Введение в обработку естественного языка

Урок 14. Transfer learning

Практическое задание

Задание

1. Взять данные из
   https://www.kaggle.com/datasets/mrapplexz/bashim-quotes
   обучить модель GPT для генерации своих цитат

Это задача безусловной генерации текста. Для её решения подходит архитектура GPT.

2. Взять новостные данные из
   https://github.com/natasha/corus
   load_lenta2
   нам понадобиться сам текст и заголовок
   обучить модель T5 или GPT для генерации заголовков для статей

Задача условной генерации текста. Для её решения подходит архитектура GPT или T5.

Выполнил ***Соковнин ИЛ***

```
import json
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Задание 1

1.1 Взять данные из
https://www.kaggle.com/datasets/mrapplexz/bashim-quotes

Reading the data

```
from google.colab import files

upload = files.upload()
```

```
Выбрать файлы dataset.jsonl
• dataset.jsonl(n/a) - 40531099 bytes, last modified: 26.11.2020 - 100% done
Saving dataset.jsonl to dataset.jsonl
```

```
!ls -la
```

```
total 39612
drwxr-xr-x 1 root root     4096 Jul 14 10:32 .
drwxr-xr-x 1 root root     4096 Jul 14 07:26 ..
drwxr-xr-x 4 root root     4096 Jul  6 13:21 .config
-rw-r--r-- 1 root root 40531099 Jul 14 10:32 dataset.jsonl
drwxr-xr-x 5 root root     4096 Jul 14 07:33 gazeta
drwxr-xr-x 1 root root     4096 Jul  6 13:22 sample_data
-rw-r--r-- 1 root root     4659 Jul 14 08:25 tweets.txt
```

```
!mkdir data
```

```
!mv 'dataset.jsonl' 'data/dataset.jsonl'
```

```
DATASET_PATH = 'data/dataset.jsonl'

with open(DATASET_PATH) as f: # open the dataset file
    df = pd.read_json(DATASET_PATH, lines=True).set_index('id') # read the .jsonl file into Pandas DataFrame
```

```
print(df.dtypes)
# display(df)
```

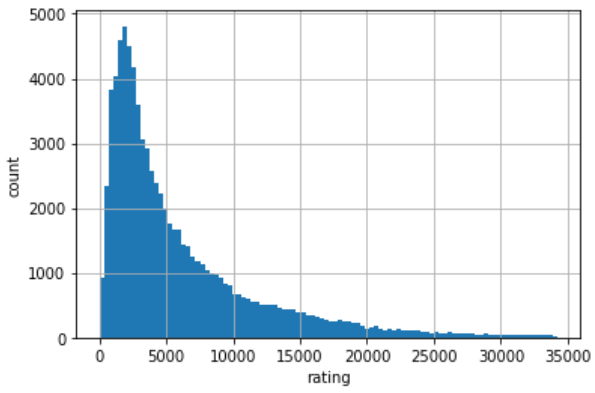
```
date      datetime64[ns, UTC]
rating                float64
text                   object
dtype: object
```

```
df.head()
```

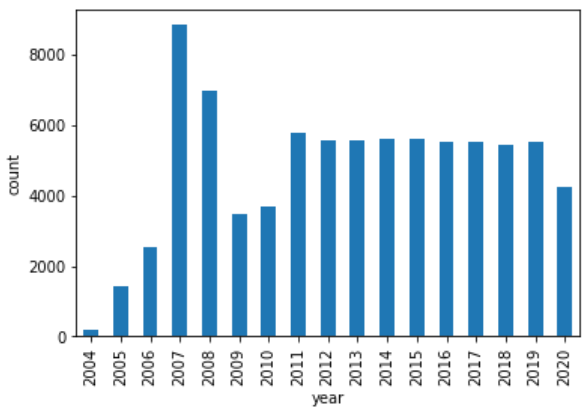
id	date	rating	text
1	2004-08-30 11:24:00+00:00	22010.0	<Ares> ppdv, все юниксы очень дружелюбны.. они...
2	2004-08-30 11:25:00+00:00	25105.0	<томатик_рад> а ты не чувствуешь красоту мира?...
3	2004-08-30 11:27:00+00:00	7192.0	<Дор> "мышка, почему у тебя такие большие глаз...
4	2004-08-30 11:28:00+00:00	29169.0	<PPDV[os2]> "Мальчики, вы что больные, бегать ...
5	2004-08-30 11:26:00+00:00	7140.0	<Ohtori_Akio> мы - как разработчики - живём с ...

Visualizing the data

