Лабораторная работа №4. Создание рекомендательной модели

Студент: Кривцов Н.А.
Группа: ИУ5-22М

Импорт библиотек

In [389]:

```
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances, manhattan_distances
```

Загрузка и просмотр датасетов

In [30]:

```
artists_df = pd.read_csv('/content/drive/MyDrive/MMO/ЛР4/artists.dat', sep='\t')
artists_df = artists_df[['id', 'name']]
artists_df.index = artists_df.id
```

In [75]:

```
artists_df
```

Out[75]:

	id	name
id		
1	1	MALICE MIZER
2	2	Diary of Dreams
3	3	Carpathian Forest
4	4	Moi dix Mois
5	5	Bella Morte
...	...	...
18741	18741	Diamanda Galás
18742	18742	Aya RL
18743	18743	Coptic Rain
18744	18744	Oz Alchemist
18745	18745	Grzegorz Tomczak

17632 rows × 2 columns

In [4]:

```
users_df = pd.read_csv('/content/drive/MyDrive/MMO/ЛР4/user_artists.dat', sep='\t')
```

In [5]:

```
users_df
```

Out[5]:

	userID	artistID	weight
0	2	51	13883
1	2	52	11690
2	2	53	11351
3	2	54	10300
4	2	55	8983
...	...	...	...
92829	2100	18726	337
92830	2100	18727	297
92831	2100	18728	281
92832	2100	18729	280
92833	2100	18730	263

92834 rows × 3 columns

In [16]:

```
tags_df = pd.read_csv('/content/drive/MyDrive/MMO/JIP4/tags.dat', sep='\t')
tags_df.index = tags_df.tagID
```

In [17]:

```
tags_df
```

Out[17]:

	tagID	tagValue
tagID		
1	1	metal
2	2	alternative metal
3	3	goth rock
4	4	black metal
5	5	death metal
...	...	...
12644	12644	suomi
12645	12645	symbiosis
12646	12646	sverige
12647	12647	eire
12648	12648	electro latino

11946 rows × 2 columns

In [63]:

```
artists_tags_df = pd.read_csv('/content/drive/MyDrive/MMO/JIP4/user_taggedartists.dat', sep='\t')
artists_tags_df = artists_tags_df[['artistID', 'tagID']]
artists_tags_df = artists_tags_df.join(tags_df, on='tagID', lsuffix='_l', rsuffix='_r')
artists_tags_df = artists_tags_df[['artistID', 'tagValue']]
# artists_tags_df = artists_tags_df.join(artists_df, on='artistID', lsuffix='_l', rsuffix='_ir')
# artists_tags_df = artists_tags_df[['id', 'name', 'tagValue']]
```

In [64]:

```
artists_tags_df
```

Out[64]:

	artistID	tagValue
0	52	chillout
1	52	downtempo
2	52	electronic
3	52	trip-hop
4	52	female vovalists
...	...	...
186474	16437	black metal
186475	16437	folk
186476	16437	depressive black metal
186477	16437	dark folk
186478	16437	atmospheric black metal

186479 rows × 2 columns

Collaborative

In [16]:

```
users_df_scaled = users_df.assign(weight=users_df.groupby('userID')['weight'].transform(lambda x: (x -
x.min()) / (x.max()- x.min())))
users_df_scaled
```

Out[16]:

	userID	artistID	weight
0	2	51	1.000000
1	2	52	0.825509
2	2	53	0.798536
3	2	54	0.714911
4	2	55	0.610121
...	...	...	...
92829	2100	18726	0.060623
92830	2100	18727	0.038376
92831	2100	18728	0.029477
92832	2100	18729	0.028921
92833	2100	18730	0.019466

92834 rows × 3 columns

In [17]:

```
def create_utility_matrix(data):
    itemField = 'artistID'
    userField = 'userID'
    valueField = 'weight'

    userList = data[userField].tolist()
    itemList = data[itemField].tolist()
    valueList = data[valueField].tolist()

    users = list(set(userList))
    items = list(set(itemList))
```

```python
    users_index = {users[i]: i for i in range(len(users))}
    pd_dict = {item: [0.0 for i in range(len(users))] for item in items}

    for i in range(0,data.shape[0]):
        item = itemList[i]
        user = userList[i]
        value = valueList[i]
        pd_dict[item][users_index[user]] = value

    X = pd.DataFrame(pd_dict)
    X.index = users

    itemcols = list(X.columns)
    items_index = {itemcols[i]: i for i in range(len(itemcols))}

    return X, users_index, items_index
```

