Вебинар 4. Домашнее задание

Само домашнее задание нахожится в конце ноутбука

In [284]:

```
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

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

In [285]:

```
data = pd.read_csv('../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()
```

Out[285]:

	user_id	basket_id	day	item_id	quantity	sales_value	store_id	retail_disc	trans_time
0	2375	26984851472	1	1004906	1	1.39	364	-0.60	1631
1	2375	26984851472	1	1033142	1	0.82	364	0.00	1631
2	2375	26984851472	1	1036325	1	0.99	364	-0.30	1631
3	2375	26984851472	1	1082185	1	1.21	364	0.00	1631
4	2375	26984851472	1	8160430	1	1.50	364	-0.39	1631

In [286]:

```
data_test['week_no'].max()
```

Out[286]:

```
102
```

In [287]:

```
item_features = pd.read_csv('../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()
```

Out[287]:

	item_id	manufacturer	department	brand	commodity_desc	sub_commodity_desc	curr
0	25671	2	GROCERY	National	FRZN ICE	ICE - CRUSHED/CUBED	
1	26081	2	MISC. TRANS.	National	NO COMMODITY DESCRIPTION	NO SUBCOMMODITY DESCRIPTION	
2	26093	69	PASTRY	Private	BREAD	BREAD:ITALIAN/FRENCH	
3	26190	69	GROCERY	Private	FRUIT - SHELF STABLE	APPLE SAUCE	
4	26355	69	GROCERY	Private	COOKIES/CONES	SPECIALTY COOKIES	

In [288]:

```
data.shape
```

Out[288]:

```
(2595732, 12)
```

In [289]:

```
item_features.shape
```

Out[289]:

```
(92353, 7)
```

In [290]:

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

Out[290]:

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

In [291]:

```
data_test
```

Out[291]:

	user_id	basket_id	day	item_id	quantity	sales_value	store_id	retail_disc	tran
2479936	790	41931228451	685	822989	1	7.29	31782	0.00	
2479937	790	41931228451	685	849003	1	2.59	31782	0.00	
2479938	790	41931228451	685	884731	1	2.19	31782	0.00	
2479939	790	41931228451	685	920109	1	3.08	31782	0.00	
2479940	790	41931228451	685	976998	3	1.77	31782	0.00	
...	...	...	...	...	...	...	...	...	
2595727	1598	42305362535	711	92130	1	0.99	3228	0.00	
2595728	1598	42305362535	711	114102	1	8.89	3228	0.00	
2595729	1598	42305362535	711	133449	1	6.99	3228	0.00	
2595730	1598	42305362535	711	6923644	1	4.50	3228	-0.49	
2595731	1598	42305362535	711	14055192	1	6.99	3228	0.00	

110194 rows × 12 columns

In [292]:

```
data_train['item_id'].nunique()
```

Out[292]:

```
90386
```

In [293]:

```
data_train.shape
```

Out[293]:

```
(2485538, 12)
```

In [294]:

```
data_train = data_train[:10000]
```

In [295]:

```
data_train.shape
```

Out[295]:

```
(10000, 12)
```

In [296]:

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

# data_train = prefilter_items(data_train, item_features, take_n_popular=5000)

n_items_after = data_train['item_id'].nunique()

print('Decreased # items from {} to {}'.format(n_items_before, n_items_after))
```

```
Decreased # items from 5913 to 5913
```

In [297]:

```
user_item_matrix = pd.pivot_table(data_train,
                                  index='user_id', columns='item_id',
                                  values='quantity', # Можно пробовать другие варианты
                                  aggfunc='count',
                                  fill_value=0
                                 )

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

user_item_matrix.head()
```

Out[297]:

item_id	34198	39592	40885	50905	70302	72816	78888	108623	113154	122280	...	98376
user_id												
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
46	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0
51	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0

5 rows × 5913 columns

In [298]:

```
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 [299]:

```
userid_to_id[14]
```

Out[299]:

```
0
```

In [300]:

```
user_item_matrix = bm25_weight(user_item_matrix.T).T  # Применяется к item-user матрице !
```

In [301]:

```
%%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)
```

```
  0%|          | 0/15 [00:00<?, ?it/s]
Wall time: 3.28 s
```

In [302]:

```
userid_to_id[25]
```

Out[302]:

```
1
```

In [ ]:

```

```

In [303]:

```
userid = 1
user_items = user_item_matrix
N =5
```

In [304]:

```
def get_recommendations(user, model, sparse_user_item, N=5):
    """Рекомендуем топ-N товаров"""

    res = [id_to_itemid[rec[0]] for rec in
                    model.recommend(userid=userid_to_id[user],
                                    user_items=sparse_user_item,   # на вход user-item matr
                                    N=N,
                                    filter_already_liked_items=False,
                                    filter_items=[itemid_to_id[1005609]],  # !!!
                                    recalculate_user=True)]
    res_2 = [list(item_features.loc[item_features['item_id']== i, 'commodity_desc']) for i

    result = pd.DataFrame(list(zip(res, res_2)), columns= ('item', 'name'))
    return result
```

In [305]:

```
get_recommendations(25, model, user_item_matrix)
```

Out[305]:

	item	name
0	903567	[CANDY - PACKAGED]
1	1130666	[SEAFOOD - FROZEN]
2	869728	[MISC. DAIRY]
3	948756	[BAKING MIXES]
4	973374	[ONIONS]

In [306]:

```
data.head()
```

Out[306]:

	user_id	basket_id	day	item_id	quantity	sales_value	store_id	retail_disc	trans_time
0	2375	26984851472	1	1004906	1	1.39	364	-0.60	1631
1	2375	26984851472	1	1033142	1	0.82	364	0.00	1631
2	2375	26984851472	1	1036325	1	0.99	364	-0.30	1631
3	2375	26984851472	1	1082185	1	1.21	364	0.00	1631
4	2375	26984851472	1	8160430	1	1.50	364	-0.39	1631

In [ ]:

```

```

In [307]:

```
b = [13,27,38,54,65]
c = ['a', 'b', 'c', 'd', 'e']
f = [0,1,1,1,0]
p = [13, 27]
```

In [308]:

```
a = pd.DataFrame([b,c,f])
a = a.T
a.columns = ['b', 'c', 'f']
a
```

Out[308]:

	b	c	f
0	13	a	0
1	27	b	1
2	38	c	1
3	54	d	1
4	65	e	0

In [309]:

```
a = a[(a['b'] > 13) & (a['b']<54)]
a
```

Out[309]:

	b	c	f
1	27	b	1
2	38	c	1

In [310]:

```
if 'f' in a:
    print('yes')
else:
    print('no')
```

```
yes
```

In [311]:

```
a.loc[(a['c'] == 'b') | (a['b']> 13)]
```

Out[311]:

	b	c	f
1	27	b	1
2	38	c	1

In [197]:

```
def prefilter_items(data, n_top= 5000):

    # Уберем самые популярные товары (их и так купят)
    if 'quantity' in data.columns:
        popularity = data.groupby('item_id')['quantity'].sum().reset_index()
        popularity.rename(columns={'quantity': 'n_sold'}, inplace=True)
        top_n = popularity.sort_values('n_sold', ascending=False).head(n_top).item_id.tolis
        data = data.loc[~data['item_id'].isin([i for i in top_n])]

    # Уберем товары, которые не продавались за последние 12 месяцев

        week_53 = list(range(53))
        data = data.loc[~data['quantity'].isin([0]) & ~data['week_no'].isin([i for i in wee


    # Уберем слишком дешевые товары (на них не заработаем). 1 покупка из рассылок стоит 60
    # Уберем слишком дорогие товары
    elif 'price' in data.columns:

        data = data[(data['price'] > 61) & (data['price'] < 5000)]
    # Уберем не интересные для рекоммендаций категории (department)
    # мясо и так купят, медикаменты можно только по рекомендации врача, 'MISC. TRANS.'- не
    else:
        cat_prod = ['MISC. TRANS.','MEAT-PCKGD','MEAT','DRUG GM']
        data = data.loc[~data['department'].isin([i for i in cat_prod])]



    return data


def postfilter_items(user_id, recommednations):
    pass
```

In [312]:

```
data.shape
```

Out[312]:

```
(2595732, 12)
```

In [199]:

```
data = prefilter_items(data)
```

In [200]:

```
data.shape
```

Out[200]:

```
(507035, 12)
```

In [201]:

```
len(item_features['department'].unique())
```

Out[201]:

```
44
```

In [202]:

```
item_features = prefilter_items(item_features)
```

In [203]:

```
len(item_features['department'].unique())
```

Out[203]:

```
40
```

In [ ]:

```

```

get_similar_items_recommendation

In [313]:

```
from collections import Counter
```

In [314]:

```
result['actual_top_5'] = [[element for element,count in Counter(i).most_common(3)] for i in
result.drop('actual',1, inplace=True)
result.head()
```

Out[314]:

	user_id	actual_top_5
0	1	[879517, 934369, 1115576]
1	3	[823704, 834117, 840244]
2	5	[913077, 1118028, 1386668]
3	6	[825541, 859676, 999318]
4	7	[929248, 948622, 1013572]

In [315]:

```
#create Series
s = result.set_index('user_id')['actual_top_5']
#create DataFrame, reshape by stack and conver MultiIndex to columns
df = pd.DataFrame(s.values.tolist(), index=s.index).stack().reset_index()
df.columns= ['user_id','i','actual_top_5']
```

In [316]:

```
df_top = df.groupby(['user_id','actual_top_5']).size().reset_index(name='count')
# res = result.groupby(['user_id', 'actual_top_5']).size()#.reset_index(name='count')
```

In [317]:

```
df_top['actual_top_5']=df['actual_top_5'].astype('int')
```

In [318]:

```
df_top.drop('count',1, inplace= True)
```

In [319]:

```
df_top.rename(columns={'actual_top_5': 'item_id'}, inplace=True)
```

In [325]:

```
df_top[:7]
```

Out[325]:

	user_id	item_id
0	1	879517
1	1	934369
2	1	1115576
3	3	823704
4	3	834117
5	3	840244
6	5	913077

In [321]:

```
res_top = data.merge(df_top, on=["item_id", 'user_id' ], how= 'right')
```

In [324]:

```
res_top.loc[res_top['user_id'] == 3, 'item_id']
```

Out[324]:

```
15    823704
16    834117
17    840244
18    840244
Name: item_id, dtype: int64
```

In [327]:

```
res_top.head()
```

Out[327]:

	user_id	basket_id	day	item_id	quantity	sales_value	store_id	retail_disc	trans_time
0	1	41970711432	687	879517	1	3.99	436	-0.50	1754
1	1	28282581446	108	934369	1	1.50	436	-0.69	1513
2	1	30578772112	235	934369	1	1.39	436	-0.80	1705
3	1	30700731221	246	934369	1	1.39	436	-0.80	1143
4	1	31172831466	282	934369	1	1.59	436	-0.60	1528

In [328]:

```
user_item_matrix = pd.pivot_table(res_top,
                                  index='user_id', columns='item_id',
                                  values='quantity', # Можно пробовать другие варианты
                                  aggfunc='count',
                                  fill_value=0
                                 )

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

user_item_matrix.head()
```

Out[328]:

item_id	27658	28268	36055	36608	40694	43020	45720	46967	57390	66746	...	17900917
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
3	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	...	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0

5 rows × 3187 columns

In [329]:

```
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 [330]:

```
user_item_matrix = bm25_weight(user_item_matrix.T).T  # Применяется к item-user матрице !
```

In [331]:

```
%%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)
```

```
  0%|          | 0/15 [00:00<?, ?it/s]
```

```
Wall time: 2.97 s
```

In [ ]:

```

```

In [332]:

```
def get_similar_items_recommendation(user, model,sparse_user_item, N=5):
    """Рекомендуем товары, похожие на топ-N купленных юзером товаров"""

    res = [id_to_itemid[rec[0]] for rec in
                    model.recommend(userid=userid_to_id[user],
                                    user_items=sparse_user_item,   # на вход user-item matr
                                    N=N,
                                    filter_already_liked_items=False,
#                                     filter_items=[itemid_to_id[1005609]],  # !!!
                                    recalculate_user=True)]
    res_2 = [list(item_features.loc[item_features['item_id']== i, 'sub_commodity_desc']) fo

    result = pd.DataFrame(list(zip(res, res_2)), columns= ('item', 'name'))
    return result

    # your_code

    return res
```

In [337]:

```
get_similar_items_recommendation(3, model, user_item_matrix)
```

Out[337]:

	item	name
0	823704	[ROLLS - PORK]
1	872137	[BUTTER]
2	840227	[FACIAL TISSUE & PAPER HANDKE]
3	909497	[CANDY REFRIGERATED]
4	824180	[SW GDS:DONUTS]

In [338]:

```
840244
item_features[item_features['item_id']== 823704]
```

Out[338]:

	item_id	manufacturer	department	brand	commodity_desc	sub_commodity_des
6354	823704	2082	MEAT-PCKGD	National	BREAKFAST SAUSAGE/SANDWICHES	ROLLS - POR

In [ ]:

```
```

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

In [339]:

```
import pandas as pd
import numpy as np

# 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


class MainRecommender:
    """Рекоммендации, которые можно получить из ALS

    Input
    -----
    user_item_matrix: pd.DataFrame
        Матрица взаимодействий user-item
    """

    def __init__(self, data, weighting=True):

        # your_code. Это не обязательная часть. Но если вам удобно что-либо посчитать тут -

        self.user_item_matrix = self.prepare_matrix(data)  # pd.DataFrame
        self.id_to_itemid, self.id_to_userid, self.itemid_to_id, self.userid_to_id = prepar

        if weighting:
            self.user_item_matrix = bm25_weight(self.user_item_matrix.T).T

        self.model = self.fit(self.user_item_matrix)
        self.own_recommender = self.fit_own_recommender(self.user_item_matrix)

    @staticmethod
    def prepare_matrix(data: pd.DataFrame):

        # your_code
        user_item_matrix = pd.pivot_table(data,
                                          index='user_id', columns='item_id',
                                          values='quantity', # Можно пробовать другие варианты
                                          aggfunc='count',
                                          fill_value=0
                                         )

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


        return user_item_matrix

    @staticmethod
    def prepare_dicts(user_item_matrix):
        """Подготавливает вспомогательные словари"""

        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))

        return id_to_itemid, id_to_userid, itemid_to_id, userid_to_id

    @staticmethod
    def fit_own_recommender(user_item_matrix):
        """Обучает модель, которая рекомендует товары, среди товаров, купленных юзером"""

        own_recommender = ItemItemRecommender(K=1, num_threads=4)
        own_recommender.fit(csr_matrix(user_item_matrix).T.tocsr())

        return own_recommender

    @staticmethod
    def fit(user_item_matrix, factors=20, regularization=0.001, iterations=15, num_threads=
        """Обучает ALS"""

        model = AlternatingLeastSquares(factors=factors,
                                        regularization=regularization,
                                        iterations=iterations,
                                        num_threads=num_threads)
        model.fit(csr_matrix(self.user_item_matrix).T.tocsr())

        return model

    def get_similar_items_recommendation(self, user, N=5):
        """Рекомендуем товары, похожие на топ-N купленных юзером товаров"""

        # your_code
        # Практически полностью реализовали на прошлом вебинаре

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

    def get_similar_users_recommendation(self, user, N=5):
        """Рекомендуем топ-N товаров, среди купленных похожими юзерами"""

        # your_code

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

Проверка, что все работает

In [340]:

```
from metrics import precision_at_k, recall_at_k
from utils import prefilter_items
from recommenders import MainRecommender
```

In [ ]:

```

```