Development of the model for image search by text query

Content

Project Description

Main task of the project is to develop demo version on imagege search by text query.

For demo version its' required to train the model, which will get as input the vectorized text and vectorized image, and will provide as output number from 0 to 1 - which will define does the text description fir to image.

Data description

Data provided in folder `/datasets/image_search/` .

File `train_dataset.csv` has the required information for models training:

- image file name;
- query id;

- query text.

One image could have up to 5 text queries. Query id has the following format: - `<image file name>#<query_number>`.

Folder `train_images` has images for models training;

File `CrowdAnnotations.tsv` — has data with infromation of correspondence of the image and query, obtained from crouwdsource. The columns numbers following:

1. Image file name.
2. Query id..
3. Percentage of people who confirmed the image and query correspondance.
4. Quantity of pepole who confirm that query fit to image.
5. Quantity of pepole who does not confirm that query fit to image.

File `ExpertAnnotations.tsv` has data on has data with infromation of correspondence of the image and query, obtained during the expers questionnaire. The columns are following:

1. Image file name.
2. Query id.

3, 4, 5 — Experts rating.

Experts are rate the correspondance of image in query using the following scale:

1) — image does not match query;

2) — query has a desription of a few elements of image, but overall not related to image;

3) — query and image correspond to each other with descreption of the details;

4) — query and image match on 100%.

File `test_queries.csv` has the data required for model testing: query id, query text, image file name. One image could have up to 5 text queries. Query if has the following format: - `<image file name>#<query_number>`.

Folder `test_images` has images for testing

1. Exploratory data analysis

In [1]:
```
# libraries import
import re
import os
import nltk
import torch
import pandas as pd
import numpy as np
import transformers
from tqdm import notebook
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LinearRegression
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from sklearn.metrics import mean_absolute_error, mean_squared_error
import psutil
from PIL import Image
import torchvision.models as t_model
import torch.nn as nn
from torchvision import transforms
from sklearn.model_selection import GroupShuffleSplit
import matplotlib.pyplot as plt
from sentence_transformers import SentenceTransformer, util
from gensim.models import Word2Vec

import warnings
warnings.filterwarnings("ignore")
```

data loading

In [2]:
```
DATA_PATH = 'initial_data/'
```

In [3]:
```
df_train = pd.read_csv(DATA_PATH + 'train_dataset.csv')
crowd_annotation = pd.read_table(DATA_PATH+'CrowdAnnotations.tsv',header=None, names=['image', 'query_id',
                                                                                      'confirmed_percentage','confimed_qty
expert_annotation = pd.read_table(DATA_PATH + '/ExpertAnnotations.tsv',header=None, names=['image', 'query_id',
```

```
                                                                                'expert_1','expert_2', 'expert_3'])
df_test_queries = pd.read_csv(DATA_PATH + 'test_queries.csv', sep = '|')
df_test_images = pd.read_csv(DATA_PATH + 'test_images.csv')
```

1.1 Data overview

1.1.1 Train dataset

In [4]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5822 entries, 0 to 5821
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   image       5822 non-null   object
 1   query_id    5822 non-null   object
 2   query_text  5822 non-null   object
dtypes: object(3)
memory usage: 136.6+ KB
```

In [5]:
```
df_train.head()
```

Out[5]:

	image	query_id	query_text
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...
1	1262583859_653f1469a9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...
2	2447284966_d6bbdb4b6e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...
3	2549968784_39bfbe44f9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...
4	2621415349_ef1a7e73be.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...

In [6]:
```
for i in df_train.columns:
    print(i, 'unique qty',len(df_train[i].unique()))
```

```
image unique qty 1000
query_id unique qty 977
query_text unique qty 977
```

Conclusion

- train dataset has 5822 rows and 3 columns: image name, query_id and query text;
- train dataset has 1000 unique photos and 977 descriptions and 977 unique queries id**

1.1.2 Expert_annotation

In [7]:
```
expert_annotation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5822 entries, 0 to 5821
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   image     5822 non-null   object
 1   query_id  5822 non-null   object
 2   expert_1  5822 non-null   int64 
 3   expert_2  5822 non-null   int64 
 4   expert_3  5822 non-null   int64 
dtypes: int64(3), object(2)
memory usage: 227.5+ KB
```

In [8]:
```
expert_annotation.head()
```

Out[8]:

	image	query_id	expert_1	expert_2	expert_3
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	1	1	1
1	1056338697_4f7d7ce270.jpg	2718495608_d8533e3ac5.jpg#2	1	1	2
2	1056338697_4f7d7ce270.jpg	3181701312_70a379ab6e.jpg#2	1	1	2
3	1056338697_4f7d7ce270.jpg	3207358897_bfa61fa3c6.jpg#2	1	2	2
4	1056338697_4f7d7ce270.jpg	3286822339_5535af6b93.jpg#2	1	1	2

In [9]:
```
total_qty_experts = 0
for i in ['expert_1','expert_2','expert_3']:
    print(i, 'unique qty', len(expert_annotation[i]))
    total_qty_experts += len(expert_annotation[i])
```

```
print('total qty', total_qty_experts)
```

```
expert_1 unique qty 5822
expert_2 unique qty 5822
expert_3 unique qty 5822
total qty 17466
```

In [10]:
```
expert_annotation['expert_1'].unique()
```

Out[10]:
```
array([1, 2, 3, 4], dtype=int64)
```

Conclusion

- dataset has 5822 rows and 4 columns: image name, query id and three expert ratings;
- total quantity of expert ratings is 17466, 5822 per expert;
- Experts are rate the image and query correlation using scale from 1 to 4, where:

  1) — image does not match query;

  2) — query has a desription of a few elements of image, but overall not related to image;

  3) — query and image correspond to each other with descreption of the details;

  4) — query and image match on 100%.

1.1.3 Crowd_annotation

In [11]:
```
crowd_annotation.head()
```

Out[11]:

	image	query_id	confirmed_percentage	confimed_qty	rejected_qty
0	1056338697_4f7d7ce270.jpg	1056338697_4f7d7ce270.jpg#2	1.0	3	0
1	1056338697_4f7d7ce270.jpg	114051287_dd85625a04.jpg#2	0.0	0	3
2	1056338697_4f7d7ce270.jpg	1427391496_ea512cbe7f.jpg#2	0.0	0	3
3	1056338697_4f7d7ce270.jpg	2073964624_52da3a0fc4.jpg#2	0.0	0	3
4	1056338697_4f7d7ce270.jpg	2083434441_a93bc6306b.jpg#2	0.0	0	3

In [12]:
```
crowd_annotation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47830 entries, 0 to 47829
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   image                 47830 non-null  object 
 1   query_id              47830 non-null  object 
 2   confirmed_percentage  47830 non-null  float64
 3   confimed_qty          47830 non-null  int64  
 4   rejected_qty          47830 non-null  int64  
dtypes: float64(1), int64(2), object(2)
memory usage: 1.8+ MB
```

In [13]:
```
total_crowd_qty = 0
for i in ['confimed_qty','rejected_qty']:
    total_crowd_qty += sum(crowd_annotation[i])
print('total_crowd_qty', total_crowd_qty)
```

```
total_crowd_qty 144860
```

In [14]:
```
crowd_annotation['confirmed_percentage'].unique()
```

Out[14]:
```
array([1.        , 0.        , 0.33333333, 0.66666667, 0.25      ,
       0.6       , 0.2       , 0.5       , 0.4       , 0.75      ,
       0.16666667, 0.8       ])
```

Total quantity of rating from crowdsource - 144 860

Crowdsource rating either confirm ot reject the match of image to query. The percentage of all rating for every image to be used as total rating from crowdsourse.

1.1.4 Test dataset

In [15]:
```
df_test_queries.head()
```

Out[15]:

	Unnamed: 0	query_id	query_text	image
0	0	1177994172_10d143cb8d.jpg#0	Two blonde boys , one in a camouflage shirt an...	1177994172_10d143cb8d.jpg
1	1	1177994172_10d143cb8d.jpg#1	Two boys are squirting water guns at each other .	1177994172_10d143cb8d.jpg
2	2	1177994172_10d143cb8d.jpg#2	Two boys spraying each other with water	1177994172_10d143cb8d.jpg
3	3	1177994172_10d143cb8d.jpg#3	Two children wearing jeans squirt water at eac...	1177994172_10d143cb8d.jpg
4	4	1177994172_10d143cb8d.jpg#4	Two young boys are squirting water at each oth...	1177994172_10d143cb8d.jpg

In [16]:
```
df_test_queries = df_test_queries.drop(columns = 'Unnamed: 0')
```

In [17]:
```
df_test_queries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   query_id    500 non-null    object
 1   query_text  500 non-null    object
 2   image       500 non-null    object
dtypes: object(3)
memory usage: 11.8+ KB
```

In [18]:
```
for i in df_test_queries.columns:
    print(i, 'unique qty',len(df_test_queries[i].unique()))
```

