Рекомендательные системы в продажах

In [ ]:
```python
# В интернет-магазине есть данные по пользовательским сессиям, каждая из которых содержит список
# просмотренных товаров (id) и список купленных товаров.
# Необходимо на основе этих данных создать рекомендательную систему, предлагающую пользователю
# k - товаров на основе его просмотров
```

In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
with open('Data/Coursera/coursera_sessions_train.txt', 'r') as f:
    sess_train = f.read().splitlines()
with open('Data/Coursera/coursera_sessions_test.txt', 'r') as f:
    sess_test = f.read().splitlines()
```

In [5]:
```python
# представленные данные: каждая строка - список просмотренных товаров (id) пользователем за одну сессию, а через ";" - купленный
sess_train[8:12]
```

Out[5]:
```
['71,72,73,74;',
 '76,77,78;',
 '84,85,86,87,88,89,84,90,91,92,93,86;86',
 '114,77,115,116,117,118,119,120,121,120,122,123,124;']
```

In [ ]:
```python
# Проект построен следующим образом

# 1. Создаём матрицы частот просматриваемых и покупаемых товаров
# (товар - кол-во просмотров и товар - кол-во покупок), сортируем м-цы по убыванию частот.
# Т.о., получили м-цу популярности просмотров и м-цу популярности покупок

# 2. Для текущей сессии просмотров [id_1,id_2,...,id_n] делаем рекомендации
# на k товаров из этого списка (k<=n), стоящие в м-це популярности ПРОСМОТРОВ выше оставшихся n-k просмотров

# 3. Для текущей сессии просмотров [id_1,id_2,...,id_n] делаем рекомендации
# на k товаров из этого списка (k<=n), стоящие в м-це популярности ПОКУПОК выше оставшихся n-k просмотров

# 4. - 6. Проверяем все то же самое на тестовой выборке (используя, разумеется, м-цы популярности тренировочной выборки)
```

In [ ]:
```python
# В итоге, значения метрик для рекомендаций по покупкам ведут себя на тестовой выборке немного лучше,
# при этом значения метрик для тестовой выборке очень близки к значениям метрик по тренировочной выборке
# для рекомендаций по просмотрам.
```

1. На обучении постройте частоты появления id в просмотренных и в купленных

In [ ]:
```python
#(id может несколько раз появляться в просмотренных, все появления надо учитывать)
```

In [3]:
```python
# разделяем м-цу сессий на просмотренные id и купленные id
# для обучающей выборки
sess_train_lp = []
for sess in sess_train:
    look_items, pur_items = sess.split(';')
    # применяем ф-цию int() ко всем id сессии
    look_items = list(map(int, look_items.split(',')))
    if len(pur_items) > 0:
        pur_items = list(map(int, pur_items.split(',')))
    else:
        pur_items = []
    sess_train_lp.append([look_items, pur_items])

# для тестовой выборки
sess_test_lp = []
for sess in sess_test:
    look_items, pur_items = sess.split(';')
    look_items = list(map(int, look_items.split(',')))
    if len(pur_items) > 0:
        pur_items = list(map(int, pur_items.split(',')))
    else:
        pur_items = []
    sess_test_lp.append([look_items, pur_items])
```

In [10]:
```python
sess_train_lp[8:12]
```

Out[10]:
```
[[[71, 72, 73, 74], []],
 [[76, 77, 78], []],
 [[84, 85, 86, 87, 88, 89, 84, 90, 91, 92, 93, 86], [86]],
 [[114, 77, 115, 116, 117, 118, 119, 120, 121, 120, 122, 123, 124], []]]
```

In [4]:
```python
# ДЛЯ ПРОСМОТРЕННЫХ

# сначала все просмотренные id загоняем в список
sess_train_lid = []
for i in range(len(sess_train_lp)):
    for id in sess_train_lp[i][0]:
        sess_train_lid.append(id)

# теперь столбец уникальных id со счетчиком
sess_train_l_cnt = np.transpose(np.unique(sess_train_lid, return_counts=True))
```

In [22]:
```python
# структура столбца: i-й элемент является двумерной строкой [id, кол-во раз которое он встречается]
sess_train_l_cnt
```

Out[22]:
```
array([[     0,      6],
       [     1,      6],
       [     2,      9],
       ...,
       [102804,      1],
       [102805,      1],
       [102806,      1]], dtype=int64)
```

In [5]:
```python
# ДЛЯ КУПЛЕННЫХ

