Командный практический тур

Задача командного тура

В командной части заключительного этапа участникам было предложено решить актуальную для научного сообщества задачу по обработке естественного языка. Длительность командного этапа — 3 дня (14,5 астрономических часов).

Участникам был предоставлен датасет, состоящий из новостных вырезок, в которых описано некоторое событие и содержалась именованная сущность. Каждый текст сопровождался предложением с пропуском, где нужно добавить эту именованную сущность из списка вариантов. Данные варианты были извлечены из текста и могли повторяться. Правильный выбор, извлеченный из любого места текста, считался правильным, т. е. правильных ответов могло быть несколько. Главное условие: чтобы краткое содержание было осмысленным.

Актуальность задачи

Решение задач на обработку естественного языка, аналогичные этой, где алгоритмы искусственного интеллекта должны учитывать не только доступную информацию, но и здравый смысл, сейчас особенно интересуют российское и международное научное сообщество, так как без их успешного решения невозможно создание развитых интеллектуальных систем и сильного искусственного интеллекта.

Подобные алгоритмы также активно применяются в бизнесе при создании автоматизированных колл-центров и сервисов клиентской поддержки, в чат-ботах, голосовых помощниках и поисковых машинах.

Описание данных

Для решения задачи предоставлен датасет из 70000+ текстов, которые представляют собой вырезки из новостей. Все текстовые примеры были собраны из открытых источников, а затем автоматически отфильтрованы с помощью QA-систем, чтобы не допустить проникновения очевидных вопросов в набор данных. Затем тексты были отфильтрованы по частоте IPM содержащихся слов и, наконец, просмотрены вручную.

В архиве доступно 3 файла:

- train и val — для обучения модели и настройки параметров;
- test — проверочный файл, в котором нужно сделать предсказание и загрузить на платформу для проверки.

Скачать данные: https://onti2020.ai-academy.ru/task/rucos.zip

Пример данных:

```
{
   'source': 'Lenta',
   'passage': {
        'text':
```

```
            Институт немецкой экономики задался вопросом: действительно ли жилье
             ↪ и офисы в ФРГ скупают инвесторы из других стран. Эксперты
             ↪ получили недвусмысленный ответ и рассказали о нем DW. @header
             ↪ Учитывались только трансграничные сделки\nСледует учесть, что в
             ↪ исследовании IW использовались данные только о трансграничных
             ↪ инвестиционных потоках, когда сделки совершались на деньги,
             ↪ поступившие, например, непосредственно из России. \"Если
             ↪ российский инвестор сначала основал в Германии фирму, а уже
             ↪ потом с ее счета оплатил покупку коммерческой недвижимости, то
             ↪ это в нашу статистику не попало\", - предупредил эксперт. К тому
             ↪ же обращает на себя внимание резкая активизация инвесторов с
             ↪ Кипра. И вот в 2011 году с маленького острова, на котором широко
             ↪ представлен российский капитал, вдруг поступили сразу 7
             ↪ миллионов евро - столько же, сколько из Испании.\n@highlight\nВ
             ↪ Германии за сутки выявлено более 100 новых заражений
             ↪ коронавирусом\n@highlight\nТысячи демонстрантов в Гамбурге
             ↪ выступили за прием беженцев\n@highlight\nКомментарий: Россия
             ↪ накануне эпидемии - виноватые назначены заранее',
        'entities': [
            {'start': 79, 'end': 82, 'text': 'ФРГ'},
            {'start': 177, 'end': 179, 'text': 'DW'},
            {'start': 416, 'end': 422, 'text': 'России'},
            {'start': 468, 'end': 476, 'text': 'Германии'},
            {'start': 678, 'end': 683, 'text': 'Кипра'},
            {'start': 839, 'end': 846, 'text': 'Испании'},
            {'start': 861, 'end': 869, 'text': 'Германии'},
            {'start': 962, 'end': 970, 'text': 'Гамбурге'},
            {'start': 1023, 'end': 1029, 'text': 'Россия'}
        ]
    },
    'qas': [
        {
            'query': ' В беседе с @placeholder профессор Фогтлендер особо указал
             ↪ на то, что в крупных городах Германии, прежде всего, в Берлине,
             ↪ доля иностранцев в сделках на рынке недвижимости существенно
             ↪ выше, чем в среднем по стране: \"Там она составляет уже не 1 и
             ↪ не 5, а, по-видимому, 10-15 процентов\".',
            'answers': [
                {'start': 177, 'end': 179, 'text': 'DW'}
            ],
            'idx': 0
        }
    ],
    'idx': 0
}
```