```
query_id unique qty 500
query_text unique qty 500
image unique qty 100
```

Test dataset has 100 images and 500 text queries

1.1.5 test images dataset

In [19]:
```
df_test_images.head()
```

Out[19]:

	image
0	3356748019_2251399314.jpg
1	2887171449_f54a2b9f39.jpg
2	3089107423_81a24eaf18.jpg
3	1429546659_44cb09cbe2.jpg
4	1177994172_10d143cb8d.jpg

In [20]:
```
df_test_images.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   image   100 non-null    object
dtypes: object(1)
memory usage: 928.0+ bytes
```

Test dataset has 100 unique images

In [21]:
```
(df_test_queries['image'].sort_values().drop_duplicates().reset_index(drop=1) ==
 (df_test_images['image'].sort_values().drop_duplicates().reset_index(drop=1))).unique()
```

Out[21]:
```
array([ True])
```

All images are correspond to file names of images from df_test

1.1.6 EDA conclusion:

- the train dataset, expert and crowdsource ratings to be used in next part of the project for the obtaining of the target values.
- text and images from test dataset to be used for the crating of test features and testing of the model.

1.2 Data preparation - target values calculation:

For the obtainig of target value the following steps to be performed:

- To merge of query dataset and experts annptations;
- To calculate the expertets rating - conventional wisdom method;
- Resale the obtained ratingsto format from 0 to 1;
- Search for errors in expert ratings;
- Deletion of images from datasets with experts errors;
- To add dataset with crowdsource ratings;
- To calculate the overall ratings:
  - ratings of queries to images by experts and crowdsource;
  - ratings of queries to images only by experts;
  - ratings of queries to images only by crowdsource;
- To merge all calculated values to one dataset and evaluate the data destribution;
- Prepare the data for the following processing - to keep in tables only image name, text anf target value.

1.2.1 Query and expert rating datasets merge

In [22]:
```
df_train_expert = pd.merge(df_train,expert_annotation,on=('image', 'query_id'),how='outer',indicator=True)
```

In [23]:
```
df_train_expert.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5822 entries, 0 to 5821
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype   
---  ------      --------------  -----   
 0   image       5822 non-null   object  
 1   query_id    5822 non-null   object  
 2   query_text  5822 non-null   object  
 3   expert_1    5822 non-null   int64   
 4   expert_2    5822 non-null   int64   
 5   expert_3    5822 non-null   int64   
 6   _merge      5822 non-null   category
dtypes: category(1), int64(3), object(3)
memory usage: 324.2+ KB
```

In [24]:
```
# deletion of merge column
df_train_expert = df_train_expert.drop(columns = '_merge')
```

In [25]:
```
df_train_expert.head()
```

Out[25]:

	image	query_id	query_text	expert_1	expert_2	expert_3
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1
1	1262583859_653f1469a9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1
2	2447284966_d6bbdb4b6e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	2	2
3	2549968784_39bfbe44f9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	4	4	4
4	2621415349_ef1a7e73be.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1

1.2.2 Calculation of optimal expert rating - conventional wisdom

In [26]:
```
# calculation of optimal rating:
cumulative = []
for i in df_train_expert.index:
    if df_train_expert['expert_1'][i] == df_train_expert['expert_2'][i] and (
        df_train_expert['expert_1'][i] == df_train_expert['expert_3'][i]):
        cumulative.append(df_train_expert['expert_1'][i])
    elif df_train_expert['expert_1'][i] == df_train_expert['expert_2'][i] and (
        df_train_expert['expert_1'][i] != df_train_expert['expert_3'][i]):
```

```
        cumulative.append(df_train_expert['expert_1'][i])
    elif df_train_expert['expert_1'][i] != df_train_expert['expert_2'][i] and (
        df_train_expert['expert_1'][i] == df_train_expert['expert_3'][i]):
        cumulative.append(df_train_expert['expert_1'][i])
    else:
        if df_train_expert['expert_2'][i] == df_train_expert['expert_3'][i]:
            cumulative.append(df_train_expert['expert_2'][i])
        else:
            cumulative.append((df_train_expert['expert_1'][i] + df_train_expert['expert_2'][i] + df_train_expert['expert_3'][i])
df_train_expert['expert_cumulative'] = cumulative
```

1.2.3 rescaling of calculated rating

In [27]:
```
temp_list = []
expert_max = df_train_expert['expert_cumulative'].max()
expert_min = df_train_expert['expert_cumulative'].min()
for i in df_train_expert['expert_cumulative']:
    temp_list.append((i - expert_min)/ (expert_max - expert_min))
df_train_expert['expert_assesment'] = temp_list
```

In [28]:
```
# удалим expert_cumulative за ненеадообностью
df_train_expert = df_train_expert.drop(columns = 'expert_cumulative')
```

In [29]:
```
df_train_expert.head()
```

Out[29]:

	image	query_id	query_text	expert_1	expert_2	expert_3	expert_assesment
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000
1	1262583859_653f1469a9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000
2	2447284966_d6bbdb4b6e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	2	2	0.333333
3	2549968784_39bfbe44f9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	4	4	4	1.000000
4	2621415349_ef1a7e73be.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000

1.2.4 Search for the errors (in case of huge differents in rating)

In [30]:
```
# adding of min and max rating
min_list = []
max_list = []
for i in df_train_expert.index:
    min_list.append(min(df_train_expert['expert_1'][i], df_train_expert['expert_2'][i], df_train_expert['expert_3'][i]))
    max_list.append(max(df_train_expert['expert_1'][i], df_train_expert['expert_2'][i], df_train_expert['expert_3'][i]))
df_train_expert['expert_min'] = min_list
df_train_expert['expert_max'] = max_list
```

In [31]:
```
df_train_expert.head()
```

Out[31]:

	image	query_id	query_text	expert_1	expert_2	expert_3	expert_assesment	expert_min	expert_max
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000	1	1
1	1262583859_653f1469a9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000	1	1
2	2447284966_d6bbdb4b6e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	2	2	0.333333	1	2
3	2549968784_39bfbe44f9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	4	4	4	1.000000	4	4
4	2621415349_ef1a7e73be.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000	1	1

In [32]:
```
# checking for the errors
check_list = []
```

