(zip(matrix_itemids, itemids))
id_to_userid = dict(zip(matrix_userids, userids))

itemid_to_id = dict(zip(itemids, matrix_itemids))
userid_to_id = dict(zip(userids, matrix_userids))
```

In [15]:
```
user_item_matrix = bm25_weight(user_item_matrix.T).T  # Применяется к item-user матрице !
```

In [16]:
```
%%time

model = AlternatingLeastSquares(factors=20,
                                regularization=0.001,
                                iterations=15,
                                calculate_training_loss=True,
                                num_threads=4)

model.fit(csr_matrix(user_item_matrix).T.tocsr(),  # На вход item-user matrix
          show_progress=True)
```

```
WARNING:root:Intel MKL BLAS detected. Its highly recommend to set the environment variable 'export MKL_NUM
_THREADS=1' to disable its internal multithreading
```

100% 15/15 [00:06<00:00, 2.41it/s, loss=0.0503]

```
CPU times: user 2.73 s, sys: 698 ms, total: 3.43 s
Wall time: 3.69 s
```

In [17]:
```
# rows, cols = user_item_matrix.nonzero()
# print(rows)
# print(cols)
# print(user_item_matrix.todense()[0:5, 0:5])
```

In [18]:
```
user_id = 0
print(id_to_userid[user_id])
matrix = user_item_matrix.getrow(user_id).todense()
matrix = matrix.tolist()

for row in matrix:
    # print(row)
    i = 0
    k = 0
    for element in row:
        if element != 0:
            print(i, id_to_itemid[i], element)
            k += 1
        i += 1
        if k == 5:
            break
```

```
1
69 825123 11.311669119549721
132 831447 17.858477286992223
172 835796 9.125795501307183
285 849066 5.832597485105489
318 852014 5.4574697730918365
```

In [19]:
```
user = 10
get_similar_items_recommendation(user, 5)
```

```
438
565
942
1726
```

Out[19]:
```
[[866158], [879876], [918345], [999999], [1000050]]
```

Домашнее задание

1. Перенесите метрики в модуль metrics.py (убедится что они там)
2. Перенесите функцию prefilter_items в модуль utils.py
3. Создайте модуль recommenders.py. Напищите код для класса ниже (задание обсуждали на вебинаре, для первой функции практически сделали) и положите его в recommenders.py
4. Проверьте, что все модули корректно импортируются

Проверка, что все работает

In [20]:
```
from src.metrics import precision_at_k, recall_at_k
from src.utils import prefilter_items
from src.recommenders import MainRecommender
```

In [21]:
```
# pwd
```

In [22]:
```
# ls
```

Инициализируем класс

In [23]:
```
data_mr = MainRecommender(data_test)
```

```
item_id  27160     27658     28268     29629     30613     31126     31704     \
user_id
1             0.0       0.0       0.0       0.0       0.0       0.0       0.0
3             0.0       0.0       0.0       0.0       0.0       0.0       0.0
5             0.0       0.0       0.0       0.0       0.0       0.0       0.0

item_id  33048     33240     33965     ...  18273019  18273051  18273115  \
user_id                                ...
1             0.0       0.0       0.0  ...       0.0       0.0       0.0
3             0.0       0.0       0.0  ...       0.0       0.0       0.0
5             0.0       0.0       0.0  ...       0.0       0.0       0.0

item_id  18273133  18292005  18293142  18293439  18293696  18294080  18316298
user_id
1             0.0       0.0       0.0       0.0       0.0       0.0       0.0
3             0.0       0.0       0.0       0.0       0.0       0.0       0.0
5             0.0       0.0       0.0       0.0       0.0       0.0       0.0

[3 rows x 23048 columns]
```

```
100%                                      15/15 [00:02<00:00, 5.12it/s]
```

```
100%                                      23048/23048 [00:00<00:00, 24506.59it/s]
```

In [24]:
```
data_mr
```

Out[24]:
```
<src.recommenders.MainRecommender at 0x7f686fe0d9d0>
```

In [25]:
```
data_mr.user_item_matrix
```

Out[25]:
```
<1991x23048 sparse matrix of type '<class 'numpy.float64'>'
        with 96922 stored elements in COOrdinate format>
```

In [26]:
```
user=10
```

Рекомендуем товары, похожие на топ-N купленных юзером товаров

In [27]:
```
data_mr.get_similar_items_recommendation(user, 5)
```

```
Список 5 товаров приобретённых пользователем 10:
 [902377, 994928, 1056746, 1094308, 1133018]
```

Out[27]:
```
[[902377], [994928], [1056746], [1094308], [1133018]]
```

Рекомендуем топ-N товаров, среди купленных похожими юзерами

In [28]:
```
data_mr.get_similar_users_recommendation(user, 5)
```

```
Список пользователей похожих на пользователя 10:
 [7, 307, 941, 1926, 400]
```

Out[28]:
```
[[821344], [9337170], [826272], [833025], [823990]]
```

In [29]:
```
userid=7
print(userid_to_id[userid])
data_mr.userid_to_id[userid]
```

```
6
```

Out[29]:
```
4
```

In [ ]:
```

```