In [18]:

```python
%%time
user_item_matrix, users_index, items_index = create_utility_matrix(users_df_scaled)
```

```
CPU times: user 7.74 s, sys: 906 ms, total: 8.65 s
Wall time: 8.44 s
```

In [37]:

```python
user_item_matrix.dropna(inplace=True)
user_item_matrix
```

Out[37]:

	1	2	3	4	5	6	7	8	9	10	...	18736	18737	18738	18739	18740	18741	18742	18743	1874
2	0.0	0.0	0.000000	0.0	0.0	0.000000	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	0.0	0.0	0.000000	0.0	0.0	0.000000	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.
4	0.0	0.0	0.000000	0.0	0.0	0.000000	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.
5	0.0	0.0	0.000000	0.0	0.0	0.000000	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.
6	0.0	0.0	0.000000	0.0	0.0	0.000000	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.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
2095	0.0	0.0	0.000000	0.0	0.0	0.000000	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.
2096	0.0	0.0	0.000000	0.0	0.0	0.000000	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.
2097	0.0	0.0	0.000000	0.0	0.0	0.000000	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.
2099	0.0	0.0	0.000000	0.0	0.0	0.000000	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.
2100	0.0	0.0	0.100111	0.0	0.0	0.097887	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.

1881 rows × 17632 columns

In [38]:

```python
user_item_matrix__test = user_item_matrix.iloc[[-1]]
user_item_matrix__test
```

Out[38]:

	1	2	3	4	5	6	7	8	9	10	...	18736	18737	18738	18739	18740	18741	18742	18743	1874
2100	0.0	0.0	0.100111	0.0	0.0	0.097887	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.

1 rows × 17632 columns

In [39]:

```
# Оставшаяся часть матрицы для обучения
user_item_matrix__train = user_item_matrix.iloc[:-1]
user_item_matrix__train
```

Out[39]:

	1	2	3	4	5	6	7	8	9	10	...	18736	18737	18738	18739	18740	18741	18742	18743	18744	18745
2	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	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	0.0	0.0	0.0	0.0	0.0
4	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	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	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2094	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	0.0	0.0	0.0	0.0	0.0
2095	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	0.0	0.0	0.0	0.0	0.0
2096	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	0.0	0.0	0.0	0.0	0.0
2097	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	0.0	0.0	0.0	0.0	0.0
2099	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	0.0	0.0	0.0	0.0	0.0

1880 rows × 17632 columns

In [40]:

```
%%time
U, S, VT = np.linalg.svd(user_item_matrix__train.T)
V = VT.T
```

```
CPU times: user 4min 9s, sys: 11.6 s, total: 4min 21s
Wall time: 2min 27s
```

In [41]:

```
# Матрица соотношения между пользователями и латентными факторами
U.shape
```

Out[41]:

```
(17632, 17632)
```

In [42]:

```
# Матрица соотношения между объектами и латентными факторами
V.shape
```

Out[42]:

```
(1880, 1880)
```

In [43]:

```
S.shape
```

Out[43]:

```
(1880,)
```

In [44]:

```
Sigma = np.diag(S)
Sigma.shape
```

Out[44]:

```
(1880, 1880)
```

In [45]:

```
# Диагональная матрица сингулярных значений
Sigma
```

Out[45]:

```
array([[1.56473318e+01, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 1.23922786e+01, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 9.95488403e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        3.93997751e-03, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 3.43405207e-03, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 2.22978369e-03]])
```

In [46]:

```
# Используем 3 первых сингулярных значения
r=3
Ur = U[:, :r]
Sr = Sigma[:r, :r]
Vr = V[:, :r]
```

In [47]:

```
# Матрица соотношения между новым пользователем и латентными факторами
test_user = np.mat(user_item_matrix__test.values)
test_user.shape, test_user
```