```
for i in df_train_expert.index:
    if (df_train_expert['expert_max'][i] - df_train_expert['expert_min'][i]) > 2:
        check_list.append('alert')
    else:
        check_list.append('ok')
df_train_expert['check'] = check_list
```

In [33]:
```
# deletion of min and max columns
df_train_expert = df_train_expert.drop(columns = ['expert_min','expert_max'])
```

In [34]:
```
df_train_expert.head()
```

Out[34]:

	image	query_id	query_text	expert_1	expert_2	expert_3	expert_assesment	check
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000	ok
1	1262583859_653f1469a9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000	ok
2	2447284966_d6bbdb4b6e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	2	2	0.333333	ok
3	2549968784_39bfbe44f9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	4	4	4	1.000000	ok
4	2621415349_ef1a7e73be.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1	1	1	0.000000	ok

Display the errors

In [35]:
```
drop_df = df_train_expert[df_train_expert['check']=='alert'].reset_index(drop = True)
```

In [36]:
```
drop_df
```

Out[36]:

	image	query_id	query_text	expert_1	expert_2	expert_3	expert_assesment	check
0	542179694_e170e9e465.jpg	300577375_26cc2773a1.jpg#2	An officer stands next to a car on a city stre...	1	2	4	0.444444	alert
1	3388330419_85d72f7cda.jpg	3358558292_6ab14193ed.jpg#2	The room full of youths reacts emotionally as ...	1	2	4	0.444444	alert
2	542317719_ed4dd95dc2.jpg	542317719_ed4dd95dc2.jpg#2	A smiling child slides down a slippery tube slide	1	4	4	1.000000	alert

1.2.5 Deletion of images with errors in expert ratings

In [37]:
```
def to_drop_ (df_train_expert):
    drop_list = []
    for i in df_train_expert.index:
        if (df_train_expert['image'][i] == drop_df['image'][0] and df_train_expert['query_id'][i] == drop_df['query_id'][0]) or
            df_train_expert['image'][i] == drop_df['image'][1] and df_train_expert['query_id'][i] == drop_df['query_id'][1]) or
            df_train_expert['image'][i] == drop_df['image'][2] and df_train_expert['query_id'][i] == drop_df['query_id'][2]) :
                drop_list.append('Y')
        else:
            drop_list.append('N')
    df_train_expert['drop'] =  drop_list
    df_train_expert = df_train_expert[df_train_expert['drop'] == 'N']
    df_train_expert = df_train_expert.drop(columns = 'drop')
    return(df_train_expert)
```

In [38]:
```
df_train_expert.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5822 entries, 0 to 5821
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   image             5822 non-null   object 
 1   query_id          5822 non-null   object 
 2   query_text        5822 non-null   object 
 3   expert_1          5822 non-null   int64  
 4   expert_2          5822 non-null   int64  
 5   expert_3          5822 non-null   int64  
 6   expert_assesment  5822 non-null   float64
 7   check             5822 non-null   object 
dtypes: float64(1), int64(3), object(4)
memory usage: 538.4+ KB
```

In [39]:
```
df_train_expert = to_drop_(df_train_expert)
df_train_expert.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5819 entries, 0 to 5821
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   image             5819 non-null   object 
 1   query_id          5819 non-null   object 
 2   query_text        5819 non-null   object 
 3   expert_1          5819 non-null   int64  
 4   expert_2          5819 non-null   int64  
 5   expert_3          5819 non-null   int64  
 6   expert_assesment  5819 non-null   float64
 7   check             5819 non-null   object 
dtypes: float64(1), int64(3), object(4)
memory usage: 409.1+ KB
```

In [40]:
```
crowd_annotation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47830 entries, 0 to 47829
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   image                 47830 non-null  object 
 1   query_id              47830 non-null  object 
 2   confirmed_percentage  47830 non-null  float64
 3   confimed_qty          47830 non-null  int64  
 4   rejected_qty          47830 non-null  int64  
dtypes: float64(1), int64(2), object(2)
memory usage: 1.8+ MB
```

In [41]:
```
crowd_annotation = to_drop_(crowd_annotation)
crowd_annotation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 47829 entries, 0 to 47829
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   image                 47829 non-null  object 
 1   query_id              47829 non-null  object 
 2   confirmed_percentage  47829 non-null  float64
 3   confimed_qty          47829 non-null  int64  
 4   rejected_qty          47829 non-null  int64  
dtypes: float64(1), int64(2), object(2)
memory usage: 2.2+ MB
```

In [42]:
```
# deletion of useles columns
df_train_expert_final = df_train_expert[['image','query_id','query_text','expert_assesment']]
```

In [43]:
```
df_train_expert_final.head()
```

Out[43]:

	image	query_id	query_text	expert_assesment
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.000000
1	1262583859_653f1469a9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.000000
2	2447284966_d6bbdb4b6e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.333333
3	2549968784_39bfbe44f9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1.000000
4	2621415349_ef1a7e73be.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.000000

1.2.6 Adding of crowdsource rating to dataset

In [44]:
```
df_train_ttl = pd.merge(df_train_expert_final,crowd_annotation,on=('image', 'query_id'),how='outer',indicator=True)
```

In [45]:
```
df_train_ttl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51320 entries, 0 to 51319
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype   
---  ------                --------------  -----   
 0   image                 51320 non-null  object  
 1   query_id              51320 non-null  object  
 2   query_text            5819 non-null   object  
 3   expert_assesment      5819 non-null   float64 
 4   confirmed_percentage  47829 non-null  float64 
 5   confimed_qty          47829 non-null  float64 
 6   rejected_qty          47829 non-null  float64 
 7   _merge                51320 non-null  category
dtypes: category(1), float64(4), object(3)
memory usage: 3.2+ MB
```

1.2.7 Calculation of joint rating (expert and crowdsource);

Ratings to be calculated as sum of

- 60% - expert ratings;
- 40% - crowdsourec ratings.

In [46]:
```
df_train_ttl[df_train_ttl['_merge'] == 'both'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2328 entries, 0 to 5817
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype   
---  ------                --------------  -----   
 0   image                 2328 non-null   object  
 1   query_id              2328 non-null   object  
 2   query_text            2328 non-null   object  
 3   expert_assesment      2328 non-null   float64 
 4   confirmed_percentage  2328 non-null   float64 
 5   confimed_qty          2328 non-null   float64 
 6   rejected_qty          2328 non-null   float64 
 7   _merge                2328 non-null   category
dtypes: category(1), float64(4), object(3)
memory usage: 147.9+ KB
```

In [47]:
```
df_train_both = df_train_ttl[df_train_ttl['_merge'] == 'both']
```

In [48]:
```
df_train_both.head()
```

Out[48]:

	image	query_id	query_text	expert_assesment	confirmed_percentage	confimed_qty	rejected_qty	_merge
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.000000	0.0	0.0	3.0	both
2	2447284966_d6bbdb4b6e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.333333	0.0	0.0	3.0	both
3	2549968784_39bfbe44f9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1.000000	1.0	3.0	0.0	both
5	3030566410_393c36a6c5.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.333333	0.0	0.0	3.0	both
9	3718964174_cb2dc1615e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.000000	0.0	0.0	3.0	both

In [49]:
```
df_train_both['text_match_img'] = (df_train_both['expert_assesment']*0.6 + df_train_both['confirmed_percentage']*0.4)
```

In [50]:
```
df_train_both_final = df_train_both[['image','query_id','query_text','text_match_img']]
```

In [51]:
```
df_train_both_final['text_match_img'].unique()
```

Out[51]:
```
array([0.        , 0.2       , 1.        , 0.53333333, 0.33333333,
       0.4       , 0.8       , 0.73333333, 0.66666667, 0.46666667,
       0.6       , 0.86666667, 0.5       , 0.28      , 0.3       ,
       0.1       , 0.7       , 0.56      , 0.72      , 0.13333333,
       0.4       ])
```

In [52]:
```
df_train_both_final.head()
```

Out[52]:

	image	query_id	query_text	text_match_img
0	1056338697_4f7d7ce270.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0
2	2447284966_d6bbdb4b6e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.2
3	2549968784_39bfbe44f9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	1.0
5	3030566410_393c36a6c5.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.2
9	3718964174_cb2dc1615e.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0

1.2.8 Calculation of rating of queries only from experts;

Previously calculated ratings to be used (only expert rating)

In [53]:
```
df_train_ttl[df_train_ttl['_merge'] == 'left_only'].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3491 entries, 1 to 5818
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype   
---  ------               --------------  -----   
 0   image                3491 non-null   object  
 1   query_id             3491 non-null   object  
 2   query_text           3491 non-null   object  
 3   expert_assesment     3491 non-null   float64 
 4   confirmed_percentage  0 non-null      float64 
 5   confimed_qty         0 non-null      float64 
 6   rejected_qty         0 non-null      float64 
 7   _merge               3491 non-null   category
dtypes: category(1), float64(4), object(3)
memory usage: 221.7+ KB
```

In [54]:
```
df_only_experts = df_train_ttl[df_train_ttl['_merge'] == 'left_only']
```

In [55]:
```
df_only_experts.head()
```

Out[55]:

	image	query_id	query_text	expert_assesment	confirmed_percentage	confimed_qty	rejected_qty	_merge
1	1262583859_653f1469a9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0	NaN	NaN	NaN	left_only
4	2621415349_ef1a7e73be.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0	NaN	NaN	NaN	left_only
6	3155451946_c0862c70cb.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0	NaN	NaN	NaN	left_only
7	3222041930_f642f49d28.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0	NaN	NaN	NaN	left_only
8	343218198_1ca90e0734.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0	NaN	NaN	NaN	left_only

In [56]:
```
df_only_experts['text_match_img'] = df_only_experts['expert_assesment']
```

In [57]:
```
df_only_experts_final = df_only_experts[['image','query_id','query_text','text_match_img']]
```

In [58]:
```
df_only_experts_final['text_match_img'].unique()
```

Out[58]:
```
array([0.        , 0.33333333, 0.66666667, 1.        ])
```

In [59]:
```
df_only_experts_final.head()
```

Out[59]:

	image	query_id	query_text	text_match_img
1	1262583859_653f1469a9.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0
4	2621415349_ef1a7e73be.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0
6	3155451946_c0862c70cb.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0
7	3222041930_f642f49d28.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0
8	343218198_1ca90e0734.jpg	2549968784_39bfbe44f9.jpg#2	A young child is wearing blue goggles and sitt...	0.0

1.2.8 Calculation of rating made only by crowdsource;

Calculation of average value of ratings

In [60]:
```
df_train_no_text = df_train_ttl[df_train_ttl['_merge'] == 'right_only']
```

1.2.8.1 due to the abssence of text query in dataset, the text from crodwsource dataset to be added.

In [61]:
```
# search for unique pairs query - text
df_query_text = df_train[['query_id','query_text']].drop_duplicates()
```

In [62]:
```
# display of daaset before adding of text to columns
df_train_no_text.head()