# сначала все просмотренные id загоняем в список
sess_train_pid = []
for i in range(len(sess_train_lp)):
    for id in sess_train_lp[i][1]:
        sess_train_pid.append(id)

# теперь столбец уникальных id со счетчиком
sess_train_p_cnt = np.transpose(np.unique(sess_train_pid, return_counts=True))
```

In [24]:
```python
sess_train_p_cnt
```

Out[24]:
```
array([[     5,      1],
       [     6,      2],
       [     7,      2],
       ...,
       [102417,      1],
       [102462,      1],
       [102646,      1]], dtype=int64)
```

In [6]:
```python
# Сортируем полученные столбцы по счетчику (запись по убыванию)
# sess_train_l_cnt[:,1].argsort() - список индексов строк нашего двумерного массива по возрастанию значения счетчика
# sess_train_l_cnt[sess_train_l_cnt[:,1].argsort()] - отсортированный в порядке возрастания счетчика массив
# операция среза - list[<start>:<stop>:<step>], [::-1] - просто вернет список с конца до начала
sess_train_l_cnt = sess_train_l_cnt[sess_train_l_cnt[:,1].argsort()][::-1]
sess_train_p_cnt = sess_train_p_cnt[sess_train_p_cnt[:,1].argsort()][::-1]
```

In [40]:
```python
# итоговая м-ца купленных товаров: id товара - сколько раз он был куплен
sess_train_p_cnt
```

Out[40]:
```
array([[  158,    14],
       [  204,    12],
       [   73,    11],
       ...,
       [38189,     1],
       [38177,     1],
       [    5,     1]], dtype=int64)
```

2. Алгоритм рекомендаций - сортировка просмотренных id по популярности (частота появления в просмотренных)

In [7]:
```python
# ф-ция расчета метрик 'Precision@k' и 'Recall@k' для k-рекомендаций
# 'reccomendations' - список 'k' рекомендуемых товаров для текущей сессии, в которой был совершен список покупок 'session'
def prec_rec_metrics(session, reccomendations, k):
    purchase = 0
    for ind in reccomendations:
        if ind in session:
            purchase += 1
    precision = purchase / k
    recall = purchase / len(session)
    return(precision, recall)
```

In [8]:
```python
# столбцы купленных и просмотренных товаров
sess_train_p = [row[1] for row in sess_train_lp]
sess_train_l = [row[0] for row in sess_train_lp]
```

In [9]:
```python
%%time
# Считаем метрики 'Precision@k' и 'Recall@k' на обучающей выборке для k=1 и k=5
prec_at_1_tr_l, rec_at_1_tr_l = [], []
prec_at_5_tr_l, rec_at_5_tr_l = [], []
k1, k5 = 1, 5
for i, sess_p in enumerate(sess_train_p):
    # не обрабатываем сессии без покупок
    if sess_p == []: continue

    # просмотренные id для сессии i
    sess_l = sess_train_l[i]

    l_ind_sess = []
    for j in range(len(sess_l)):
        # выбираем индекс (!) столбца-счетчика встречаемости просмотренных id,
        # элемент в котором соответствует значению текущего id сессии.
        l_ind_sess.append(np.where(sess_train_l_cnt[:,0] == sess_l[j])[0][0])
    # список индексов м-цы-счетчика купленных товаров для купленных товаров из текущей сессии
    l_ind_sess_sorted = np.unique(l_ind_sess)

    # k1 рекомендаций
    num_of_recs_k1 = min(k1, len(sess_l))
    if num_of_recs_k1 == 0: continue
    # берем первые Min(k1,кол-во id в текущей сессии) кол-во id из текущей сессии,
    # согласно сортировке по частоте просмотров по всей обучающей выборке
    recs_k1 = sess_train_l_cnt[l_ind_sess_sorted[:num_of_recs_k1],0]

    # расчет k1-метрик
    prec_1, rec_1 = prec_rec_metrics(sess_p, recs_k1, k1)
    prec_at_1_tr_l.append(prec_1)
    rec_at_1_tr_l.append(rec_1)

    # k5 рекомендаций
    num_of_recs_k5 = min(k5, len(sess_l))
    if num_of_recs_k5 == 0: continue
    recs_k5 = sess_train_l_cnt[l_ind_sess_sorted[:num_of_recs_k5],0]

    # расчет k5-метрик
    prec_5, rec_5 = prec_rec_metrics(sess_p, recs_k5, k5)
    prec_at_5_tr_l.append(prec_5)
    rec_at_5_tr_l.append(rec_5)
```