Метрика и формат решений

На платформу для проверки нужно загрузить jsonl-файл, соответствующий примеру, в котором указан id вопроса и ответ на него.

Качество решения оценивалось автоматически по метрике F1.

Базовое решение

Решение задачи RuCoS

Информация о задаче: https://russiansuperglue.com/tasks/task_info/RuCoS.

Загрузка данных.

```
!wget https://russiansuperglue.com/tasks/download/RuCoS -O RuCoS.zip
!unzip RuCoS.zip
```

```
--2021-04-28 17:27:23--  https://russiansuperglue.com/tasks/download/RuCoS
Resolving russiansuperglue.com (russiansuperglue.com)... 37.18.107.48
Connecting to russiansuperglue.com (russiansuperglue.com)|37.18.107.48|:443...
↪  connected.
HTTP request sent, awaiting response... 200 OK
Length: 56208297 (54M) [application/zip]
Saving to: ‘RuCoS.zip’

RuCoS.zip           100%[===================>]  53.60M  6.25MB/s    in 8.8s

2021-04-28 17:27:32 (6.11 MB/s) - ‘RuCoS.zip’ saved [56208297/56208297]

Archive:  RuCoS.zip
   creating: RuCoS/
  inflating: RuCoS/train.jsonl
   creating: __MACOSX/
   creating: __MACOSX/RuCoS/
  inflating: __MACOSX/RuCoS/._train.jsonl
  inflating: RuCoS/.DS_Store
  inflating: __MACOSX/RuCoS/._.DS_Store
  inflating: RuCoS/test.jsonl
  inflating: __MACOSX/RuCoS/._test.jsonl
  inflating: RuCoS/val.jsonl
  inflating: __MACOSX/RuCoS/._val.jsonl
  inflating: __MACOSX/._RuCoS
```

Установка зависимостей.

```
!pip install -q transformers seaborn
!pip install nltk
!pip install tqdm
!pip install pandas
!pip install numpy
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages
↪  (3.2.5)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
↪  (from nltk) (1.15.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
↪  (4.41.1)
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages
↪  (1.1.5)
Requirement already satisfied: python-dateutil>=2.7.3 in
↪  /usr/local/lib/python3.7/dist-packages (from pandas) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in
↪  /usr/local/lib/python3.7/dist-packages (from pandas) (2018.9)
Requirement already satisfied: numpy>=1.15.4 in
↪  /usr/local/lib/python3.7/dist-packages (from pandas) (1.19.5)
Requirement already satisfied: six>=1.5 in
↪  /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas)
↪  (1.15.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
↪  (1.19.5)
```

Подготовка данных

Подключаем библиотеки.

```
import pandas as pd
import numpy as np
import json
from tqdm.notebook import tqdm
import re
import warnings
import seaborn as sns

from nltk import sent_tokenize
import nltk
```

Инициализация и основные константы.

```
nltk.download('punkt')
tqdm.pandas()

warnings.simplefilter("ignore")

SEED = 1337
TRAIN_BATCH_SIZE = 64
VAL_BATCH_SIZE = 64
SEQ_LEN = 220
MODEL_NAME = 'DeepPavlov/rubert-base-cased'
device = 'cuda:0'
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Считываем данные.

```
with open('RuCoS/train.jsonl', 'r', encoding='utf-8') as f:
    train_content = f.read()
```

```
with open('RuCoS/test.jsonl', 'r', encoding='utf-8') as f:
    test_content = f.read()

with open('RuCoS/val.jsonl', 'r', encoding='utf-8') as f:
    val_content = f.read()

train_dict = [json.loads(jline) for jline in train_content.splitlines()]
test_dict = [json.loads(jline) for jline in test_content.splitlines()]
val_dict = [json.loads(jline) for jline in val_content.splitlines()]
```

Для ускорения обучения загрузим лишь часть данных. Для итогового решения надо использовать весь набор.