```

Out[62]:

	image	query_id	query_text	expert_assesment	confirmed_percentage	confimed_qty	rejected_qty	_merg
5819	1056338697_4f7d7ce270.jpg	1056338697_4f7d7ce270.jpg#2	NaN	NaN	1.0	3.0	0.0	right_on
5820	1056338697_4f7d7ce270.jpg	114051287_dd85625a04.jpg#2	NaN	NaN	0.0	0.0	3.0	right_on
5821	1056338697_4f7d7ce270.jpg	1427391496_ea512cbe7f.jpg#2	NaN	NaN	0.0	0.0	3.0	right_on
5822	1056338697_4f7d7ce270.jpg	2073964624_52da3a0fc4.jpg#2	NaN	NaN	0.0	0.0	3.0	right_on
5823	1056338697_4f7d7ce270.jpg	2083434441_a93bc6306b.jpg#2	NaN	NaN	0.0	0.0	3.0	right_on

In [63]:
```
# adding the text to columns
text_list = []
for i in df_train_no_text['query_id']:
    try:
        text_list.append(df_query_text[df_query_text['query_id'] == i]['query_text'].values[0])
    except:
        text_list.append('')
df_train_no_text['query_text'] = text_list
```

1.2.8.2 Deletion of queries without any text description

In [64]:
```
df_train_no_text[df_train_no_text['query_text'] == '']['query_text'].count()
```

Out[64]:
```
1109
```

In [65]:
```
df_train_no_text = df_train_no_text[df_train_no_text['query_text'] != '']
```

1.2.8.3 Using the calculated presentage os correct rating

In [66]:
```
df_train_no_text['text_match_img'] = df_train_no_text['confirmed_percentage']
```

1.2.8.4 deleting the useles columns and overview the table

In [67]:
```
df_train_no_text_final = df_train_no_text[['image','query_id','query_text','text_match_img']]
```

In [68]:
```
# display unique ratings
df_train_no_text_final['text_match_img'].unique()
```

Out[68]:
```
array([1.        , 0.        , 0.33333333, 0.66666667, 0.25      ,
       0.6       , 0.2       , 0.5       , 0.4       , 0.75      ,
       0.16666667, 0.8       ])
```

In [69]:
```
# display the table
df_train_no_text_final.head()
```

Out[69]:

	image	query_id	query_text	text_match_img
5819	1056338697_4f7d7ce270.jpg	1056338697_4f7d7ce270.jpg#2	A woman is signaling is to traffic , as seen f...	1.0
5820	1056338697_4f7d7ce270.jpg	114051287_dd85625a04.jpg#2	A boy in glasses is wearing a red shirt .	0.0
5821	1056338697_4f7d7ce270.jpg	1427391496_ea512cbe7f.jpg#2	A young boy holds onto a blue handle on a pier .	0.0
5822	1056338697_4f7d7ce270.jpg	2073964624_52da3a0fc4.jpg#2	A woman wearing black clothes , a purple scarf...	0.0
5823	1056338697_4f7d7ce270.jpg	2083434441_a93bc6306b.jpg#2	An older woman with blond hair rides a bicycle...	0.0

1.2.9 Merge of all dataset with calculated ratings

In [70]:
```
df_train_final = df_train_no_text_final.append(df_only_experts_final)
```

In [71]:
```
# index reset
df_train_final = df_train_final.append(df_train_both_final).reset_index(drop=True)
```

In [72]:
```
df_train_final.head()
```

Out[72]:

	image	query_id	query_text	text_match_img
0	1056338697_4f7d7ce270.jpg	1056338697_4f7d7ce270.jpg#2	A woman is signaling is to traffic , as seen f...	1.0
1	1056338697_4f7d7ce270.jpg	114051287_dd85625a04.jpg#2	A boy in glasses is wearing a red shirt .	0.0
2	1056338697_4f7d7ce270.jpg	1427391496_ea512cbe7f.jpg#2	A young boy holds onto a blue handle on a pier .	0.0
3	1056338697_4f7d7ce270.jpg	2073964624_52da3a0fc4.jpg#2	A woman wearing black clothes , a purple scarf...	0.0
4	1056338697_4f7d7ce270.jpg	2083434441_a93bc6306b.jpg#2	An older woman with blond hair rides a bicycle...	0.0

In [73]:
```
df_train_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50211 entries, 0 to 50210
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   image           50211 non-null  object 
 1   query_id        50211 non-null  object 
 2   query_text      50211 non-null  object 
 3   text_match_img  50211 non-null  float64
dtypes: float64(1), object(3)
memory usage: 1.5+ MB
```

1.2.9.1 Ratings destribution overwiew

In [74]:
```
# calculation of quantity of image per rating
df_train_final_count = df_train_final.groupby(['text_match_img'])['image'].count()
```

In [75]:
```
#  transform to df
df_train_final_count = pd.DataFrame(df_train_final_count)
```

In [76]:
```
# index reset
df_train_final_count = df_train_final_count.reset_index()
```

In [77]:
```
# calculation of percentage
df_train_final_count['percetnage'] = round(df_train_final_count['image'] / df_train_final_count['image'].sum()*100,2)
```

In [78]:
```
# display the table and plot the destribution
df_train_final_count.sort_values(by = 'image', ascending = False)
```

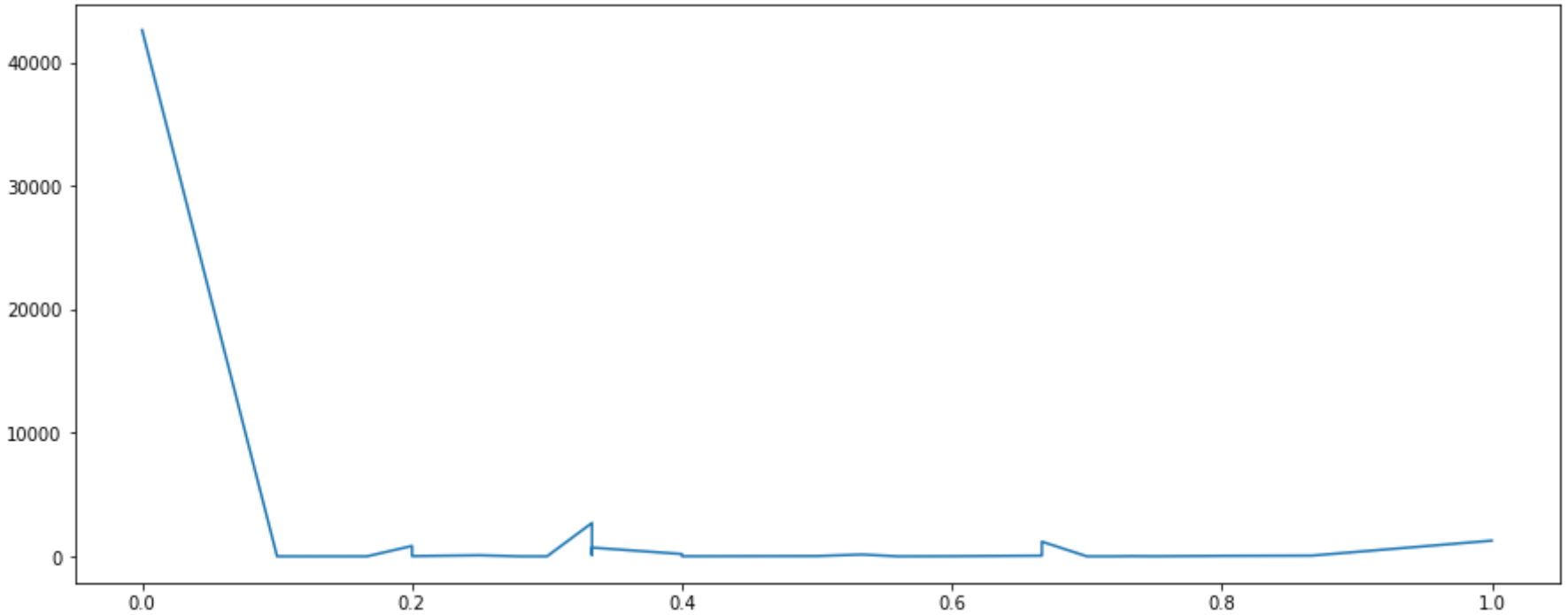
Out[78]:

	text_match_img	image	percetnage
0	0.000000	42621	84.88
9	0.333333	2705	5.39
28	1.000000	1272	2.53
21	0.666667	1190	2.37
4	0.200000	843	1.68
11	0.333333	707	1.41
12	0.400000	190	0.38
16	0.533333	148	0.29
6	0.250000	88	0.18
10	0.333333	87	0.17
20	0.666667	77	0.15
27	0.866667	61	0.12
19	0.666667	55	0.11
26	0.800000	39	0.08
15	0.500000	26	0.05
5	0.200000	20	0.04
14	0.466667	19	0.04
24	0.733333	18	0.04
18	0.600000	12	0.02
13	0.400000	8	0.02
25	0.750000	7	0.01
2	0.133333	4	0.01
3	0.166667	3	0.01
17	0.560000	2	0.00

	text_match_img	image	percetnage
22	0.700000	2	0.00
23	0.720000	2	0.00
8	0.300000	2	0.00
7	0.280000	2	0.00
1	0.100000	1	0.00

In [79]:
```
plt.figure(figsize = (15,6))
plt.plot(df_train_final_count['text_match_img'], df_train_final_count['image'])
```

Out[79]:
```
[<matplotlib.lines.Line2D at 0x1bac2b05060>]
```



1.2.10 Conclusions:

- During the data preparation stage the sataset with 50211 ratings was ontained;

- Datahest has 4 columns - image file name, query name, query text and rating;
- 42621 pairs of querie anf images are definetely not fit to each other - moer than 85% of sample;
- The peaks on the graphs are - 0.33, 0.66 and 1 - and are related to experts ratings (10% of dataset);
- Remaining 5% was destributed freely without any cspecific pattern.

2. Data verification

In several countries where product could be release the are restriction related to search services: it's forbidden to publish information: texts, videos, audios, pictures, etc. with description, photo, video or voice of child. The child is a person under 16 age.

The service under development shall complie with laws in all countries where it will be represented. Due to that fact when someone wonts to see the image forbidden to show by law, it's required to diaplu the follwoing disclaimer:

> This image is unavailable in your country in compliance with local laws

However in this project it's required to delete from train samle the images which couldn't be display in accordance with law restrictions.

In [80]:
```
# function for censore content determination
def is_deprecated(text):
    censored_list = ['boy', 'boys', 'girls','girl', 'teenagers', 'teenager','young', 'children', 'kid', 'kids', 'baby', 'child']
    text = text.lower()
    set_of_words = set(nltk.word_tokenize(text))
    if set(censored_list) & set_of_words:
        return True
    else:
        return False
```

In [81]:
```
# applying of function
df_train_final["deprecated"] = df_train_final['query_text'].apply(is_deprecated)
```

In [82]:
```
df_train_final.head()
```

Out[82]:

	image	query_id	query_text	text_match_img	deprecated
0	1056338697_4f7d7ce270.jpg	1056338697_4f7d7ce270.jpg#2	A woman is signaling is to traffic , as seen f...	1.0	False
1	1056338697_4f7d7ce270.jpg	114051287_dd85625a04.jpg#2	A boy in glasses is wearing a red shirt .	0.0	True
2	1056338697_4f7d7ce270.jpg	1427391496_ea512cbe7f.jpg#2	A young boy holds onto a blue handle on a pier .	0.0	True
3	1056338697_4f7d7ce270.jpg	2073964624_52da3a0fc4.jpg#2	A woman wearing black clothes , a purple scarf...	0.0	False
4	1056338697_4f7d7ce270.jpg	2083434441_a93bc6306b.jpg#2	An older woman with blond hair rides a bicycle...	0.0	False

In [83]:
```
# search of data to be deleted
df_train_censored = df_train_final[(df_train_final['deprecated'] == True) & (df_train_final['text_match_img'] >=0.5)]
```

In [84]:
```
df_train_censored.head()
```

Out[84]:

	image	query_id	query_text	text_match_img	deprecated
171	1096395242_fc69f0ae5a.jpg	1096395242_fc69f0ae5a.jpg#2	A young boy with his foot outstretched aims a ...	0.666667	True
371	1122944218_8eb3607403.jpg	1122944218_8eb3607403.jpg#2	A baby wearing a white gown waves a Muslim flag .	1.000000	True
493	1131932671_c8d17751b3.jpg	2461616306_3ee7ac1b4b.jpg#2	a boy jumps into the blue pool water .	0.666667	True
539	114051287_dd85625a04.jpg	114051287_dd85625a04.jpg#2	A boy in glasses is wearing a red shirt .	1.000000	True
660	1174525839_7c1e6cfa86.jpg	1352410176_af6b139734.jpg#2	A young girl balances on wooden pylons at the ...	0.666667	True

In [85]:
```
# deletion of censored content
df_train_final.drop(df_train_censored.index.values, axis = 0, inplace=True)
```

In [86]:
```
df_train_final = df_train_final.reset_index(drop = True)
```

In [87]:
```
df_train_final.head()
```

Out[87]:

	image	query_id	query_text	text_match_img	deprecated
0	1056338697_4f7d7ce270.jpg	1056338697_4f7d7ce270.jpg#2	A woman is signaling is to traffic , as seen f...	1.0	False
1	1056338697_4f7d7ce270.jpg	114051287_dd85625a04.jpg#2	A boy in glasses is wearing a red shirt .	0.0	True
2	1056338697_4f7d7ce270.jpg	1427391496_ea512cbe7f.jpg#2	A young boy holds onto a blue handle on a pier .	0.0	True
3	1056338697_4f7d7ce270.jpg	2073964624_52da3a0fc4.jpg#2	A woman wearing black clothes , a purple scarf...	0.0	False
4	1056338697_4f7d7ce270.jpg	2083434441_a93bc6306b.jpg#2	An older woman with blond hair rides a bicycle...	0.0	False

In [88]:
```
df_train_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49681 entries, 0 to 49680
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   image           49681 non-null  object
 1   query_id        49681 non-null  object
 2   query_text      49681 non-null  object
 3   text_match_img  49681 non-null  float64
 4   deprecated      49681 non-null  bool
dtypes: bool(1), float64(1), object(3)
memory usage: 1.6+ MB
```

2.1 Conclusion:

- After the deletion of censored images the data has only 49681 records.
- All image with rateing 0.5 and above were deleted.

3. Image Vectorization

The images to be vectorized with pretrained ResNet-18 model with architecture of AdaptiveAvgPool2d and lining.

In [89]:
```
resnet = t_model.resnet18(pretrained = True)
```

In [90]:
```
for param in resnet.parameters():
    param.requires_grad_(False)
```

In [91]:
```
list(resnet.children())
```

Out[91]:
```
[Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False),
 BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True),
 ReLU(inplace=True),
 MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False),
 Sequential(
   (0): BasicBlock(
     (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (relu): ReLU(inplace=True)
     (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   )
   (1): BasicBlock(
     (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (relu): ReLU(inplace=True)
     (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   )
 ),
 Sequential(
   (0): BasicBlock(
     (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
     (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (relu): ReLU(inplace=True)
     (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (downsample): Sequential(
       (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
       (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     )
   )
   (1): BasicBlock(
     (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
     (relu): ReLU(inplace=True)
     (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
     (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
   )
 ),
 Sequential(
   (0): BasicBlock(
     (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
     (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
),
Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
),
AdaptiveAvgPool2d(output_size=(1, 1)),
Linear(in_features=512, out_features=1000, bias=True)]
```

In [92]:

```
modules = list(resnet.children())[:-1]
resnet_1 = nn.Sequential(*modules)
```

In [93]:
```
resnet_1.eval()

norm = transforms.Normalize(
    mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    norm,
])
```

In [94]:
```
def img_to_vec(data, path):
    tensor_list = []
    tmp = 0
    for i in data['image']:
        img = Image.open(path+i).convert('RGB')
        image_tensor = preprocess(img)
        image_tensor = image_tensor.unsqueeze(0)
        output_tensor = resnet_1(image_tensor).flatten()
        tensor_list.append(output_tensor.numpy())
        tmp +=1
        if tmp % 100 == 0:
            print(round(tmp/(len(data['image']))*100,0),'%')
    return tensor_list
```

Train_img_features / vectorization of train images

In [95]:
```
train_img_unique = df_train_final.copy()
```

In [96]:
```
train_img_unique = train_img_unique['image'].drop_duplicates().reset_index(drop=True)
```

In [97]:
```
train_img_unique = pd.DataFrame(train_img_unique, columns = ['image'])
```

In [98]:
```
train_img_unique_features = img_to_vec(train_img_unique, 'initial_data/train_images/')
```

```
10.0 %
20.0 %
30.0 %
40.0 %
50.0 %
60.0 %
70.0 %
80.0 %
90.0 %
100.0 %
```

In [99]:
```
# vectors to df
train_img_unique_features = pd.DataFrame(train_img_unique_features)
```

In [100…
```
train_img_unique_features['image'] = train_img_unique
```

In [101…
```
train_img = df_train_final.copy()
train_img = pd.DataFrame(train_img, columns = ['image'])
```

In [102…
```
train_img_features_df = pd.merge(train_img,train_img_unique_features,on=('image'),how='outer')
```

In [103…
```
train_img_features_df = train_img_features_df.drop(columns = 'image')
```

In [104…
```
train_img_features_df
```

Out[104]:

	0	1	2	3	4	5	6	7	8	9	...	502	503	504	505	
0	0.693940	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.444168	0.717752	0.294673	0.728979	1.
1	0.693940	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.444168	0.717752	0.294673	0.728979	1.
2	0.693940	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.444168	0.717752	0.294673	0.728979	1.
3	0.693940	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.444168	0.717752	0.294673	0.728979	1.
4	0.693940	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.444168	0.717752	0.294673	0.728979	1.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49676	1.836281	0.537826	0.549317	1.469246	0.357578	1.096264	0.521474	0.780281	2.023147	0.112635	...	0.964279	1.641931	1.305999	1.212959	0.
49677	1.836281	0.537826	0.549317	1.469246	0.357578	1.096264	0.521474	0.780281	2.023147	0.112635	...	0.964279	1.641931	1.305999	1.212959	0.
49678	1.836281	0.537826	0.549317	1.469246	0.357578	1.096264	0.521474	0.780281	2.023147	0.112635	...	0.964279	1.641931	1.305999	1.212959	0.
49679	1.836281	0.537826	0.549317	1.469246	0.357578	1.096264	0.521474	0.780281	2.023147	0.112635	...	0.964279	1.641931	1.305999	1.212959	0.
49680	1.836281	0.537826	0.549317	1.469246	0.357578	1.096264	0.521474	0.780281	2.023147	0.112635	...	0.964279	1.641931	1.305999	1.212959	0.

49681 rows × 512 columns

Test img features / vectorisation of test images

In [105...
```
test_img_features_df = img_to_vec(df_test_images, 'initial_data/test_images/')
```