```
Wall time: 8.23 s
```

In [10]:
```python
avg_prec_at_1_tr_l = np.mean(prec_at_1_tr_l)
avg_rec_at_1_tr_l = np.mean(rec_at_1_tr_l)
avg_prec_at_5_tr_l = np.mean(prec_at_5_tr_l)
avg_rec_at_5_tr_l = np.mean(rec_at_5_tr_l)
```

In [13]:
```python
r1 = round(avg_rec_at_1_tr_l, 2)
p1 = round(avg_prec_at_1_tr_l, 2)
r5 = round(avg_rec_at_5_tr_l, 2)
p5 = round(avg_prec_at_5_tr_l, 2)
```

In [24]:
```python
# будем заносить все результаты этого проекта в единый датафрейм 'metrics'
metrics = pd.DataFrame({'precision':p1, 'recall': r1},index={'Train_look_k=1'})
metrics.loc['Train_look_k=5'] = [p5,r5]
```

3. Алгоритм рекомендаций - сортировка просмотренных id по покупаемости (частота появления в покупках)

In [25]:
```python
%%time
# Считаем метрики 'Precision@k' и 'Recall@k' на обучающей выборке для k=1 и k=5
prec_at_1_tr_p, rec_at_1_tr_p = [], []
prec_at_5_tr_p, rec_at_5_tr_p = [], []
k1, k5 = 1, 5

for i, sess_p in enumerate(sess_train_p):
    # не обрабатываем сессии без покупок
    if sess_p == []: continue

    # просмотренные id для сессии i
    sess_l = sess_train_l[i]

    l_ind_sess = []
    for j in range(len(sess_l)):
        # выбираем индекс столбца-счетчика встречаемости просмотренных id,
        # элемент в котором соответствует значению текущего id сессии
        # здесь первый [0] выбирает первую строку в найденном однострочном элементе,
        # второй [0] выбирает первый элемент строки, являющимся как раз индексом
        if sess_l[j] not in sess_train_p_cnt[:,0]: continue
        l_ind_sess.append(np.where(sess_train_p_cnt[:,0] == sess_l[j])[0][0])
    l_ind_sess_sorted = np.unique(l_ind_sess)

    # k1 рекомендаций
    num_of_recs_k1 = min(k1, len(sess_l), len(l_ind_sess_sorted))
    if num_of_recs_k1 == 0: continue
    # берем первые Min(k1,кол-во id в текущей сессии) кол-во id из текущей сессии,
    # согласно сортировке по частоте просмотров по всей обучающей выборке
    recs_k1 = sess_train_p_cnt[l_ind_sess_sorted[:num_of_recs_k1],0]

    # расчет k1-метрик
    prec_1, rec_1 = prec_rec_metrics(sess_p, recs_k1, k1)
    prec_at_1_tr_p.append(prec_1)
    rec_at_1_tr_p.append(rec_1)

    # k5 рекомендаций
    num_of_recs_k5 = min(k5, len(sess_l), len(l_ind_sess_sorted))
    if num_of_recs_k5 == 0: continue
    recs_k5 = sess_train_p_cnt[l_ind_sess_sorted[:num_of_recs_k5],0]

    # расчет k5-метрик
    prec_5, rec_5 = prec_rec_metrics(sess_p, recs_k5, k5)
    prec_at_5_tr_p.append(prec_5)
    rec_at_5_tr_p.append(rec_5)
```

```
Wall time: 1.74 s
```

In [26]:
```python
avg_prec_at_1_tr_p = np.mean(prec_at_1_tr_p)
avg_rec_at_1_tr_p = np.mean(rec_at_1_tr_p)
avg_prec_at_5_tr_p = np.mean(prec_at_5_tr_p)
avg_rec_at_5_tr_p = np.mean(rec_at_5_tr_p)

r1 = round(avg_rec_at_1_tr_p, 2)
p1 = round(avg_prec_at_1_tr_p, 2)
r5 = round(avg_rec_at_5_tr_p, 2)
p5 = round(avg_prec_at_5_tr_p, 2)
```

In [27]:
```python
metrics.loc['Train_purch_k=1'] = [p1,r1]
metrics.loc['Train_purch_k=5'] = [p5,r5]
```

4. На тестовой выборке построим частоты появления id в просмотренных и в купленных

In [53]:
```python
# Формируем столбец просмотренных товаров за все тренировочные сессии
sess_test_l = [row[0] for row in sess_test_lp]
sess_test_l_np = []
for sess in sess_test_l:
    for idd in sess:
        sess_test_l_np.append(idd)
sess_test_l_np = np.array(sess_test_l_np)