Out[47]:

```
((1, 17632),
 matrix([[0.        , 0.        , 0.10011123, ..., 0.        , 0.        ,
          0.        ]]))
```

In [48]:

```
tmp = test_user * Ur * np.linalg.inv(Sr)
tmp
```

Out[48]:

```
matrix([[-0.00025344,  0.00224003,  0.0007716 ]])
```

In [49]:

```
test_user_result = np.array([tmp[0,0], tmp[0,1], tmp[0,2]])
test_user_result
```

Out[49]:

```
array([-0.00025344,  0.00224003,  0.0007716 ])
```

In [53]:

```
# Вычисляем косинусную близость между текущим пользователем
# и остальными пользователями
cos_sim = cosine_similarity(Vr, test_user_result.reshape(1, -1))
cos_sim[:10]
```

Out[53]:

```
array([[ 0.10311833],
       [ 0.9946266 ],
       [-0.00734549],
       [ 0.99792168],
       [ 0.58052523],
       [-0.13404812],
       [-0.14614659],
       [ 0.67328805],
       [ 0.99698873],
       [-0.10375559]])
```

In [54]:

```
# Преобразуем размерность массива
cos_sim_list = cos_sim.reshape(-1, cos_sim.shape[0])[0]
cos_sim_list[:10]
```

Out[54]:

```
array([ 0.10311833,  0.9946266 , -0.00734549,  0.99792168,  0.58052523,
       -0.13404812, -0.14614659,  0.67328805,  0.99698873, -0.10375559])
```

In [55]:

```
# Находим наиболее близкого пользователя
recommended_user_id = np.argsort(-cos_sim_list)[0]
recommended_user_id
```

Out[55]:

```
416
```

In [76]:

```
# Получение исполнителя
artistID_list = list(user_item_matrix.columns)
def artist_name_by_artistID(ind):
    artistID = artistID_list[ind]
        # flt_links = users_df[users_df['artistID'] == artistID]
        # tmdbId = int(flt_links['tmdbId'].values[0])
        # md_links = df_md[df_md['id'] == tmdbId]
        # res = artists_df['name'].values[0]
        # return res
    res = artists_df[artists_df['id'] == artistID]
    return res.values[0][1]
```

In [77]:

```
artist_name_by_artistID(6)
```

Out[77]:

```
'Marilyn Manson'
```

In [104]:

```
i=1
for idx, item in enumerate(np.ndarray.flatten(np.array(test_user))):
    if item > 0:
        artist_name = artist_name_by_artistID(idx)
```

```
        print('{} - {} - {}'.format(idx, artist_name, item))
        # if i==20:
        #     break
        # else:
        #     i+=1
```

```
2 - Carpathian Forest - 0.10011123470522804
5 - Moonspell - 0.09788654060066741
11 - Behemoth - 0.5700778642936596
26 - Gorgoroth - 0.37986651835372637
41 - Emperor - 0.04282536151279199
828 - Eluveitie - 0.29699666295884314
832 - Slayer - 0.1807563959955506
1100 - Yann Tiersen - 0.6145717463848721
1102 - Tenhi - 0.4638487208008899
1125 - Agalloch - 0.5478309232480534
1251 - Marduk - 0.6846496106785317
1267 - Immortal - 0.503337041156841
1272 - Cannibal Corpse - 0.19187986651835373
2729 - Deathspell Omega - 0.16907675194660735
2745 - Vader - 0.12736373748609567
2752 - Blut aus Nord - 0.14293659621802002
2754 - Dark Funeral - 0.11568409343715239
3730 - Rotting Christ - 0.08954393770856507
4098 - ColdWorld - 0.01668520578420467
4187 - Burzum - 1.0
4517 - Alcest - 0.07341490545050056
4860 - Drudkh - 0.21468298109010012
6132 - Summoning - 0.09733036707452725
6516 - Lifelover - 0.27975528364849833
7735 - Nokturnal Mortum - 0.0339265850945495
8127 - Dominia - 0.03114571746384872
8129 - Amžius - 0.2347052280311457
8130 - Obtest - 0.12680756395995552
8131 - Nahash - 0.4671857619577308
8133 - Satanic Warmaster - 0.22135706340378197
8134 - Altorių Šešėliai - 0.21412680756395996
8139 - Luctus - 0.2374860956618465
8151 - Anubi - 0.22914349276974416
8324 - Dissimulation - 0.002224694104560623
8328 - Skepticism - 0.1117908787541713
8330 - Argharus - 0.21078976640711902
8332 - Shape of Despair - 0.27586206896551724
9548 - The Kilimanjaro Darkjazz Ensemble - 0.314238042269188
10573 - Peste Noire - 0.26529477196885426
13134 - Moëvöt - 0.027808676307007785
13136 - Celestia - 0.06562847608453838
13404 - Vilkduja - 0.1707452725250278
15609 - Les Discrets - 0.11957730812013348
17614 - Mortifera - 0.29477196885428253
17615 - Nyktalgia - 0.060622914349276975
17616 - Atsakau  niekadA - 0.03837597330367074
17617 - Domantas Razauskas - 0.029477196885428252
17618 - Atalyja - 0.028921023359288096
17619 - Les Chants de Nihil - 0.01946607341490545
```