```
MAX_TRAIN = 30000
MAX_VAL = 2000

train_dict = train_dict[:MAX_TRAIN]
val_dict = val_dict[:MAX_VAL]
```

Обработаем тренировочные данные чтобы далее с ними было удобнее работать.

```
train_prettified = {
    'text_id': [],
    'q_id': [],
    'a_id': [],
    'text': [],
    'query': [],
    'end': [],
    'start': [],
    'qtext': [],
    'label': []
}

for i, el in enumerate(tqdm(train_dict)):
    passage = el['passage']

    text_id = el['idx']
    q_id = el['qas'][0]['idx']
    text = passage['text']
    query = el['qas'][0]['query']

    a_id = 0
    for ent in passage['entities']:

        label = 0
        for ans in el['qas'][0]['answers']:
            if ans['end'] == ent['end'] and ans['start'] == ent['start']:
                label = 1
                break

        qtext = text[ent['start']:ent['end']]
        train_prettified['text_id'].append(text_id)
```

```
        train_prettified['q_id'].append(q_id)
        train_prettified['a_id'].append(a_id)
        train_prettified['text'].append(text)
        train_prettified['query'].append(query)
        train_prettified['end'].append(ent['end'])
        train_prettified['start'].append(ent['start'])
        train_prettified['qtext'].append(qtext)
        train_prettified['label'].append(label)

        a_id += 1

print('Размер обучающей выборки:', len(train_prettified['text_id']), '| Верных',
↪  sum(train_prettified['label'])/len(train_prettified['text_id']))
```

```
HBox(children=(FloatProgress(value=0.0, max=30000.0), HTML(value='')))

Размер обучающей выборки: 400890 | Верных 0.17734041757090474
```

Аналогичная обработка для validation выборки и test.

```
val_prettified = {
    'text_id': [],
    'q_id': [],
    'a_id': [],
    'text': [],
    'query': [],
    'end': [],
    'start': [],
    'qtext': [],
    'label': []
}

for i, el in enumerate(tqdm(val_dict)):
    passage = el['passage']

    text_id = el['idx']
    q_id = el['qas'][0]['idx']
    text = passage['text']
    query = el['qas'][0]['query']

    a_id = 0
    for ent in passage['entities']:

        label = 0
        for ans in el['qas'][0]['answers']:
            if ans['end'] == ent['end'] and ans['start'] == ent['start']:
                label = 1
                break
        qtext = text[ent['start']:ent['end']]
        val_prettified['text_id'].append(text_id)
        val_prettified['q_id'].append(q_id)
        val_prettified['a_id'].append(a_id)
        val_prettified['text'].append(text)
```

```
        val_prettified['query'].append(query)
        val_prettified['end'].append(ent['end'])
        val_prettified['start'].append(ent['start'])
        val_prettified['qtext'].append(qtext)
        val_prettified['label'].append(label)

        a_id += 1

print('Размер валидационной выборки', len(val_prettified['text_id']), '|
↪  Верных', sum(val_prettified['label'])/len(val_prettified['text_id']))
```

```
HBox(children=(FloatProgress(value=0.0, max=2000.0), HTML(value='')))
```

```
Размер валидационной выборки 26139 | Верных 0.15008225257278396
```

```
test_prettified = {
    'text_id': [],
    'q_id': [],
    'a_id': [],
    'text': [],
    'query': [],
    'end': [],
    'start': [],
    'qtext': []
}

for i, el in enumerate(tqdm(test_dict)):
    passage = el['passage']

    text_id = el['idx']
    q_id = el['qas'][0]['idx']
    text = passage['text']
    query = el['qas'][0]['query']