# Формируем столбец купленных товаров за все тренировочные сессии
sess_test_p = [row[1] for row in sess_test_lp]
sess_test_p_np = []
for sess in sess_test_p:
    for idd in sess:
        sess_test_p_np.append(idd)
sess_test_p_np = np.array(sess_test_p_np)
```

5. Алгоритм рекомендаций на тестовой выборке - сортировка просмотренных id по популярности (частота появления в просмотренных)

In [57]:
```python
%%time
# метрики по тестовым данным
prec_at_1_tst_l, rec_at_1_tst_l = [], []
prec_at_5_tst_l, rec_at_5_tst_l = [], []
k1, k5 = 1, 5

for i, sess_p in enumerate(sess_test_p):
    # пропускаем сессии без покупок
    if sess_p == []: continue

    # проходимся по id из просмотренных товаров
    sess_l = sess_test_l[i]

    # выбираем индекс столбца-счетчика встречаемости просмотренных id в тренировочной приоритетной матрице
    l_ind_sess = []
    new_ids = []
    for j in range(len(sess_l)):
        if sess_l[j] not in sess_train_l_cnt[:,0]:
            new_ids.append(sess_l[j])
            continue
        l_ind_sess.append(np.where(sess_train_l_cnt[:,0] == sess_l[j])[0][0])
    l_ind_sess_sorted = np.unique(l_ind_sess)

    # k1 рекомендации
    num_of_recs_k1 = min(k1, len(sess_l))
    if num_of_recs_k1 == 0: continue
    # добавляем это условие, поскольку все просмотренные в сессии товары могут не быть в тренировочной м-це
    if l_ind_sess != []:
        recs_k1 = sess_train_l_cnt[l_ind_sess_sorted[:num_of_recs_k1],0]
    else:
        recs_k1 = []
    # здесь объединяем отсортированные 'в порядке убывания приоритета приоритетной м-цы тренировочных данных'
    # рекомендуемые товары и новые товары, не присутствующие ранее в приоритетной м-це тренировочных данных
    recs_k1 = np.concatenate((np.array(recs_k1, dtype='int64'), np.unique(np.array(new_ids, dtype='int64'))))[:num_of_recs_k1]

    # k1 метрики
    prec_1, rec_1 = prec_rec_metrics(sess_p, recs_k1, k1)
    prec_at_1_tst_l.append(prec_1)
    rec_at_1_tst_l.append(rec_1)

    # k5 рекомендации
    num_of_recs_k5 = min(k5, len(sess_l))
    if num_of_recs_k5 == 0: continue
    if l_ind_sess != []:
        recs_k5 = sess_train_l_cnt[l_ind_sess_sorted[:num_of_recs_k5],0]
    else:
        recs_k5 = []
    recs_k5 = np.concatenate((np.array(recs_k5, dtype='int64'), np.unique(np.array(new_ids, dtype='int64'))))[:num_of_recs_k5]

    # k5 метрики
    prec_5, rec_5 = prec_rec_metrics(sess_p, recs_k5, k5)
    prec_at_5_tst_l.append(prec_5)
    rec_at_5_tst_l.append(rec_5)
```

```
Wall time: 12.9 s
```

In [59]:
```python
avg_prec_at_1_tst_l = np.mean(prec_at_1_tst_l)
avg_rec_at_1_tst_l = np.mean(rec_at_1_tst_l)
avg_prec_at_5_tst_l = np.mean(prec_at_5_tst_l)
avg_rec_at_5_tst_l = np.mean(rec_at_5_tst_l)

r1 = round(avg_rec_at_1_tst_l, 2)
p1 = round(avg_prec_at_1_tst_l, 2)
r5 = round(avg_rec_at_5_tst_l, 2)
p5 = round(avg_prec_at_5_tst_l, 2)
```

In [60]:
```python
metrics.loc['Test_look_k=1'] = [p1,r1]
metrics.loc['Test_look_k=5'] = [p5,r5]
```

6. Алгоритм рекомендаций на тестовой выборке - сортировка просмотренных id по покупаемости (частота появления в покупках)

In [84]:
```python
prec_at_1_tst_p, rec_at_1_tst_p = [], []
prec_at_5_tst_p, rec_at_5_tst_p = [], []
k1, k5 = 1, 5
for i, sess_p in enumerate(sess_test_p):

    if sess_p == []: continue


    sess_l = sess_test_l[i]


    l_ind_sess = []
    new_ids = []
    for j in range(len(sess_l)):
        if sess_l[j] not in sess_train_p_cnt[:,0]:
            new_ids.append(sess_l[j])
            continue
        l_ind_sess.append(np.where(sess_train_p_cnt[:,0] == sess_l[j])[0][0])
    l_ind_sess_sorted = np.unique(l_ind_sess)