In [108]:

```
i=1
recommended_user_item_matrix = user_item_matrix.iloc[[recommended_user_id]]
for idx, item in enumerate(np.ndarray.flatten(np.array(recommended_user_item_matrix))):
    if item > 0:
        artist_name = artist_name_by_artistID(idx)
        print('{} - {} - {}'.format(idx, artist_name, item))
        # if i==20:
        #     break
        # else:
        #     i+=1
```

```
107 - Dustin O'Halloran - 0.03896103896103896
114 - Deru - 0.170995670995671
132 - Library Tapes - 0.14155844155844155
142 - The Boats - 0.007792207792207792
```

```
148 - Radiohead - 0.03333333333333333
188 - The Tiger Lillies - 0.06796536796536796
232 - Massive Attack - 0.01774891774891775
412 - Sigur Rós - 0.08008658008658008
738 - Carbon Based Lifeforms - 0.0735930735930736
773 - Solar Fields - 0.25627705627705627
1727 - Balmorhea - 0.10476190476190476
1928 - Flying Lotus - 0.015151515151515152
2585 - Bonobo - 0.2354978354978355
3477 - Biosphere - 0.09783549783549783
3480 - Zoviet France - 0.15670995670995672
3491 - Alva Noto - 0.06147186147186147
3786 - Nino Katamadze & Insight - 0.06233766233766234
4932 - Lusine - 0.00735930735930736
6440 - Ulrich Schnauss - 0.16536796536796536
6617 - Helios - 0.35064935064935066
6835 - Between Interval - 0.003463203463203463
6858 - Magnitarus - 1.0
6859 - Blind Divine - 0.8588744588744589
6860 - Ab Ovo - 0.43982683982683984
6861 - Lusine ICL - 0.42597402597402595
6862 - Phaeleh - 0.2696969696969697
6863 - All India Radio - 0.2623376623376623
6864 - Bad Sector - 0.19090909090909092
6865 - H.U.V.A. Network - 0.16406926406926406
6866 - Hol Baumann - 0.15844155844155844
6867 - Silencide - 0.12164502164502164
6868 - ksandr and I.M.M.U.R.E. - 0.11471861471861472
6869 - Aes Dana - 0.10476190476190476
6870 - Noto - 0.09567099567099567
6871 - Oval - 0.08311688311688312
6872 - Jagjit Singh - 0.045887445887445889
6873 - Anúna - 0.02813852813852814
6874 - Nino Katamadze - 0.026406926406926406
6875 - Spiraal Aurel - 0.022943722943722943
6876 - R.D.Burman - 0.01948051948051948
6877 - Asura - 0.018614718614718615
6878 - Si - 0.015584415584415584
6879 - ksandr - 0.014285714285714285
6880 - Nina Karlsson - 0.011688311688311689
6881 - EugeneKha - 0.008658008658008658
6882 - Alva Noto + Ryuichi Sakamoto with Ensemble Modern - 0.008658008658008658
6883 - Orsten - 0.007792207792207792
6884 - Arbre Noir - 0.00735930735930736
6885 - Komet - 0.0004329004329004329
```

In [101]:

```
recommended_user_item_matrix
```

Out[101]:

	1	2	3	4	5	6	7	8	9	10	...	18736	18737	18738	18739	18740	18741	18742	18743	18744	18745
447	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	0.0	0.0	0.0	0.0	0.0

1 rows × 17632 columns

Content-based

In [65]:

```
artists_tags_df.head()
```

Out[65]:

	artistID	tagValue
0	52	chillout
1	52	downtempo

	artistID	tagValue
2	52	electronic
3	52	trip-hop
4	52	female vovalists

In [67]:

```
%%time
artists_tags_grouped = artists_tags_df.groupby(by='artistID')
```

```
CPU times: user 347 µs, sys: 10 µs, total: 357 µs
Wall time: 365 µs
```

In [68]:

```
artists_tags_grouped
```

Out[68]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f677fc3ec50>
```

In [69]:

```
%%time
artists_tag_sets = artists_tags_grouped['tagValue'].apply(set)
```

```
CPU times: user 216 ms, sys: 3.71 ms, total: 220 ms
Wall time: 225 ms
```

In [86]:

```
def tag_set_to_str(tags):
  return ", ".join([tag for tag in tags])
```

In [77]:

```
tag_sets_df = pd.DataFrame(artists_tag_sets)
tag_sets_df = tag_sets_df.join(artists_df)
tag_sets_df
```

Out[77]:

artistID	tagValue	id	name
1	{japanese, gothic, better than lady gaga, weea...	1.0	MALICE MIZER
2	{ambient, true goth emo, dark, vocal, seen liv...	2.0	Diary of Dreams
3	{norsk arysk metal, saxophones, true norwegian...	3.0	Carpathian Forest
4	{japanese, bazarov, gothic metal, gothic, visu...	4.0	Moi dix Mois
5	{covers, gothic, deathrock, gothic rock, darkw...	5.0	Bella Morte
...	...	...	...
18737	{trip beat, electronica, alternative, 80s, noise}	18737.0	Ciccone Youth
18739	{favorite, electronica, uk, alternative, rock,...	18739.0	Apollo 440
18740	{ebm, industrial}	18740.0	Die Krupps
18741	{experimental, dead music}	18741.0	Diamanda Galás
18744	{ambient, aphex twin, downtempo, chillout, dar...	18744.0	Oz Alchemist

12523 rows × 3 columns

In [94]:

```
%time
tag_sets_df['tags'] = tag_sets_df['tagValue'].apply(lambda x: tag_set_to_str(x))
```

```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 7.15 µs
```

In [97]:

```
tag_sets_df = tag_sets_df[['name', 'tags']]
```

In [314]:

```
tag_sets_df.index = range(0, tag_sets_df.shape[0])
```

In [371]:

```
tag_sets_df.loc[250:270]
```

Out[371]:

	name	tags
250	Rihanna	dance-pop, diva, 2000s diva, hit, american, su...
251	Britney Spears	dance-pop, songs to make florchuchizz cry, blo...
252	Jordin Sparks	eletropop, crazy, perfect, lembra alguem, amer...
253	Kelly Clarkson	fatty, diva, american, idols, break up, romant...
254	Christina Aguilera	crazy, dance-pop, diva, i love girls, beautifu...
255	Ashlee Simpson	best of 2005, beautiful voice, porcaria vician...
256	Leona Lewis	weekly top artists, hot, diva, vocally perfect...
257	Beyoncé	dance-pop, diva, american, girl shit, rnb, bea...
258	Sugababes	weekly top artists, 2007, siobhan, british, mu...
259	David Cook	male vocalists, 4m4zinq, excelent, american id...
260	Lily Allen	hot, 4m4zinq, british, tyler adam, cool, elect...
261	Jennifer Lopez	weekly top artists, dance-pop, beauty, perfect...
262	Katy Perry	eletropop, dance-pop, diva, the vampire diarie...
263	Alicia Keys	diva, charmed, r and b, cool, american, neo-so...
264	P!nk	weekly top artists, hot, 4m4zinq, fuck bush, t...
265	Panic at the Disco	male vocalist, pop, pop rock, alternative, rock
266	David Archuleta	american, addictive, rnb, american idol 7, dan...
267	Katharine McPhee	drive, american idol 5, american, idols, summe...
268	Black Eyed Peas	eletropop, crazy, dance-pop, jesse, american, ...
269	Kate Voegele	american, addictive, female vocalist, singer-s...
270	Kat DeLuna	female vocalists, female, dance, urban, sexy, ...

In [316]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [317]:

```
tags_raw = tag_sets_df['tags'].values
tags_raw[:5]
```

Out[317]:

```
array(['japanese, gothic, better than lady gaga, weeabo, visual kei, jrock, j-rock',
       'ambient, true goth emo, dark, vocal, seen live, gothic, gothic rock, industrial, darkwave, elec
tronic, german',
       'norsk arysk metal, saxophones, true norwegian black metal, black metal, norwegian black metal,
very kvlt',
       'japanese, bazarov, gothic metal, gothic, visual kei, rock, gothic japanese, metal, j-rock',
       'covers, gothic, deathrock, gothic rock, darkwave'], dtype=object)
```

In [318]:

```
tags_raw [294:300]
```

Out[318]:

```
array(["diva, songs to make florchuchizz cry, omg this is so good, american, summer song, romantic, avr
il, beautiful, music lembra quando eu thava 7 serie lol, rnb, fail, indie rock, covers, acoustic, dance
, death song, pop-rock, let's dance, 2010, emo, hip hop, melancholy, female vocalists, piano, love song
s, nelly furtado, pop punk, mellow, acoustic rock, teen pop, 00s, memories, piano rock, female vocals,
power pop, never gets old, awesome, happy, alternative, male version, rainy day, good mood songs, smile
, none, metal, i love dancing about to this like a complete and utter idiot, <3, singer-songwriter, the
best, favourite, fav female singers, vocal, balada, album favourite, under 2000 listeners, seen in conc
ert, fun, alice in wonderland, hyper, horrible, female, soon to be classic, holiday soundtrack, soundtr
ack, jet set radio future, hard rock, albums i own, punk rock, girl power, dream pop, electronic, love
at first listen, female voices, pop, female vocal, alternative rock, emotional, because sometimes i fee
l like a 13-year-old teenie bopper, hot, satanic black metal from hell, blonde, canada, b-side, good mo
od music, ballad, love song, singalong, legend, we should be together, sexy, canadian, sad, punk, rock,
garage rock, catchy, all time favorite, cute, rock female, anime, these songs are just amazing, nice, b
est, favourites, excellent reason for crying, brazilian, guilty pleasure, loved, electropop, wth, pop r
ock, bouncy, avril lavigne, amazing, female artists, songs diana sang on x factor, i am in love with th
is song, love, magic, indie pop, infatuation, pop rock female vocalists canadian punk alternative pop r
ock female, tyler adam, charmed, ballads, female vocalist, poprock, best songs of the 00s, guilty pleas
ures, beautiful lyrics, favorites, vocalists, officially shit, indie, cover, furnoo lovers, remix, amer
ican idol, english, great lyrics, powerpop, good mood, great voice, dance punk, teen punk, fucking awes
ome, pure, seen live, soul, amazing cover, soft rock",
       'female vocalists, popstars, girl groups, pop, german',
       'brasil, sandy leah, pop, brazil, rock, german',
       'female vocalists, popstars, pop, amazing, pop covers, sweet',
       'deutschland sucht den superstar 5, pop covers',
       'powerful voice, diva, goddess, better than britney spears, american, 70s, cher, romantic, balla
d, rnb, female vocalist, girl shit, better than lady gaga, legend, glam, great songs, dance, oldies, be
st music ever, sexy, tutancamon, tinosoft, rock, divas, adult contemporary, female vocalists, club, fem
ale, better than madonna, country, disco, gritos ressussitadores de mortos, 90s, perfection, electronic
, 60s, n 1 hot 100 billboard, power pop, mumia, urban, pop, amazing, soul, 80s, vocalista feminimo, gla
mour, folk, show woman'],
      dtype=object)
```

In [319]:

```
%%time
tfidfv = TfidfVectorizer()
overview_matrix = tfidfv.fit_transform(tags_raw)
```

```
CPU times: user 164 ms, sys: 0 ns, total: 164 ms
Wall time: 165 ms
```

In [320]:

```
overview_matrix
```

Out[320]:

```
<12523x7615 sparse matrix of type '<class 'numpy.float64'>'
 with 141185 stored elements in Compressed Sparse Row format>
```

In [321]:

```
class SimpleKNNRecommender:
```

```
    def __init__(self, X_matrix, X_ids, X_title, X_overview):
        """
        Входные параметры:
        X_matrix - обучающая выборка (матрица объект-признак)
        X_ids - массив идентификаторов объектов
        X_title - массив названий объектов
        X_overview - массив описаний объектов
        """
        #Сохраняем параметры в переменных объекта
        self._X_matrix = X_matrix
        self.df = pd.DataFrame(
            {'id': pd.Series(X_ids, dtype='int'),
            'title': pd.Series(X_title, dtype='str'),
            'overview': pd.Series(X_overview, dtype='str'),
            'dist': pd.Series([], dtype='float')})


    def recommend_for_single_object(self, K: int, \
                X_matrix_object, cos_flag = True, manh_flag = False):
        """
        Метод формирования рекомендаций для одного объекта.
        Входные параметры:
        K - количество рекомендуемых соседей
        X_matrix_object - строка матрицы объект-признак, соответствующая объекту
        cos_flag - флаг вычисления косинусного расстояния
        manh_flag - флаг вычисления манхэттэнского расстояния
        Возвращаемое значение: K найденных соседей
        """
        scale = 1000000
        # Вычисляем косинусную близость
        if cos_flag:
            print(self._X_matrix.shape)
            print(X_matrix_object.shape)
            dist = cosine_similarity(self._X_matrix, X_matrix_object)
            print(dist.shape)
            self.df['dist'] = dist * scale
            res = self.df.sort_values(by='dist', ascending=False)
            # Не учитываем рекомендации с единичным расстоянием,
            # так как это искомый объект
            res = res[res['dist'] < scale]

        else:
            if manh_flag:
                dist = manhattan_distances(self._X_matrix, X_matrix_object)
            else:
                dist = euclidean_distances(self._X_matrix, X_matrix_object)
            self.df['dist'] = dist * scale
            res = self.df.sort_values(by='dist', ascending=True)
            # Не учитываем рекомендации с единичным расстоянием,
            # так как это искомый объект
            res = res[res['dist'] > 0.0]

        # Оставляем K первых рекомендаций
        res = res.head(K)
        return res
```

In [381]:

```
beyonce_id = 257
tag_sets_df.iloc[beyonce_id]['name']
```

Out[381]:

'Beyoncé'

In [382]:

```
tag_sets_df.loc[beyonce_id]
```

Out[382]:

name	Beyoncé

```
tags    dance-pop, diva, american, girl shit, rnb, bea...
Name: 257, dtype: object
```

In [383]:

```
beyonce_matrix = overview_matrix[beyonce_id]
beyonce_matrix
```

Out[383]:

```
<1x7615 sparse matrix of type '<class 'numpy.float64'>'
 with 160 stored elements in Compressed Sparse Row format>
```

In [384]:

```
artist_ids = tag_sets_df.index.values
artist_ids.shape
```

Out[384]:

```
(12523,)
```

In [385]:

```
artist_names = tag_sets_df['name'].values
artist_names.shape
```

Out[385]:

```
(12523,)
```

In [386]:

```
skr1 = SimpleKNNRecommender(overview_matrix, artist_ids, artist_names, tags_raw)
skr1._X_matrix
```

Out[386]:

```
<12523x7615 sparse matrix of type '<class 'numpy.float64'>'
 with 141185 stored elements in Compressed Sparse Row format>
```

In [387]:

```
rec1 = skr1.recommend_for_single_object(15, beyonce_matrix)
rec1
```

```
(12523, 7615)
(1, 7615)
(12523, 1)
```

Out[387]:

	id	title	overview	dist
247	247	Janet Jackson	dance-pop, diva, hit, american, nineties, old,...	432519.634673
64	64	Madonna	dance-pop, ginuwine, diva, jump, house, hit, g...	398468.317903
263	263	Alicia Keys	diva, charmed, r and b, cool, american, neo-so...	393389.455545
52	52	Kylie Minogue	eletropop, dance-pop, aussie, diva, 80s dance,...	375653.997156
817	817	Destiny's Child	christmas, r and b, cool, american, ballad, rn...	368584.515837
250	250	Rihanna	dance-pop, diva, 2000s diva, hit, american, su...	367303.371318
273	273	Natasha Bedingfield	soulful, gorgeous, british, perfect, drive, su...	367110.104215
254	254	Christina Aguilera	crazy, dance-pop, diva, i love girls, beautifu...	365780.379993

	id	title	overview	dist
251	251	Britney Spears	dance-pop, songs to make florchuchizz cry, blo...	365613.6430[illegible]
256	256	Leona Lewis	weekly top artists, hot, diva, vocally perfect...	363698.248310
264	264	P!nk	weekly top artists, hot, 4m4zinq, fuck bush, t...	362578.011069
249	249	Monica	ginuwine, american, ballad, rnb, female vocali...	362464.693217
244	244	Brandy	4m4zinq, ginuwine, r and b, american, neo-soul...	362180.131131
224	224	Mariah Carey	diva, house, american, 1993, pharrell - our fa...	358959.580708
11130	11130	Tigarah	japanese, dance, hip-hop, female voices, j-pop...	357275.380136

In [390]:

```
rec2 = skr1.recommend_for_single_object(15, beyonce_matrix, cos_flag=False)
rec2
```

Out[390]:

	id	title	overview	dist
9306	9306	Prljavo Kazalište	s-r-b-i-j-a	1.000000e+06
9278	9278	Administrator	<3	1.000000e+06
11688	11688	Boban Marković Orkestar	s-r-b-i-j-a	1.000000e+06
247	247	Janet Jackson	dance-pop, diva, hit, american, nineties, old,...	1.065345e+06
64	64	Madonna	dance-pop, ginuwine, diva, jump, house, hit, g...	1.096842e+06
263	263	Alicia Keys	diva, charmed, r and b, cool, american, neo-so...	1.101463e+06
52	52	Kylie Minogue	eletropop, dance-pop, aussie, diva, 80s dance,...	1.117449e+06
817	817	Destiny's Child	christmas, r and b, cool, american, ballad, rn...	1.123758e+06
250	250	Rihanna	dance-pop, diva, 2000s diva, hit, american, su...	1.124897e+06
273	273	Natasha Bedingfield	soulful, gorgeous, british, perfect, drive, su...	1.125069e+06
254	254	Christina Aguilera	crazy, dance-pop, diva, i love girls, beautifu...	1.126250e+06
251	251	Britney Spears	dance-pop, songs to make florchuchizz cry, blo...	1.126398e+06
256	256	Leona Lewis	weekly top artists, hot, diva, vocally perfect...	1.128097e+06
264	264	P!nk	weekly top artists, hot, 4m4zinq, fuck bush, t...	1.129090e+06
249	249	Monica	ginuwine, american, ballad, rnb, female vocali...	1.129190e+06

In [392]:

```
rec3 = skr1.recommend_for_single_object(15, beyonce_matrix, cos_flag=False, manh_flag=True)
rec3
```

Out[392]:

	id	title	overview	dist
9306	9306	Prljavo Kazalište	s-r-b-i-j-a	1.133577e+07
9278	9278	Administrator	<3	1.133577e+07
11688	11688	Boban Marković Orkestar	s-r-b-i-j-a	1.133577e+07
12060	12060	NaN	dance	1.200133e+07
12015	12015	NaN	dance	1.200133e+07
10580	10580	Magic Box	dance	1.200133e+07
4281	4281	Hott 22	dance	1.200133e+07
5788	5788	Cory Lee	dance	1.200133e+07
4229	4229	Dennis Ferrer	dance	1.200133e+07
8823	8823	Electrovamp	dance	1.200133e+07
10676	10676	NaN	dance	1.200133e+07

	id	title	overview	dist
7260	7260	Tiffany Evans	rnb	1.207411e+07
9068	9068	Xscape	rnb	1.207411e+07
10793	10793	NaN	rnb	1.207411e+07
10719	10719	Martin Kember	rnb	1.207411e+07