```
# plot rating histogram with outliers removed
rating = df.rating.dropna()
quantile = rating.quantile(.99)
rating.hist(bins=100, range=(rating.min(), quantile))
plt.xlabel('rating')
plt.ylabel('count')
plt.show()
```

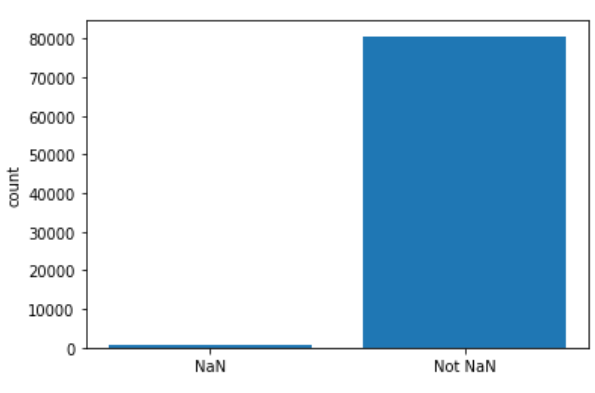



```
# plot quote count by year
by_year = df.groupby(df.date.dt.year)['text'].count()
by_year.plot.bar()
plt.xlabel('year')
plt.ylabel('count')
plt.show()
```




```
# plot count of NaN and not-NaN rated quotes
nans = df.rating.isna().sum()
not_nans = len(df) - nans


bars = [nans, not_nans]
y_pos = np.arange(len(bars))
plt.bar(y_pos, bars)
plt.xticks(y_pos, ('NaN', 'Not NaN'))
plt.ylabel('count')
plt.show()
```




```
pd.set_option('display.max_colwidth', 150)
# pd.set_option('display.width', 500)


# df['text'][0:5]
df['text'][2]
```

```
'<томатик_рад> а ты не чувствуешь красоту мира?\n<fox> честно говоря, я сейчас чувствую только отсутствие http.\n<томатик_рад> не туда смотришь, глянь вокруг!\n<fox> как я гляну, если http не работает? :/'
```

Задание 1

1.2 Обучить модель GPT для генерации своих цитат

Это задача условной генерации текста. Для её решения подходит архитектура GPT.

Цита́та (от лат. citare, citatum — провозглашать, приводить) — дословная выдержка из какого-либо текста.

Обработка сообщений, генерируя из них цитаты.

```
# Загрузка твитов
!wget https://gist.githubusercontent.com/avidale/d3da0ded85a4a16db6eb84d8331638ce/raw/a188084e5ef37b43b01fef0534b55c865b9a569e/tweets.txt
```

```
--2022-07-14 10:34:14--  https://gist.githubusercontent.com/avidale/d3da0ded85a4a16db6eb84d8331638ce/raw/a188084e5ef37b43b01fef0534b55c865b9a569e/tweets.txt
```

```
Resolving gist.githubusercontent.com (gist.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to gist.githubusercontent.com (gist.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4659 (4.5K) [text/plain]
Saving to: ‘tweets.txt.1’

tweets.txt.1        100%[===================>]   4.55K  --.-KB/s    in 0s

2022-07-14 10:34:14 (70.5 MB/s) - ‘tweets.txt.1’ saved [4659/4659]
```

```
!ls
```

```
data  gazeta  sample_data  tweets.txt  tweets.txt.1
```

```
# Закачиваем сохранённые твиты
with open('tweets.txt', 'r') as f:
    tweets = f.read().strip().split('\n\n')
print(len(tweets))
for i in range(3):
    print(tweets[i])
```

```
26
Соловьев наконец-то вышел на новый уровень - теперь его стали банить и в офлайне
Дарим мы тебе бутылку игристого вина. Пить тебе еще рано, но встретиться с ней за некоторые преступления ты уже можешь. ПОЗ-ДРАВ-ЛЯ-ЕМ!
Да. Еще очень многие помнят, что такое госплан, как планировалось, талоны на еду, очереди, дефицит, выездные визы. Но спасибо, что напомнил
```