    a_id = 0
    for ent in passage['entities']:
        qtext = text[ent['start']:ent['end']]
        test_prettified['text_id'].append(text_id)
        test_prettified['q_id'].append(q_id)
        test_prettified['a_id'].append(a_id)
        test_prettified['text'].append(text)
        test_prettified['query'].append(query)
        test_prettified['end'].append(ent['end'])
        test_prettified['start'].append(ent['start'])
        test_prettified['qtext'].append(qtext)

        a_id += 1
print('Размер тестовой выборки:', len(test_prettified['text_id']))
```

```
HBox(children=(FloatProgress(value=0.0, max=7257.0), HTML(value='')))
```

```
Размер тестовой выборки: 96996
```

Теперь, обработаем конструкции вида @name в датасете.

Создадим все возможные запросы с @placeholder замененным на возможный ответ.

```
def prep(x):
    x =  re.split(r'\n@highlight\n', x)[0].replace('\n', ' ').replace('@header',
    ↪  '')
    return x

    %%time
train = pd.DataFrame(train_prettified)
test = pd.DataFrame(test_prettified)
val = pd.DataFrame(val_prettified)

train = train[train['query'].apply(lambda x: x.count('@placeholder') ==
↪  1)].reset_index(drop=True)

train['text'] = train['text'].progress_apply(prep)
val['text'] = val['text'].progress_apply(prep)
test['text'] = test['text'].progress_apply(prep)

# необходимая замена
train['query_prep'] = train[['query', 'qtext']].apply(lambda x:
↪  x['query'].replace('@placeholder', x['qtext']), axis=1)
test['query_prep'] = test[['query', 'qtext']].apply(lambda x:
↪  x['query'].replace('@placeholder', x['qtext']), axis=1)
val['query_prep'] = val[['query', 'qtext']].apply(lambda x:
↪  x['query'].replace('@placeholder', x['qtext']), axis=1)

# удалим дубликаты
train = train.drop_duplicates(['text', 'query_prep'])
test = test.drop_duplicates(['text', 'query_prep'])
val = val.drop_duplicates(['text', 'query_prep'])

HBox(children=(FloatProgress(value=0.0, max=393952.0), HTML(value='')))

HBox(children=(FloatProgress(value=0.0, max=26139.0), HTML(value='')))

HBox(children=(FloatProgress(value=0.0, max=96996.0), HTML(value='')))

CPU times: user 12.5 s, sys: 562 ms, total: 13 s
Wall time: 13.1 s
```

Посмотрим, что получилось.

```
train[['text', 'query', 'query_prep']].head()
```

	text	query	query_prep
0	Наблюдатели полагают, что подоплекой теракта в...	Кроме того, серьезным вызовом для России стано...	Кроме того, серьезным вызовом для России стано...
2	Наблюдатели полагают, что подоплекой теракта в...	Кроме того, серьезным вызовом для России стано...	Кроме того, серьезным вызовом для России стано...
4	Наблюдатели полагают, что подоплекой теракта в...	Кроме того, серьезным вызовом для России стано...	Кроме того, серьезным вызовом для России стано...
6	Наблюдатели полагают, что подоплекой теракта в...	Кроме того, серьезным вызовом для России стано...	Кроме того, серьезным вызовом для России стано...
7	Наблюдатели полагают, что подоплекой теракта в...	Кроме того, серьезным вызовом для России стано...	Кроме того, серьезным вызовом для России стано...

```
train[['text', 'query', 'query_prep']].iloc[0]['query'], train[['text', 'query',
↪  'query_prep']].iloc[0]['query_prep'], train[['text', 'query',
↪  'query_prep']].iloc[1]['query_prep']
```

```
('Кроме того, серьезным вызовом для России становится стремительно развивающийся
↪  Китай.Еще в понедельник @placeholder в рамках спора о системе
↪  противоракетной обороны пригрозил размещением дополнительных ракетных
↪  комплексов.',
 'Кроме того, серьезным вызовом для России становится стремительно развивающийся
 ↪  Китай.Еще в понедельник Домодедово в рамках спора о системе противоракетной
 ↪  обороны пригрозил размещением дополнительных ракетных комплексов.',
 'Кроме того, серьезным вызовом для России становится стремительно развивающийся
 ↪  Китай.Еще в понедельник Süddeutsche Zeitung в рамках спора о системе
 ↪  противоракетной обороны пригрозил размещением дополнительных ракетных
 ↪  комплексов.')
```

Подготовка модели SBERT

Инициализация и основные константы.