```
100.0 %
```

In [106...
```
test_img_features_df = pd.DataFrame(test_img_features_df)
```

In [107...
```
test_img_features_df['image'] = df_test_images['image']
```

In [108...
```
test_img_features_df
```

Out[108]:

	0	1	2	3	4	5	6	7	8	9	...	503	504	505	506	5
0	0.669249	0.004512	0.236133	0.882863	1.709107	0.125581	1.020888	0.168621	1.178943	0.865995	...	0.749478	2.343132	0.512026	1.853229	1.385(
1	0.949634	3.252454	0.698395	1.281548	0.327134	0.496780	0.039867	0.161799	0.413237	0.498238	...	0.314517	1.681127	0.658190	2.667417	0.8137
2	1.417580	0.973497	0.584019	0.330045	2.646095	0.159388	1.164999	0.870697	0.707525	1.298972	...	0.402308	0.314999	0.221909	0.883103	0.4386
3	0.189787	1.876468	0.825718	0.621149	2.310625	0.565720	0.033543	0.556440	0.402366	4.919666	...	0.729771	0.114588	0.915989	0.860376	1.0786
4	0.443970	2.334722	0.006340	2.394032	0.022515	0.329775	1.684274	1.900405	0.215839	0.033497	...	0.322904	1.906021	1.066521	0.000000	2.2322
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
95	1.208322	0.739464	0.050839	0.083080	0.920781	0.655538	0.497049	0.089066	1.798971	0.400781	...	0.555691	0.512230	0.740154	1.496848	0.3933
96	1.183169	1.067630	1.575608	2.379034	1.213933	0.715755	0.162690	1.838291	0.662763	1.722738	...	0.926513	3.265692	1.928150	1.340862	1.1241
97	0.090710	1.954055	0.718825	0.601076	0.089057	0.158459	1.597549	0.435845	0.418540	0.376853	...	0.656670	0.903572	0.796692	0.846823	0.7180
98	0.193619	0.847930	0.737440	0.553259	0.133539	0.554890	0.057230	1.708564	1.222293	0.635215	...	0.988613	0.946414	0.568584	1.705734	0.1227
99	0.094777	0.438139	2.430755	0.666201	0.313920	0.448235	0.630196	1.002802	1.046606	0.097983	...	1.649578	0.877142	0.746205	1.250985	0.5619

100 rows × 513 columns

3.1 Conclusions:

- Image vectorization successfully completed;
- the following datasets are obtained:

  1) Train - 49681 * 512

  2) Test - 100 * 512 + image file name

4. Text Vectorization

- Text vectorization to be executed via Word2vec model with vector of average vectors of words for each query.

In [109…
```
stop_words = set(stopwords.words('english'))
```

In [110…
```
# function for obtaining of list of words from text
def df_to_text(df_train_censored):
    train_text_ = []
    for i in df_train_censored['query_text']:
        processed_text = i.lower()
        processed_text  = re.sub('[^a-zA-Z]', ' ', processed_text)
        processed_text  = re.sub(r'\s+', ' ', processed_text)
        processed_text = nltk.word_tokenize(processed_text)
        filtered_words= []
        for w in processed_text:
            if w not in stop_words:
                filtered_words.append(w)
        train_text_.append(filtered_words)
    return(train_text_)
```

In [111…
```
train_text_ = df_to_text(df_train_final)
```

In [112…
```
test_text = df_to_text(df_test_queries)
```

In [113…
```
_text_ = train_text_+test_text
```

In [114…
```
model_wv = Word2Vec(_text_, min_count=1, vector_size = 300)
```

In [115…
```
# function trasnform text to vector
def text_to_vec(df_test_queries):
    text_vector =[]
    for j in df_test_queries['query_text']:
        processed_text = j.lower()
        processed_text  = re.sub('[^a-zA-Z]', ' ', processed_text)
        processed_text  = re.sub(r'\s+', ' ', processed_text)
        processed_text = nltk.word_tokenize(processed_text)
        filtered_words= []
        for w in processed_text:
            if w not in stop_words:
                filtered_words.append(w)
        text_vector.append(filtered_words)
```

```
    vector_list = []
    for h in text_vector:
        v1 = model_wv.wv[h]
        vector_list.append(np.mean(v1, axis=0))

    df = pd.DataFrame(vector_list)
    return(df)
```

In [116…
```
train_text_features = text_to_vec(df_train_final)
test_text_features = text_to_vec(df_test_queries)
```

In [117…
```
train_text_features
```

Out[117]:

	0	1	2	3	4	5	6	7	8	9	...	290	291	292	
0	-0.223816	0.043280	-0.078086	0.221208	-0.001817	-0.249398	-0.274902	0.260870	-0.007998	-0.012579	...	0.289845	0.128652	0.490216	-0
1	-0.128972	0.093621	-0.237345	-0.089551	0.059418	-0.434698	0.084664	0.379581	0.555430	-0.072260	...	0.414929	0.258088	0.404733	0
2	0.143807	0.444889	-0.324407	-0.197311	0.073131	-0.598512	0.209585	-0.076982	0.168799	-0.078752	...	0.036598	0.248512	0.169012	0
3	-0.139792	0.177083	0.015613	-0.184344	-0.255431	-0.502678	0.414699	0.654368	0.386166	-0.275497	...	0.357414	0.623727	0.509810	0
4	-0.105150	0.551294	-0.398036	0.083901	-0.034659	-0.630937	0.002720	0.914875	-0.032187	0.034839	...	0.038328	0.454754	0.603128	-0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
49676	-0.120095	0.041895	-0.311475	-0.180811	-0.079752	-0.743445	0.101740	0.564519	-0.155389	-0.220305	...	0.180551	0.309318	0.293546	0
49677	0.035119	0.012204	0.046422	-0.108008	-0.115551	-0.194905	0.034466	0.596069	0.127441	-0.217881	...	0.025535	0.241457	0.129289	-0
49678	-0.092819	0.047741	-0.013888	0.120759	0.091784	-0.138202	-0.230959	0.161287	-0.010925	0.131612	...	0.065074	-0.022302	0.218122	-0
49679	0.211670	-0.080929	-0.232407	-0.074551	0.133177	-0.439170	-0.220965	0.452511	0.419318	0.016256	...	-0.039866	0.117169	0.195517	-0
49680	-0.032769	0.286745	0.170310	0.043912	-0.010254	0.217619	-0.002461	0.200521	-0.103633	0.101296	...	-0.149822	0.207247	-0.322923	0

49681 rows × 300 columns

In [118…
```
len(train_text_features.columns)
```

Out[118]:
```
300
```

In [119…
```
text_col_names = []
number = 0
for i in range(len(train_text_features.columns)):
    text_col_names.append('text_col_'+str(number))
    number+=1
```

In [120…
```
train_text_features.columns = text_col_names
test_text_features.columns = text_col_names
```

In [121…
```
train_text_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49681 entries, 0 to 49680
Columns: 300 entries, text_col_0 to text_col_299
dtypes: float32(300)
memory usage: 56.9 MB
```

In [122…
```
train_text_features.head()
```

Out[122]:

	text_col_0	text_col_1	text_col_2	text_col_3	text_col_4	text_col_5	text_col_6	text_col_7	text_col_8	text_col_9	...	text_col_290	text_col_291	text_c
0	-0.223816	0.043280	-0.078086	0.221208	-0.001817	-0.249398	-0.274902	0.260870	-0.007998	-0.012579	...	0.289845	0.128652	0.4
1	-0.128972	0.093621	-0.237345	-0.089551	0.059418	-0.434698	0.084664	0.379581	0.555430	-0.072260	...	0.414929	0.258088	0.4
2	0.143807	0.444889	-0.324407	-0.197311	0.073131	-0.598512	0.209585	-0.076982	0.168799	-0.078752	...	0.036598	0.248512	0.1
3	-0.139792	0.177083	0.015613	-0.184344	-0.255431	-0.502678	0.414699	0.654368	0.386166	-0.275497	...	0.357414	0.623727	0.5
4	-0.105150	0.551294	-0.398036	0.083901	-0.034659	-0.630937	0.002720	0.914875	-0.032187	0.034839	...	0.038328	0.454754	0.6

5 rows × 300 columns

In [123…
```
train_text_features['image'] = df_train_final['image']
train_text_features['target'] = df_train_final['text_match_img']
```

In [124…
```
for i in ['query_id','query_text','image']:
    test_text_features[i] = df_test_queries[i]
```

In [125…
```
test_text_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Columns: 303 entries, text_col_0 to image
dtypes: float32(300), object(3)
memory usage: 597.8+ KB
```

4.1 Conclusions:

- Vectorization succesfully performed
- the following datasets are prepared:

  1) train - 49681 * 1302 + target and image file name

  2) tst - 500 * 1302 + columns with text and image file name

5. Vectors merging

Preparation of full datasets with features and targets for train and tset samples

train

In [126…
```
train_df_prepared = train_img_features_df.join(train_text_features)
```

In [127…
```
train_df_prepared.head()
```

Out[127]:

	0	1	2	3	4	5	6	7	8	9	...	text_col_292	text_col_293	text_col_294	text_co
0	0.69394	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.490216	-0.059087	0.164893	0.0
1	0.69394	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.404733	0.224961	0.241238	-0.1
2	0.69394	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.169012	0.203210	0.337754	-0.0
3	0.69394	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.509810	0.049272	0.461286	-0.0
4	0.69394	3.031836	2.916933	0.951898	0.936295	1.245117	0.826524	1.107943	0.169679	0.365382	...	0.603128	-0.060601	0.316756	0.2

5 rows × 814 columns

In [128...

```
train_df_prepared.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49681 entries, 0 to 49680
Columns: 814 entries, 0 to target
dtypes: float32(812), float64(1), object(1)
memory usage: 156.0+ MB
```

5.1 Conclusion:

- vectors are merge tp dataset 49681 * 1814 of features + target and image file name
- we can start the models training

6. Models training

- For models training the train sample to be splitted on train and valid samples using groupshufflesplit;
- Neural networks to be trained and and their scores compared to select the best model for future testing.

Data splitting on train and valid samples

In [129…
```
gss = GroupShuffleSplit(n_splits=1, train_size=.7, random_state=42)
```

In [130…
```
train_indices, valid_indices = next(gss.split(X=train_df_prepared.drop(columns=['target','image']), y=train_df_prepared['target'
```

In [131…
```
train_df, valid_df = train_df_prepared.loc[train_indices], train_df_prepared.loc[valid_indices]
```

In [132…
```
train_features, train_target = train_df.drop(columns=['target','image']), train_df['target']
valid_features, valid_target = valid_df.drop(columns=['target','image']), valid_df['target']
```

Linear regression model training

In [133…
```
lr_model = LinearRegression()
```

In [134…
```
lr_model.fit(train_features,train_target)
```

Out[134]:
```
LinearRegression
LinearRegression()
```

In [135…
```
lr_prediction = lr_model.predict(valid_features)
```

In [136…
```
rmse_lr_model = mean_squared_error(lr_prediction, valid_target, squared = False)
```

In [137…
```
rmse_lr_model
```

Out[137]:
```
0.18813374349341422
```

Neural Networks training

In [138…
```
train_features = torch.tensor(train_features.values)
```

In [139…
```
valid_features = torch.tensor(valid_features.values)
```

In [140…
```
valid_target = torch.tensor(valid_target.values)
```

In [141…
```
train_target = torch.tensor(train_target.values)
```

In [142…

```
# building of nerual network 1
torch.manual_seed(1234)
input_size = 812
output_size = 1

class NeuralNet(nn.Module):
    def __init__(self, input_size, output_size):
        super(NeuralNet, self).__init__()
        self.fc1 = nn.Linear(input_size, output_size)
        self.act1 = nn.ReLU()

    def forward(self, x):
        x = self.fc1(x)
        x = self.act1(x)
        return x

model_nn = NeuralNet(input_size,output_size)
```

In [143…

```
# train of model 1

optimizer = torch.optim.Adam(model_nn.parameters(),lr=0.00001)

loss = torch.nn.MSELoss()

num_epochs = 25

for epoch in range(num_epochs):
    optimizer.zero_grad()
    preds = model_nn.forward(train_features.float()).flatten()
    loss_value = loss(preds,train_target.float())
    loss_value.backward()
    optimizer.step()
    if (epoch % 2 == 0) or (epoch == 25):
            model_nn.eval()
            valid_preds_nn = model_nn.forward(valid_features.float()).flatten()
            loss_preds = loss(valid_preds_nn,valid_target)
            print('valid_loss:',loss_preds)
            rmse_nn = mean_squared_error(valid_preds_nn.detach().numpy(), valid_target.detach().numpy(), squared = False)
            print('valid_rmse_x1000:',round(rmse_nn*1000))
```

```
valid_loss: tensor(0.1906, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 437
valid_loss: tensor(0.1846, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 430
valid_loss: tensor(0.1788, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 423
valid_loss: tensor(0.1732, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 416
valid_loss: tensor(0.1677, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 410
valid_loss: tensor(0.1625, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 403
valid_loss: tensor(0.1574, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 397
valid_loss: tensor(0.1525, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 390
valid_loss: tensor(0.1477, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 384
valid_loss: tensor(0.1432, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 378
valid_loss: tensor(0.1388, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 373
valid_loss: tensor(0.1346, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 367
valid_loss: tensor(0.1305, dtype=torch.float64, grad_fn=<MseLossBackward0>)
valid_rmse_x1000: 361
```

In [144…

```python
# building of nerual network 2
torch.manual_seed(1234)
input_size = 812
hidden_size_1 = 512
hidden_size_2 = 32
output_size = 1

class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size_1, hidden_size_2, output_size):
        super(NeuralNet, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size_1)
        self.fc2 = nn.Linear(hidden_size_1, hidden_size_2)
        self.fc3 = nn.Linear(hidden_size_2, output_size)
        self.act3 = nn.ReLU()


    def forward(self, x):
```

```
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)
        x = self.act3(x)
        return x

model_nn_2 = NeuralNet(input_size, hidden_size_1, hidden_size_2, output_size)
```

In [145…

```
optimizer = torch.optim.Adam(model_nn_2.parameters(),lr=0.00005)

loss = torch.nn.MSELoss()

num_epochs = 80

for epoch in range(num_epochs):
    optimizer.zero_grad()
    preds = model_nn_2.forward(train_features.float()).flatten()
    loss_value = loss(preds,train_target.float())
    loss_value.backward()
    optimizer.step()
    if (epoch % 10 == 0) or (epoch == 80):
            model_nn_2.eval()
            valid_preds_nn_2 = model_nn_2.forward(valid_features.float()).flatten()
            loss_preds_2 = loss(valid_preds_nn_2,valid_target.float())
            print('valid_loss:',loss_preds_2)
            rmse_nn_2 = mean_squared_error(valid_preds_nn_2.detach().numpy(), valid_target.detach().numpy(), squared = False)
            print('valid_rmse:',round(rmse_nn_2*1000))
```

```
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse: 201
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse: 201
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse: 201
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse: 201
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse: 201
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse: 201
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse: 201
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse: 201
```

In [146…

```
# building of nerual network 3

torch.manual_seed(1234)
input_size = 812
hidden_size_1 = 512
hidden_size_2 = 64
output_size = 1

class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size_1, hidden_size_2, output_size):
        super(NeuralNet, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size_1)
        self.dp1 = nn.Dropout(p = 0.1)
        self.act1 = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size_1, hidden_size_2)
        self.dp2 = nn.Dropout(p = 0.05)
        self.act2 = nn.ReLU()
        self.fc3 = nn.Linear(hidden_size_2, output_size)
        self.dp3 = nn.Dropout(p = 0.05)
        self.act3 = nn.ReLU()

    def forward(self, x):
        x = self.fc1(x)
        x = self.dp1(x)
        x = self.act1(x)
        x = self.fc2(x)
        x = self.dp2(x)
```

```
        x = self.act2(x)
        x = self.fc3(x)
        x = self.dp3(x)
        x = self.act3(x)
        return x

model_nn_3 = NeuralNet(input_size, hidden_size_1, hidden_size_2, output_size)
```

In [147…

```
optimizer = torch.optim.Adam(model_nn_3.parameters(),lr=0.00005)

loss = torch.nn.MSELoss()

num_epochs = 60

for epoch in range(num_epochs):
    optimizer.zero_grad()
    preds = model_nn_3.forward(train_features.float()).flatten()
    loss_value = loss(preds,train_target.float())
    loss_value.backward()
    optimizer.step()
    if (epoch % 6 == 0) or (epoch == 60):
            model_nn_3.eval()
            valid_preds_nn_3 = model_nn_3.forward(valid_features.float()).flatten()
            loss_preds_3 = loss(valid_preds_nn_3,valid_target.float())
            print('valid_loss:',loss_preds_3)
            rmse_nn_3 = mean_squared_error(valid_preds_nn_3.detach().numpy(), valid_target.detach().numpy(), squared = False)
            print('valid_rmse*1000:',round(rmse_nn_3*1000))
```

```
valid_loss: tensor(0.0404, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 201
valid_loss: tensor(0.0386, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 196
valid_loss: tensor(0.0368, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 192
valid_loss: tensor(0.0362, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 190
valid_loss: tensor(0.0357, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 189
valid_loss: tensor(0.0353, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 188
valid_loss: tensor(0.0351, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 187
valid_loss: tensor(0.0348, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 187
valid_loss: tensor(0.0347, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 186
valid_loss: tensor(0.0345, grad_fn=<MseLossBackward0>)
valid_rmse*1000: 186
```

Selection of best model

In [148…
```
model_list = [lr_model,model_nn,model_nn_2,model_nn_3]
loss_preds_list = [rmse_lr_model,rmse_nn,rmse_nn_2,rmse_nn_3]
```

In [149…
```
loss_preds_list
```

Out[149]:
```
[0.18813374349341422,
 0.361310782909795,
 0.20091421231540807,
 0.1858130876712399]
```

In [150…
```
models_df = pd.DataFrame(model_list,loss_preds_list, columns = ['model'])
```

In [151…
```
models_df = models_df.reset_index()
```

In [152…
```
models_df = models_df.rename(columns = {'index' : 'loss'})
```

In [153…
```
models_df.sort_values(by = 'loss')
```

Out[153]:

	loss	model
3	0.185813	NeuralNet(\n (fc1): Linear(in_features=812, o...
0	0.188134	LinearRegression()
2	0.200914	NeuralNet(\n (fc1): Linear(in_features=812, o...
1	0.361311	NeuralNet(\n (fc1): Linear(in_features=812, o...

In [154...
```
best_model = models_df[models_df['loss'] == models_df['loss'].min()]['model']
```

In [155...
```
best_model = best_model.values[0]
```

In [156...
```
best_model
```

Out[156]:
```
NeuralNet(
  (fc1): Linear(in_features=812, out_features=512, bias=True)
  (dp1): Dropout(p=0.1, inplace=False)
  (act1): ReLU()
  (fc2): Linear(in_features=512, out_features=64, bias=True)
  (dp2): Dropout(p=0.05, inplace=False)
  (act2): ReLU()
  (fc3): Linear(in_features=64, out_features=1, bias=True)
  (dp3): Dropout(p=0.05, inplace=False)
  (act3): ReLU()
)
```

7. Model testing

For model testing it's required to one function that will select the input numbers of queries and search for the best suitable image to each quiery and display the results - image. query and percentage of correspondance.

After model testing it's required to analyze the results.

In [157...
```
# demo testing function
def test(n):
    # create a dataset with n quantity of random text queries
```

```
    df_ten_test_queries = pd.DataFrame(test_text_features['query_text'].sample(n))
    df_ten_test_queries = df_ten_test_queries.reset_index(drop = True)
    df_test_final = pd.DataFrame(index=range(len(df_ten_test_queries)*len(test_img_features_df['image'])),columns=range(2))
    df_test_final.columns = ['query_text','image']

    # create a dataset of all possible pairs of n quantity of queries and images
    n = 0
    for i in range(len(df_ten_test_queries.index)):
        for j in test_img_features_df.index:
            df_test_final['query_text'][j+n] = df_ten_test_queries['query_text'][i]
            df_test_final['image'][j+n] = test_img_features_df['image'][j]
        n+=100

    # transform dataset to vector
    df_test_final = pd.merge(df_test_final,test_img_features_df,on=('image'),how='inner')
    df_text_vectors = test_text_features.drop(columns =['image','query_id'])
    df_test_final = pd.merge(df_test_final,df_text_vectors,on=('query_text'),how='inner')


    # prediction of percentage of image and text correspondance
    test_features = df_test_final.drop(columns = ['image','query_text'])
    test_features = torch.tensor(test_features.values)
    predictions = best_model.forward(test_features.float()).flatten()
    df_test_final['result'] = predictions.detach().numpy()
    df_test_final = df_test_final[['query_text','image','result']]

    #  selection the image with highest predicted rating for each query
    img_list = []
    prediction_list = []
    for i in df_ten_test_queries['query_text']:
        temp_df_test = df_test_final[df_test_final['query_text'] == i]
        img_list.append(temp_df_test[temp_df_test['result'] == temp_df_test['result'].max()]['image'].values[0])
        prediction_list.append(temp_df_test[temp_df_test['result'] == temp_df_test['result'].max()]['result'].values[0])
    df_ten_test_queries['image'] = img_list
    df_ten_test_queries['prediction'] = prediction_list

    #  censore check
    df_ten_test_queries["deprecated"] = df_ten_test_queries['query_text'].apply(is_deprecated)

#     display of image, text and % correspondance
    for i in range(len(df_ten_test_queries['image'])):
        if df_ten_test_queries['deprecated'][i] == False:
            img = Image.open('initial_data/test_images/'+df_ten_test_queries['image'][i]).convert('RGB')
            print('Text:',df_ten_test_queries['query_text'][i],'\n','Prediction:', round(df_ten_test_queries['prediction'][i]*10
```

```
            display(img)
            print('\n')
        else:
            print('Text:',df_ten_test_queries['query_text'][i],'\n','Prediction:', round(df_ten_test_queries['prediction'][i]*10(
            print('\n','\033[1m'+'\033[91m'+'This image is unavailable in your country in compliance with local laws','\n')
            print('\u001b[0m'+'\n')
```

In [158…

```
test(10)
```

```
Text: A shirtless male looks to his right while water flows over him .
 Prediction: 18.83 %
```

Text: A group of people in anime cosplay costumes .
 Prediction: 15.5 %

Text: Women wearing red and black are clapping .
 Prediction: 14.32 %

Text: Closeup of a man at an event with formal attire .
 Prediction: 16.5 %

Text: A small boy wearing glasses stands on a rope and holds two ropes with his hands .
 Prediction: 13.62 %

 This image is unavailable in your country in compliance with local laws

Text: A girl with Indian clothing on and henna on her hand goes through paperwork .
 Prediction: 14.23 %

 This image is unavailable in your country in compliance with local laws

Text: A dog wrapped with straps is walking away from a red tray holding a bag .
 Prediction: 17.8 %

Text: a woman dumping water on a small child who is in a pool
 Prediction: 16.05 %

This image is unavailable in your country in compliance with local laws

Text: Two men standing near a metal structure in from of a brick wall .
 Prediction: 21.66 %

Text: A black and white dog with a green collar stands in front of a sign .
 Prediction: 16.96 %

The result of model testing are quaite low, below you can find the alternative proposed model for demo version of search.

Applying of Alternative model

In [159…

```
#Load CLIP model
model = SentenceTransformer('clip-ViT-B-32')
```

```
ftfy or spacy is not installed using BERT BasicTokenizer instead of ftfy.
```

In [160…

```python
def test_clip(n):
    # create a dataset with n quantity of random text queries
    df_ten_test_queries = pd.DataFrame(test_text_features['query_text'].sample(n))
    df_ten_test_queries = df_ten_test_queries.reset_index(drop = True)
    df_test_final = pd.DataFrame(index=range(len(df_ten_test_queries)*len(test_img_features_df['image'])),columns=range(2))
    df_test_final.columns = ['query_text','image']

    # create a dataset of all possible pairs of n quantity of queries and images
    n = 0
    for i in range(len(df_ten_test_queries.index)):
        for j in test_img_features_df.index:
            df_test_final['query_text'][j+n] = df_ten_test_queries['query_text'][i]
            df_test_final['image'][j+n] = test_img_features_df['image'][j]
        n+=100
    list_result = []

    # prediction of percentage of image and text correspondance
    for i in range(len(df_test_final['query_text'])):
        #Encode an image:
        img_emb = model.encode(Image.open(DATA_PATH+'test_images/' + df_test_final['image'][int(i)]))

        #Encode text descriptions
        text_emb = model.encode(df_test_final['query_text'][i])

        #Compute cosine similarities
        cos_scores = util.cos_sim(img_emb, text_emb)
        cos_scores = cos_scores.detach().numpy()
        list_result.append(cos_scores[0][0])
    df_test_final['result'] = list_result

    #  selection the image with highest predicted rating for each query
    img_list = []
    prediction_list = []
    for i in df_ten_test_queries['query_text']:
        temp_df_test = df_test_final[df_test_final['query_text'] == i]
        img_list.append(temp_df_test[temp_df_test['result'] == temp_df_test['result'].max()]['image'].values[0])
        prediction_list.append(temp_df_test[temp_df_test['result'] == temp_df_test['result'].max()]['result'].values[0])
    df_ten_test_queries['image'] = img_list
    df_ten_test_queries['prediction'] = prediction_list

    #  censore check
    df_ten_test_queries["deprecated"] = df_ten_test_queries['query_text'].apply(is_deprecated)
```

```
    # display of image, text and % correspondance
    for i in range(len(df_ten_test_queries['image'])):
        if df_ten_test_queries['deprecated'][i] == False:
            img = Image.open('initial_data/test_images/'+df_ten_test_queries['image'][i]).convert('RGB')
            print('Text:',df_ten_test_queries['query_text'][i],'\n','Prediction:', round(df_ten_test_queries['prediction'][i]*10
            display(img)
            print('\n')
        else:
            print('Text:',df_ten_test_queries['query_text'][i],'\n','Prediction:', round(df_ten_test_queries['prediction'][i]*10
            print('\n','\033[1m'+'\033[91m'+'This image is unavailable in your country in compliance with local laws'+'\033[91m'
            print('\u001b[0m'+'\n')
```

In [161…

```
test_clip(10)
```

```
Text: Woman with glasses working at a sewing machine .
 Prediction: 34.13 %
```

```
Text: The heavy man is sitting on a bus and sleeping .
 Prediction: 32.29 %
```

Text: A black dog jumps into the air to get a treat from its owner .
 Prediction: 29.87 %

Text: A dog in a harness pulling a pink carrier behind it on snow .
 Prediction: 31.03 %

Text: a brown and white dog jumps on the sidewalk .
 Prediction: 29.94 %

Text: Grey horse wearing blue cover eating from a orange bucket held by a person in a green shirt .
 Prediction: 33.28 %

Text: a guy and a girl jumping up in the air
 Prediction: 28.84 %

This image is unavailable in your country in compliance with local laws

Text: Women play lacrosse .
 Prediction: 32.15 %

Text: While holding tight to the ball , the man in red socks is getting tackled .
 Prediction: 26.85 %

```
Text: A little girls waist high in sand
 Prediction: 26.95 %
```

```
 This image is unavailable in your country in compliance with local laws
```

Conclustion of model testing and searching of images by query:

- demo version of search successfully developed;
- model tested, but the score prdiction is very low, model making a mistakes and select the images not corresponded to query;
- clip model has the higher score and execute the demo version of search with higher resulr, compare to pytorch nn model.

- Over conclusion of the results - the clip model has successfully complete the search task.
- Clip model mostly has select the correct image for the query.