```
import torch
device = torch.device("cuda")
```

```
from transformers import AutoTokenizer, AutoModelForCausalLM
```

```
# Загружаем large GPT3, которая основана на GPT2
model_name = 'sberbank-ai/rugpt3large_based_on_gpt2'
#model_name = 'Grossmend/rudialogpt3_medium_based_on_gpt2'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name).to(device)
```


```
  "n_layer": 24,
  "n_positions": 2048,
  "reorder_and_upcast_attn": false,
  "resid_pdrop": 0.1,
  "scale_attn_by_inverse_layer_idx": false,
  "scale_attn_weights": true,
  "summary_activation": null,
  "summary_first_dropout": 0.1,
  "summary_proj_to_labels": true,
  "summary_type": "cls_index",
  "summary_use_proj": true,
  "transformers_version": "4.20.1",
  "use_cache": true,
  "vocab_size": 50257
}

Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
loading configuration file https://huggingface.co/sberbank-ai/rugpt3large_based_on_gpt2/resolve/main/config.json from cache at /root/.cache/huggingface/transformers/3a4ae
Model config GPT2Config {
  "_name_or_path": "sberbank-ai/rugpt3large_based_on_gpt2",
  "activation_function": "gelu_new",
  "architectures": [
    "GPT2LMHeadModel"
  ],
  "attn_pdrop": 0.1,
  "bos_token_id": 50256,
  "embd_pdrop": 0.1,
  "eos_token_id": 50256,
  "gradient_checkpointing": false,
  "initializer_range": 0.02,
  "layer_norm_epsilon": 1e-05,
  "model_type": "gpt2",
  "n_ctx": 2048,
  "n_embd": 1536,
  "n_head": 16,
  "n_inner": null,
  "n_layer": 24,
  "n_positions": 2048,
  "reorder_and_upcast_attn": false,
  "resid_pdrop": 0.1,
  "scale_attn_by_inverse_layer_idx": false,
  "scale_attn_weights": true,
  "summary_activation": null,
  "summary_first_dropout": 0.1,
  "summary_proj_to_labels": true,
  "summary_type": "cls_index",
  "summary_use_proj": true,
  "transformers_version": "4.20.1",
  "use_cache": true,
  "vocab_size": 50257
}

loading weights file https://huggingface.co/sberbank-ai/rugpt3large_based_on_gpt2/resolve/main/pytorch_model.bin from cache at /root/.cache/huggingface/transformers/7cf24
All model checkpoint weights were used when initializing GPT2LMHeadModel.

All the weights of GPT2LMHeadModel were initialized from the model checkpoint at sberbank-ai/rugpt3large_based_on_gpt2.
If your task is similar to the task the model of the checkpoint was trained on, you can already use GPT2LMHeadModel for predictions without further training.
```


```
import random


# Мы хотим, чтобы GPT выводила суть, что после 3-х звёздочек ('***')
#  нужно генерировать какой то осмысленный текст похожий на твит.
#
# Пишем рэндомное сэмплирование 5 твитов (берём 5 твитов из 26 случайным обазом)

sep = '\n***\n'  # Признак того, что твит закончился и нужногенерировать ещё один твит
# sep = '\n27479153 Sandy_mustache  2021-02-18 16:44:00 '
```

```
# Так как мы постоянно сэмплируем разные твиты,
# мы будем постоянно получать разное распределение
prefix = sep.join(['']) + random.sample(tweets, k=5) + ['']

tokens = tokenizer(prefix, return_tensors='pt')
tokens = {k: v.to(model.device) for k, v in tokens.items()}
end_token_id = tokenizer.encode('***')[0]  # '***' - токен который будет оканчивать твит

# выводим то, что мы передаём на вход
print(prefix)
```

```
***
Дарим мы тебе бутылку игристого вина. Пить тебе еще рано, но встретиться с ней за некоторые преступления ты уже можешь. ПОЗ-ДРАВ-ЛЯ-ЕМ!
***
Соловьев наконец-то вышел на новый уровень - теперь его стали банить и в офлайне
***
Да. Еще очень многие помнят, что такое госплан, как планировалось, талоны на еду, очереди, дефицит, выездные визы. Но спасибо, что напомнил
***
Ну, с Михалковым все понятно. А для тех, кто тоже не различает понятия “Кооператив Озеро” и “Россия” еще раз напомним, что санкции предлагается вводить не против России, а г
***
А на такие случаи у нас припасена поправочка, принятая на пенечке - и вертели мы этот ЕСПЧ на нашей Конституции!
***
```

```
# Генерируем какой то осмысленный текст похожий на твит.

size = tokens['input_ids'].shape[1]
output = model.generate(
    **tokens,
    #end_token=end_token_id,
    do_sample=False,  # вкл/выкл режим выдачи нескольких вариантов д.б. ещё один параметр
    max_length=size+128,
    # max_length=size+64,
    repetition_penalty=4.2,  # штраф за повторы одинарных токенов
    temperature=0.7,  # температура
    num_beams=10,  # Строим дерево глубины 10
    # no_repeat_ngram_size=3  # ! тройки подряд идущих токенов не должны повторяться (3 и меньше токенов не должны повторяться)
)
decoded = tokenizer.decode(output[0])
result = decoded[len(prefix):]
print(result)
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Короче говоря, если вы думаете, что это шутка, то глубоко заблуждаетесь...
***
Уважаемые дамы и господа, напоминаем вам о том, что наша Конституция гарантирует каждому гражданину право на неприкосновенность частной жизни, личную и семейную тайну, защит
```

```
# Генерируем какой то осмысленный текст похожий на твит.

size = tokens['input_ids'].shape[1]
output = model.generate(
    **tokens,
    #end_token=end_token_id,
    do_sample=False,
    max_length=size+128,
    # max_length=size+64,
    repetition_penalty=4.2,   # штраф за повторы одинарных токенов
    temperature=1.3,  # температура
    num_beams=7,  # Строим дерево глубины 10
    no_repeat_ngram_size=5  # ! тройки подряд идущих токенов не должны повторяться (3 и меньше токенов не должны повторяться)
)
decoded = tokenizer.decode(output[0])
result = decoded[len(prefix):]
print(result)
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Оказывается, есть люди, которые считают, что Путин должен быть пожизненно лишен права баллотироваться на пост Президента Российской Федерации...
***
Удивительно, но факт: по данным опроса Левада-Центра, большинство россиян считает, что Владимир Путин должен остаться Президентом РФ до 2036 года
***
И это только то, что я узнал из открытых источников (в том числе от людей, которых знаю лично)
***
Надеюсь, вы понимаете, о чем идет речь?<s>
Мои твиты Чт, 12:51: RT @Sandy_mustache: https://t.co
```

Вывод

Источник сгенерированного текста в ответах узнаваем. Сгенениранные последовательности имют определённый смысл.

Задание 2

2.1 Взять новостные данные из https://github.com/natasha/corus

load_lenta2

нам понадобиться сам текст и заголовок

2.2 обучить модель T5 или GPT для генерации заголовков для статей

Задача условной генерации текста. Для её решения подходит архитектура GPT или T5.

По abstact сгенерировать заголовок статьи. Используем T5_trainer.ipynb

```
!pip install razdel networkx pymorphy2[fast] nltk rouge==0.3.1
!pip install --upgrade datasets tqdm transformers
```

```
     |████████████████████████████████| 140 kB 77.8 MB/s
Collecting xxhash
  Downloading xxhash-3.0.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (212 kB)
     |████████████████████████████████| 212 kB 71.9 MB/s
Collecting aiohttp
```

```
  Downloading aiohttp-3.8.1-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.1 MB)
     |████████████████████████████████| 1.1 MB 64.9 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from huggingface-hub<1.0.0,>=0.1.0->datasets) (3.7.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from huggingface-hub<1.0.0,>=0.1.0->datasets) (4.1.1)
Collecting pyyaml>=5.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (596 kB)
     |████████████████████████████████| 596 kB 74.4 MB/s
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging->datasets) (3.0.9)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->datasets) (2022.6.15)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->datasets) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->datasets) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->datasets) (3.0.4)
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
  Downloading urllib3-1.25.11-py2.py3-none-any.whl (127 kB)
     |████████████████████████████████| 127 kB 71.4 MB/s
Collecting tokenizers!=0.11.3,<0.13,>=0.11.1
  Downloading tokenizers-0.12.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (6.6 MB)
     |████████████████████████████████| 6.6 MB 65.0 MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (2022.6.2)
Collecting multidict<7.0,>=4.5
  Downloading multidict-6.0.2-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (94 kB)
     |████████████████████████████████| 94 kB 4.6 MB/s
Collecting aiosignal>=1.1.2
  Downloading aiosignal-1.2.0-py3-none-any.whl (8.2 kB)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (21.4.0)
Collecting yarl<2.0,>=1.0
  Downloading yarl-1.7.2-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (271 kB)
     |████████████████████████████████| 271 kB 72.5 MB/s
Collecting async-timeout<5.0,>=4.0.0a3
  Downloading async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Collecting frozenlist>=1.1.1
  Downloading frozenlist-1.3.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (144 kB)
     |████████████████████████████████| 144 kB 73.1 MB/s
Collecting asynctest==0.13.0
  Downloading asynctest-0.13.0-py3-none-any.whl (26 kB)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp->datasets) (2.1.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->datasets) (3.8.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->datasets) (2022.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas->datasets) (1.15.0)
Installing collected packages: multidict, frozenlist, yarl, urllib3, asynctest, async-timeout, aiosignal, pyyaml, fsspec, aiohttp, xxhash, tokenizers, responses, huggingf
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.24.3
    Uninstalling urllib3-1.24.3:
      Successfully uninstalled urllib3-1.24.3
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency confli
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is incompatible.
Successfully installed aiohttp-3.8.1 aiosignal-1.2.0 async-timeout-4.0.2 asynctest-0.13.0 datasets-2.3.2 frozenlist-1.3.0 fsspec-2022.5.0 huggingface-hub-0.8.1 multidict-
```

Обучим модель на задачу суммаризации

```
# !pip install datasets
```

```
!ls
```

```
sample_data
```

```
from datasets import load_dataset

dataset_train = load_dataset('IlyaGusev/gazeta', revision="v1.0", split= 'train[:10%]')
dataset_test = load_dataset('IlyaGusev/gazeta', revision="v1.0", split= 'test[:10%]')
```

```
Downloading builder script: 100%  2.98k/2.98k [00:00<00:00, 81.5kB/s]
Downloading metadata: 100%  3.87k/3.87k [00:00<00:00, 119kB/s]
No config specified, defaulting to: gazeta/default
Downloading and preparing dataset gazeta/default (download: 545.11 MiB, generated: 542.44 MiB, post-processed: Unknown size, total: 1.06 GiB) to /root/.cache/huggingface/dat
Downloading data files: 100%  3/3 [00:14<00:00, 3.91s/it]
Downloading data: 100%  471M/471M [00:10<00:00, 21.8MB/s]
Downloading data: 100%  48.6M/48.6M [00:01<00:00, 29.5MB/s]
Downloading data: 100%  52.1M/52.1M [00:01<00:00, 43.8MB/s]
Extracting data files: 100%  3/3 [00:00<00:00, 36.08it/s]
Generating train split: 99%  51660/52400 [00:12<00:00, 3757.25 examples/s]
Generating test split: 91%  5276/5770 [00:00<00:00, 7905.92 examples/s]
Generating validation split: 97%  5128/5265 [00:01<00:00, 4336.33 examples/s]
Dataset gazeta downloaded and prepared to /root/.cache/huggingface/datasets/IlyaGusev___gazeta/default/1.0.0/ef9349c3c0f3112ca4036520d76c4bc1b8a79d30bc29643c6cae5a094d44e457
No config specified, defaulting to: gazeta/default
Reusing dataset gazeta (/root/.cache/huggingface/datasets/IlyaGusev___gazeta/default/1.0.0/ef9349c3c0f3112ca4036520d76c4bc1b8a79d30bc29643c6cae5a094d44e457)
```

```
dataset_train
```

```
Dataset({
    features: ['text', 'summary', 'title', 'date', 'url'],
    num_rows: 5240
})
```

```
dataset_test
```

```
Dataset({
    features: ['text', 'summary', 'title', 'date', 'url'],
    num_rows: 577
})
```

```
dataset_test['summary'][0]
```

```
'В NASA назвали четыре миссии в дальний космос, которые в этом десятилетии могут быть запущены американцами. Среди них — две миссии по изучению Венеры, полет к спутнику Юпитера и экспедиция к Тритону, спутнику Нептуна.'
```

```
dataset_test['title'][0]
```

```
'Венера, Ио или Тритон: куда полетит NASA'
```

```
model_name = "IlyaGusev/rut5_base_sum_gazeta"


def len_tok(text):
    return len(text.split())


max_len_sum, max_len_tl = max(map(len_tok, dataset_train['summary'])), max(map(len_tok, dataset_train['title']))
max_len_sum, max_len_tl
```

```
(75, 18)
```

```
max_len_sum, max_len_tl = 60, 15


from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained(model_name)

def tokenize(batch):
    tokenized_input = tokenizer(batch['summary'], padding='max_length', truncation=True, max_length=max_len_sum)
    tokenized_label = tokenizer(batch['title'], padding='max_length', truncation=True, max_length=max_len_tl)

    tokenized_input['labels'] = tokenized_label['input_ids']
    return tokenized_input

dataset_train = dataset_train.map(tokenize, batched=True, batch_size=8)
dataset_test = dataset_test.map(tokenize, batched=True, batch_size=8)

dataset_train.set_format('numpy', columns=['input_ids', 'attention_mask', 'labels'])
dataset_test.set_format('numpy', columns=['input_ids', 'attention_mask', 'labels'])
```

```
Downloading: 100%  279/279 [00:00<00:00, 8.12kB/s]
Downloading: 100%  808k/808k [00:00<00:00, 11.8MB/s]
Downloading: 100%  1.25M/1.25M [00:00<00:00, 16.0MB/s]
Downloading: 100%  65.0/65.0 [00:00<00:00, 2.41kB/s]
Parameter 'function'=<function tokenize at 0x7efcb0410ef0> of the transform datasets.arrow_dataset.Dataset._map_single couldn't be hashed properly, a random hash was used ir
100%  655/655 [00:03<00:00, 199.44ba/s]
100%  73/73 [00:00<00:00, 174.88ba/s]
```

```
dataset_train.save_to_disk('gazeta/train')
dataset_test.save_to_disk('gazeta/test')


!ls
```

```
gazeta  sample_data
```

```
from transformers import T5ForConditionalGeneration, Trainer, TrainingArguments


model = T5ForConditionalGeneration.from_pretrained(model_name)
```

```
Downloading: 100%  766/766 [00:00<00:00, 20.8kB/s]
Downloading: 100%  932M/932M [00:17<00:00, 62.4MB/s]
```

```
output_dir = 'gazeta/output'

training_args = TrainingArguments(
    output_dir=output_dir,
    num_train_epochs=10,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    eval_accumulation_steps=1, # Number of eval steps to keep in GPU (the higher, the mor vRAM used)
    prediction_loss_only=True, # If I need co compute only loss and not other metrics, setting this to true will use less RAM
    learning_rate=0.00001,
    evaluation_strategy='steps', # Run evaluation every eval_steps
    save_steps=1000, # How often to save a checkpoint
    save_total_limit=1, # Number of maximum checkpoints to save
    remove_unused_columns=True, # Removes useless columns from the dataset
    run_name='run_gazeta', # Wandb run name
    logging_steps=500, # How often to log loss to wandb
    eval_steps=500, # How often to run evaluation on the val_set
    logging_first_step=False, # Whether to log also the very first training step to wandb
    load_best_model_at_end=True, # Whether to load the best model found at each evaluation.
    metric_for_best_model="loss", # Use loss to evaluate best model.
    greater_is_better=False # Best model is the one with the lowest loss, not highest.
)


%%time

# Обучение. У нас 10 эпох.
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset_train,
    eval_dataset=dataset_test
```