    # k1 recommendations
    num_of_recs_k1 = min(k1, len(sess_l))
    if num_of_recs_k1 == 0: continue
    if l_ind_sess != []:
        recs_k1 = sess_train_p_cnt[l_ind_sess_sorted[:num_of_recs_k1],0]
    else:
        recs_k1 = []
    # химичим тут
    recs_k1 = np.concatenate((np.array(recs_k1, dtype='int64'), np.unique(np.array(new_ids, dtype='int64'))))[:num_of_recs_k1]

    # k1 метрики
    prec_1, rec_1 = prec_rec_metrics(sess_p, recs_k1, k1)
    prec_at_1_tst_p.append(prec_1)
    rec_at_1_tst_p.append(rec_1)

    # k5 рекомендации
    num_of_recs_k5 = min(k5, len(sess_l))
    if num_of_recs_k5 == 0: continue
    if l_ind_sess != []:
        recs_k5 = sess_train_p_cnt[l_ind_sess_sorted[:num_of_recs_k5],0]
    else:
        recs_k5 = []
    recs_k5 = np.concatenate((np.array(recs_k5, dtype='int64'), np.unique(np.array(new_ids, dtype='int64'))))[:num_of_recs_k5]

    # k5 метрики
    prec_5, rec_5 = prec_rec_metrics(sess_p, recs_k5, k5)
    prec_at_5_tst_p.append(prec_5)
    rec_at_5_tst_p.append(rec_5)
```

In [85]:
```python
avg_prec_at_1_tst_p = np.mean(prec_at_1_tst_p)
avg_rec_at_1_tst_p = np.mean(rec_at_1_tst_p)
avg_prec_at_5_tst_p = np.mean(prec_at_5_tst_p)
avg_rec_at_5_tst_p = np.mean(rec_at_5_tst_p)

r1 = round(avg_rec_at_1_tst_p, 2)
p1 = round(avg_prec_at_1_tst_p, 2)
r5 = round(avg_rec_at_5_tst_p, 2)
p5 = round(avg_prec_at_5_tst_p, 2)
```

```
Answer 4: 0.42 0.49 0.80 0.20
```

In [86]:
```python
metrics.loc['Test_purch_k=1'] = [p1,r1]
metrics.loc['Test_purch_k=5'] = [p5,r5]
```

7. Анализ результатов

In [87]:
```python
metrics
```

Out[87]:

	precision	recall
Train_look_k=1	0.51	0.44
Train_look_k=5	0.21	0.83
Train_purch_k=1	0.79	0.68
Train_purch_k=5	0.25	0.93
Test_look_k=1	0.48	0.42
Test_look_k=5	0.2	0.8
Test_purch_k=1	0.49	0.42
Test_purch_k=5	0.20	0.80

In [93]:
```python
metrics.loc[['Train_look_k=1','Test_purch_k=1']]
```

Out[93]:

	precision	recall
Train_look_k=1	0.51	0.44
Test_purch_k=1	0.49	0.42

In [94]:
```python
metrics.loc[['Train_look_k=5','Test_purch_k=5']]
```

Out[94]:

	precision	recall
Train_look_k=5	0.21	0.83
Test_purch_k=5	0.20	0.80

In [95]:
```python
metrics.loc[['Train_look_k=1','Test_look_k=1']]
```

Out[95]:

	precision	recall
Train_look_k=1	0.51	0.44
Test_look_k=1	0.48	0.42

In [96]:
```python
metrics.loc[['Train_look_k=5','Test_look_k=5']]
```

Out[96]:

	precision	recall
Train_look_k=5	0.21	0.83
Test_look_k=5	0.2	0.8

In [97]:
```python
metrics.loc[['Train_purch_k=1','Test_purch_k=1']]
```

Out[97]:

	precision	recall
Train_purch_k=1	0.79	0.68
Test_purch_k=1	0.49	0.42

In [98]:
```python
metrics.loc[['Train_purch_k=5','Test_purch_k=5']]
```

Out[98]:

	precision	recall
Train_purch_k=5	0.25	0.93
Test_purch_k=5	0.20	0.80

In [ ]: