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, coo_matrix

# Матричная факторизация
from implicit.als import AlternatingLeastSquares
from implicit.nearest_neighbours import bm25_weight, tfidf_weight

from lightfm import LightFM

# 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)
```

```
C:\Users\voron\AppData\Roaming\Python\Python37\site-packages\lightfm\_lightf
m_fast.py:10: UserWarning: LightFM was compiled without OpenMP support. Only
a single thread will be used.
  "LightFM was compiled without OpenMP support. "
```

In [2]:

```
from lightfm.evaluation import precision_at_k, recall_at_k

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

In [3]:

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

item_features = pd.read_csv('../data/product.csv')
user_features = pd.read_csv('../data/hh_demographic.csv')

# column processing
item_features.columns = [col.lower() for col in item_features.columns]
user_features.columns = [col.lower() for col in user_features.columns]

item_features.rename(columns={'product_id': 'item_id'}, inplace=True)
user_features.rename(columns={'household_key': 'user_id'}, inplace=True)

# train test split
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[3]:

	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	-0.6	1631
1	2375	26984851472	1	1033142	1	0.82	364	0.0	1631

In [4]:

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

Out[4]:

	user_id	actual
0	1	[821867, 834484, 856942, 865456, 889248, 90795...
1	3	[835476, 851057, 872021, 878302, 879948, 90963...

In [5]:

```
item_features.head(2)
```

Out[5]:

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

In [6]:

```
user_features.head(2)
```

Out[6]:

	age_desc	marital_status_code	income_desc	homeowner_desc	hh_comp_desc	household_s
0	65+	A	35-49K	Homeowner	2 Adults No Kids	
1	45-54	A	50-74K	Homeowner	2 Adults No Kids	

In [7]:

```
user_features['age_desc'].unique()
```

Out[7]:

```
array(['65+', '45-54', '25-34', '35-44', '19-24', '55-64'], dtype=object)
```

In [8]:

```
user_features['marital_status_code'].unique()
```

Out[8]:

```
array(['A', 'U', 'B'], dtype=object)
```

In [9]:

```
user_features['household_size_desc'].unique()
```

Out[9]:

```
array(['2', '3', '4', '1', '5+'], dtype=object)
```

1. Filter items

In [10]:

```
data_train.head()
```

Out[10]:

	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 [11]:

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

data_train_filtered = prefilter_items(data_train, take_n_popular=5000, item_features=item_f

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

```
C:\Users\voron\a учеба\рекомендательные системы\les 5\utils.py:20: SettingWi
thCopyWarning:
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: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data['price'] = data['sales_value'] / (np.maximum(data['quantity'], 1))

Decreased # items from 86865 to 5001
```

2. Prepare data set

2.1 Prepare csr train matrix

In [12]:

```

user_item_matrix = pd.pivot_table(data_train_filtered,
                                  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

# переведем в формат sparse matrix
sparse_user_item = csr_matrix(user_item_matrix).tocsr()

user_item_matrix.head(2)
```

Out[12]:

item_id	117847	818981	819255	819308	819400	819487	819590	819594	819840	819845	..
user_id											
1	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	0.0	0.0	0.0	0.0	0.0	0.0	..

2 rows × 5001 columns

2.2 Prepare CSR test matrix

In [13]:

```
data_test = data_test[data_test['item_id'].isin(data_train['item_id'].unique())]

test_user_item_matrix = pd.pivot_table(data_test,
                                  index='user_id', columns='item_id',
                                  values='quantity', # Можно пробоват ьдругие варианты
                                  aggfunc='count',
                                  fill_value=0
                                 )

test_user_item_matrix = test_user_item_matrix.astype(float) # необходимый тип матрицы для i
```

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))
```

3. Prepare user and item features

In [15]:

```
user_feat = pd.DataFrame(user_item_matrix.index)
user_feat = user_feat.merge(user_features, on='user_id', how='left')
user_feat.set_index('user_id', inplace=True)
user_feat.head(2)
```

Out[15]:

	age_desc	marital_status_code	income_desc	homeowner_desc	hh_comp_desc	housel
user_id						
1	65+	A	35-49K	Homeowner	2 Adults No Kids	
2	NaN	NaN	NaN	NaN	NaN	

In [16]:

```
user_feat.shape
```

Out[16]:

```
(2497, 7)
```

In [17]:

```
item_feat = pd.DataFrame(user_item_matrix.columns)
item_feat = item_feat.merge(item_features, on='item_id', how='left')
item_feat.set_index('item_id', inplace=True)

item_feat.head(2)
```

Out[17]:

	manufacturer	department	brand	commodity_desc	sub_commodity_desc	curr_size_
item_id						
117847	450.0	NUTRITION	National	REFRIGERATED	SOY/RICE MILK	
818981	194.0	GROCERY	National	COLD CEREAL	ALL FAMILY CEREAL	

In [18]:

```
item_feat.shape
```

Out[18]:

```
(5001, 6)
```

Encoding features

In [19]:

```
user_feat_lightfm = pd.get_dummies(user_feat, columns=user_feat.columns.tolist())
item_feat_lightfm = pd.get_dummies(item_feat, columns=item_feat.columns.tolist())
```

In [20]:

```
user_feat_lightfm.head(2)
```

Out[20]:

	age_desc_19-24	age_desc_25-34	age_desc_35-44	age_desc_45-54	age_desc_55-64	age_desc_65+
user_id						
1	0	0	0	0	0	1
2	0	0	0	0	0	0

2 rows × 41 columns

Init model

In [21]:

```
from sklearn.model_selection import GridSearchCV
```

In [23]:

```
param_grid = {'learning_rate': [0.05, 0.1], }
```

In [ ]:

```

```

In [24]:

```
model = LightFM(no_components=40,
                loss='warp', # "logistic","bpr"
                item_alpha=0.4,
                user_alpha=0.1,
                random_state=42,
                k=5,
                n=15,
                max_sampled=100)
```

In [25]:

```

grid = GridSearchCV(estimator=model, param_grid=param_grid, cv= 5)
```

Train

In [26]:

```
model.fit((sparse_user_item > 0) * 1,  # user-item matrix из 0 и 1
          sample_weight=coo_matrix(user_item_matrix),
          user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
          item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
          epochs=20,
          num_threads=20,
          verbose=True)
```

```
Epoch: 100%|████████████████████████████████████████████████████████████████
███████████| 20/20 [03:44<00:00, 11.20s/it]
```

Out[26]:

```
<lightfm.lightfm.LightFM at 0x24e0b4883c8>
```

Getting embeddings

вектора по пользователям

In [27]:

```
user_emb = model.get_user_representations(features=csr_matrix(user_feat_lightfm.values).toc
```

In [28]:

```
user_emb[0].shape # biases
```

Out[28]:

```
(2497,)
```

In [29]:

```
user_emb[1].shape # users vectors
```

Out[29]:

```
(2497, 40)
```

вектора по товарам

In [30]:

```
item_emb = model.get_item_representations(features=csr_matrix(item_feat_lightfm.values).toc
```

In [31]:

```
item_emb[0].shape # biases
```

Out[31]:

```
(5001,)
```

In [32]:

```
item_emb[1].shape # items vectors
```

Out[32]:

```
(5001, 40)
```

Evaluation -> Train precision

In [33]:

```
# мы можем использовать встроенные метрики lightFM
train_precision = precision_at_k(model, sparse_user_item,
                                 user_features=csr_matrix(user_feat_lightfm.values).tocsr()
                                 item_features=csr_matrix(item_feat_lightfm.values).tocsr()
                                 k=5).mean()

print(f"Train precision {train_precision}")
```

```
Train precision 0.2941129505634308
```

Predict

In [34]:

```
# подготавливаемм id для юзеров и товаров в порядке пар user-item
users_ids_row = data_train_filtered['user_id'].apply(lambda x: userid_to_id[x]).values.asty
items_ids_row = data_train_filtered['item_id'].apply(lambda x: itemid_to_id[x]).values.asty
```

In [35]:

```
# модель возвращает меру/скор похожести между соответствующим пользователем и товаром
predictions = model.predict(user_ids=users_ids_row,
                            item_ids=items_ids_row,
                            user_features=csr_matrix(user_feat_lightfm.values).tocsr(),
                            item_features=csr_matrix(item_feat_lightfm.values).tocsr(),
                            num_threads=10)
```

In [36]:

```
# добавляем наш полученный скор в трейн датафрейм
data_train_filtered['score'] = predictions
```

In [37]:

```
# создаем предикт датафрейм в формате списка това
predict_result = data_train_filtered[['user_id','item_id','score']][data_train_filtered.ite
            unique().reset_index()
```

In [38]:

```
# объединяем предикт и тест датасет для подсчета precision
df_result_for_metrics = result.merge(predict_result, on='user_id', how='inner')
```

In [39]:

```
df_result_for_metrics.head()
```

Out[39]:

	user_id	actual	item_id
0	1	[821867, 834484, 856942, 865456, 889248, 90795...	[1029743, 857006, 6034857, 952163, 1004906, 82...
1	3	[835476, 851057, 872021, 878302, 879948, 90963...	[1106523, 866211, 899624, 965267, 1127831, 558...
2	6	[920308, 926804, 946489, 1006718, 1017061, 107...	[1070820, 1029743, 1126899, 866211, 878996, 11...
3	7	[840386, 889774, 898068, 909714, 929067, 95347...	[1029743, 1126899, 1106523, 866211, 878996, 11...
4	8	[835098, 872137, 910439, 924610, 992977, 10412...	[1106523, 1070820, 1029743, 878996, 916122, 60...

Test with custom precision func

In [40]:

```
precision = df_result_for_metrics.apply(lambda row: custom_precision(row['item_id'], row['a
print(f"Precision: {precision}")
```

```
Precision: 0.1386535303776683
```

Links

Neural networks for RS: http://d2l.ai/chapter_recommender-systems/mf.html (http://d2l.ai/chapter_recommender-systems/mf.html)

LigthFM -> https://arxiv.org/pdf/1507.08439.pdf (https://arxiv.org/pdf/1507.08439.pdf)

https://making.lyst.com/lightfm/docs/home.html (https://making.lyst.com/lightfm/docs/home.html)

Домашнее задание

1) Прочитать статьи про BPR, WARP loss

2) Сделать грид серч текущей модели, смотрите на метрику precision@5, считаем на тесте нашей функцией