```
)

trainer.train()
```

```
The following columns in the training set don't have a corresponding argument in `T5ForConditionalGeneration.forward` and have been ignored: url, text, title, date, summary
/usr/local/lib/python3.7/dist-packages/transformers/optimization.py:310: FutureWarning: This implementation of AdamW is deprecated and will be removed in a future version. U
  FutureWarning,
***** Running training *****
  Num examples = 5240
  Num Epochs = 10
  Instantaneous batch size per device = 8
  Total train batch size (w. parallel, distributed & accumulation) = 8
  Gradient Accumulation steps = 1
  Total optimization steps = 6550
/usr/local/lib/python3.7/dist-packages/transformers/data/data_collator.py:131: UserWarning: Creating a tensor from a list of numpy.ndarrays is extremely slow. Please conside
  batch[k] = torch.tensor([f[k] for f in features])
```



```
[6550/6550 28:54, Epoch 10/10]
```

```
trainer.save_model(output_dir + '/model')
```

```
Saving model checkpoint to gazeta/output/model
Configuration saved in gazeta/output/model/config.json
Model weights saved in gazeta/output/model/pytorch_model.bin
```

```
INX = 100
print("SUMMARY: | {}".format(dataset_test['summary'][INX]))
print("TITLE: | {}".format(dataset_test['title'][INX]))
```

```
SUMMARY: | Российские фрегаты «Адмирал Макаров» и «Адмирал Григорович» с крылатыми ракетами «Калибр-НК» проходят через турецкие проливы Босфор и Дарданеллы в направлении Сре
TITLE: | Снова в Сирию: российские фрегаты идут в Средиземное море
```

```
device = "cuda"
```

```
5000        2.401000        2.794934
```

```
import torch

input_text = dataset_test['summary'][INX]

with torch.no_grad():
    tokenized_text = tokenizer(input_text, truncation=True, padding=True, return_tensors='pt')

    source_ids = tokenized_text['input_ids'].to(device, dtype = torch.long)
    source_mask = tokenized_text['attention_mask'].to(device, dtype = torch.long)

    generated_ids = model.generate(
        input_ids = source_ids,
        attention_mask = source_mask,
        max_length=512,
        num_beams=7,
        temperature = 1.3,
        repetition_penalty=1,
        length_penalty=1,
        early_stopping=True,
        no_repeat_ngram_size=2  # количество повторов n-грамм > 2 запрещено.
    )

    # Параметры подбираются эксперементально

    pred = tokenizer.decode(generated_ids[0], skip_special_tokens=True, clean_up_tokenization_spaces=True)

print("\noutput:\n" + pred)
```

```
 Asking to truncate to max_length but no maximum length is provided and the model has no predefined maximum length. Default to no truncation.

 output:
 «Адмирал Григорович» и «Калибр-НК»
```

```
INX = 0
print("SUMMARY: | {}".format(dataset_test['summary'][INX]))
print("TITLE: | {}".format(dataset_test['title'][INX]))

input_text = dataset_test['summary'][INX]

with torch.no_grad():
    tokenized_text = tokenizer(input_text, truncation=True, padding=True, return_tensors='pt')

    source_ids = tokenized_text['input_ids'].to(device, dtype = torch.long)
    source_mask = tokenized_text['attention_mask'].to(device, dtype = torch.long)

    generated_ids = model.generate(
        input_ids = source_ids,
        attention_mask = source_mask,
        max_length=512,
        num_beams=7,
        temperature = 1.3,
        repetition_penalty=1,
        length_penalty=1,
        early_stopping=True,
        no_repeat_ngram_size=2  # количество повторов n-грамм > 2 запрещено.
    )

    # Параметры подбираются эксперементально
    pred = tokenizer.decode(generated_ids[0], skip_special_tokens=True, clean_up_tokenization_spaces=True)

print("\noutput:\n" + pred)
```

```
 SUMMARY: | В NASA назвали четыре миссии в дальний космос, которые в этом десятилетии могут быть запущены американцами. Среди них — две миссии по изучению Венеры, полет к спу
 TITLE: | Венера, Ио или Тритон: куда полетит NASA

 output:
 Запущены четыре миссии
```

```
Model weights saved in gazeta/output/checkpoint-6000/pytorch_model.bin
```

Вывод

Модель сформировала достаточно приемлемый заголовок.

Правда во втором случае он сообщает о событии как о случившемся, хотя оно ещё не произошло.

```
Loading best model from gazeta/output/checkpoint-6000 (score: 2.788262367248535).
CPU times: user 25min 50s, sys: 2min 10s, total: 28min 1s
Wall time: 29min 6s
```