```
from transformers import AutoTokenizer, AutoModel,
↪  get_linear_schedule_with_warmup

import torch
from torch import nn, optim
from torch.utils.data import Dataset, DataLoader
import torch.nn.functional as F
import random
import os
from sklearn.metrics import f1_score
from collections import defaultdict

from torch.cuda.amp import autocast, GradScaler
```

Установим состояние генераторов случайных чисел для воспроизводимости результата.

```
def set_seed(seed = 42, set_torch=True):
    random.seed(seed)
    np.random.seed(seed)
    os.environ["PYTHONHASHSEED"] = str(seed)
    if set_torch:
        torch.manual_seed(seed)
        torch.cuda.manual_seed(seed)
        torch.backends.cudnn.deterministic = True
        torch.backends.cudnn.benchmark = False
set_seed(SEED)
```

Загрузим токенайзер и саму модель. Будем использовать уже предобученые модели.

```
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
bert_model = AutoModel.from_pretrained(MODEL_NAME)

# Класс для работы с датасетом в формате, принимаемым моделью
class CQDataset(Dataset):
    def __init__(self, texts, queries, targets, tokenizer, max_len=512):
        self.texts = texts
        self.queries = queries
        self.targets = targets
        self.tokenizer = tokenizer
        self.max_len = max_len

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, item):
        text = str(self.texts[item])
        query = str(self.queries[item])
        target = self.targets[item]

        tt = self.tokenizer.tokenize(text)
        qt = self.tokenizer.tokenize(query)

        # Мы используем DeepPavlov RuBERT поэтому ввод должен иметь вид
        # [CLS] context_tokens [SEP] query_tokens [SEP]
        tokens = tt + ['[SEP]'] + qt
        tokens = tokens[-self.max_len+2:]
        tokens =
        ↪  ['[CLS]']+tokens+['[SEP]']+['[PAD]']*(self.max_len-2-len(tokens))

        attention_mask = [1*(token!='[PAD]') for token in tokens]
        fe = tokens.index('[SEP]')
        token_type_ids = [(i>=fe)*1 for i in range(len(tokens))]
        input_ids = self.tokenizer.convert_tokens_to_ids(tokens)

        input_ids = torch.tensor(input_ids, dtype=torch.long)
        attention_mask = torch.tensor(attention_mask, dtype=torch.long)
        token_type_ids = torch.tensor(token_type_ids, dtype=torch.long)
        target = torch.tensor(target, dtype=torch.float16)

        return {
          'input_ids': input_ids,
          'attention_mask': attention_mask,
          'token_type_ids': token_type_ids,
          'target': target
        }

def create_data_loader(df, tokenizer, max_len, batch_size, shuffle=True):
    ds = CQDataset(
        df['text'].values,
        df['query_prep'].values,
        df['label'].values,
        tokenizer,
```

```
        max_len
    )

    return DataLoader(
        ds,
        batch_size=batch_size,
        num_workers=2,
        shuffle=shuffle
    )
```

Подготовим загрузчики данных на основании датасетов.

```
train_dl = create_data_loader(train, tokenizer, SEQ_LEN, TRAIN_BATCH_SIZE)
val_dl = create_data_loader(val, tokenizer, SEQ_LEN, VAL_BATCH_SIZE,
↪  shuffle=False)
```

Будем использовать свой классификатор, принимающий основную модель.

```
class CQClassifier(nn.Module):
    def __init__(self, base_model, units=768):
        super(CQClassifier, self).__init__()
        self.bert = base_model
        self.mlp_head = nn.Sequential(
            nn.Dropout(0.2),
            nn.Linear(units, 1)
        )
    def forward(self, input_ids, attention_mask, token_type_ids):
        # INPUT
        bert_out = self.bert(
            input_ids=input_ids,
            token_type_ids=token_type_ids,
            attention_mask=attention_mask
        )
        x = bert_out['pooler_output']
        x = self.mlp_head(x).view(-1)
        return x

model = CQClassifier(bert_model, units=768)
model = model.to(device)

print('Итоговое число параметров:', sum(p.numel() for p in model.parameters()))
print('Обучаемые параметры:', sum(p.numel() for p in model.parameters() if
↪  p.requires_grad))
```

```
Итоговое число параметров: 177854209
Обучаемые параметры: 177854209
```

Проверим состояние видеокарты.

```
! nvidia-smi
```

```
Wed Apr 28 17:28:18 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.19.01    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   47C    P0    38W / 250W |   1629MiB / 16280MiB |      3%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Обучение

```
EPOCHS = 1
num_train_steps = len(train_dl) * EPOCHS

def configure_optimizers(model, lr):
    # В качестве оптимизатора AdamW с decay параметров
    param_optimizer = list(model.named_parameters())
    no_decay = ["bias", "gamma", "beta"]
    optimizer_grouped_parameters = [
        {
            "params": [p for n, p in param_optimizer if not any(nd in n for nd
            ↪  in no_decay)],
            "weight_decay_rate": 0.01
        },
        {
            "params": [p for n, p in param_optimizer if any(nd in n for nd in
            ↪  no_decay)],
            "weight_decay_rate": 0.0
        },]
    optimizer = optim.AdamW(
                optimizer_grouped_parameters,
                lr=lr,
    )
    return optimizer

scaler = GradScaler()
optimizer = configure_optimizers(model, 2e-5)
loss_fn = nn.BCEWithLogitsLoss().to(device)
# Для улучшения также применяем warm up
scheduler = get_linear_schedule_with_warmup(optimizer, int(num_train_steps*.1),
↪  num_train_steps)
```

Функция для обучения на одну эпоху.

```
def train_epoch(model, data_loader, loss_fn, optimizer, device, scheduler):
    model = model.train()

    losses = []
    pred = []
    true = []

    pbar = tqdm(data_loader)
    for d in pbar:
        optimizer.zero_grad()

        # Перенос информации на видеокарту
        input_ids = d['input_ids'].to(device)
        attention_mask = d['attention_mask'].to(device)
        token_type_ids = d['token_type_ids'].to(device)
        targets = d["target"].to(device)

        with autocast():
            outputs = model(
                input_ids=input_ids,
                attention_mask=attention_mask,
                token_type_ids=token_type_ids
            )

            true += list(targets.cpu().numpy())
            pred += list(torch.round(F.sigmoid(outputs.detach())).cpu().numpy())

            loss = loss_fn(outputs, targets)
            losses.append(loss.detach().item())

        scaler.scale(loss).backward()
        scaler.unscale_(optimizer)
        nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
        scaler.step(optimizer) # выполняем шаг оптимизации
        scaler.update()
        scheduler.step()
        pbar.set_postfix({'loss': loss.detach().item()}) # будем выводить
        ↪  текущую loss чтобы следить за прогрессом

    return np.mean(losses), f1_score(true, pred, average='macro')

# аналогичная train функция, но для оценки
def eval_epoch(model, data_loader, loss_fn, device):
    model = model.eval()

    losses = []
    pred = []
    true = []

    with torch.no_grad():
        for d in tqdm(data_loader):
            input_ids = d['input_ids'].to(device)
```

```
            attention_mask = d['attention_mask'].to(device)
            token_type_ids = d['token_type_ids'].to(device)
            targets = d["target"].to(device)

            outputs = model(
                input_ids=input_ids,
                attention_mask=attention_mask,
                token_type_ids=token_type_ids
            )

            true += list(targets.cpu().numpy())
            pred += list(torch.round(F.sigmoid(outputs.detach())).cpu().numpy())

            loss = loss_fn(outputs, targets)
            losses.append(loss.item())

    return np.mean(losses), f1_score(true, pred, average='macro')
```

Запустим обучение.

```
%%time

print('Number of train steps:', num_train_steps)
for epoch in range(EPOCHS):
    print(f'Epoch {epoch + 1}/{EPOCHS}:')
    print('-' * 10)

    train_loss, train_f1 = train_epoch(
        model,
        train_dl,
        loss_fn,
        optimizer,
        device,
        scheduler
    )
    print(f'    loss: {train_loss:.4f}, f1: {train_f1:.4f} |', end=' ')

    val_loss, val_f1 = eval_epoch(
        model,
        val_dl,
        loss_fn,
        device
    )
    print(f'val_loss: {val_loss:.4f}, val_f1: {val_f1:.4f}')
    torch.save(model.state_dict(), f'bert_model_{epoch+1}.bin')
```

```
Number of train steps: 4681
Epoch 1/1:
----------
HBox(children=(FloatProgress(value=0.0, max=4681.0), HTML(value='')))

    loss: 0.2758, f1: 0.7118 |
HBox(children=(FloatProgress(value=0.0, max=292.0), HTML(value='')))
```

```
val_loss: 0.1845, val_f1: 0.7976
CPU times: user 1h 14s, sys: 45min 38s, total: 1h 45min 52s
Wall time: 1h 46min 19s
```

Предсказание на тестовой выборке

```
test['label'] = 0
```

Функция предсказания на данных, используя модель.

```
def predict(dl, model):
    model = model.eval()
    preds = []
    with torch.no_grad(): # без градиентов - только для предсказания
        for d in tqdm(dl):
            input_ids = d['input_ids'].to(device)
            attention_mask = d['attention_mask'].to(device)
            token_type_ids = d['token_type_ids'].to(device)

            outputs = model(
                input_ids=input_ids,
                attention_mask=attention_mask, token_type_ids=token_type_ids
            )
            outputs = (F.sigmoid(outputs).cpu()).tolist()
            preds += outputs
    return preds

test_dl = create_data_loader(test, tokenizer, SEQ_LEN, VAL_BATCH_SIZE,
↪  shuffle=False)
test['pred'] = predict(test_dl, model)
```

```
HBox(children=(FloatProgress(value=0.0, max=1059.0), HTML(value='')))
```

Подготовка решения в необходимом формате.

```
submission = []

for text_idx in tqdm(test['text_id'].unique()):
    pdict = {'idx':int(text_idx)}
    tmp = test[test['text_id'] == text_idx]
    tmp = tmp.iloc[np.argmax(tmp['pred'])]
    pdict['start'] = int(tmp['start'])
    pdict['end'] = int(tmp['end'])
    pdict['text'] = str(tmp['qtext'])

    submission.append(pdict)
submission = [json.dumps(el) for el in submission]
submission = '\n'.join(submission)
```

```
HBox(children=(FloatProgress(value=0.0, max=7257.0), HTML(value='')))
```

Сохранение решения.

```
with open('submission.jsonl', 'w', encoding='utf-8') as f:
    f.write(submission)
```

Посмотрим на случайные строки.

```
pd.DataFrame(test).sample(10)[['text', 'query', 'pred']]
```

	text	query	pred
7413	Близкий к турецкому правительству фонд SETA оп...	При этом авторы исследования критикуют, в част...	0.002272
32866	Дочь россиянки, которую планировали госпитализ...	Россиянка поехала в больницу в поселке @placeh...	0.002103
82735	Бывшего командира украинского националистическ...	@placeholder предоставили русскоязычного адвок...	0.017532
28317	Комиссия ООН по разоружению снова перенесла на...	17 февраля глава МИД России Сергей Лавров заяв...	0.667688
31452	В Совет Федерации поступили результаты голосов...	Накануне Совет Федерации одобрил президентский...	0.005855
92523	Правительство Украины расширило действие режим...	По последним данным, на @placeholder зарегистр...	0.885366
83964	Самая молодая миллиардерша в мире и предприним...	Подписчики обратили внимание на палец @placeho...	0.047775
69178	Министр энергетики России Александр Новак расс...	Спрос не падает, у нас объемы экспорта угля в ...	0.001590
57605	Официальный представитель министерства иностра...	Помимо сторон конфликта, в берлинской конферен...	0.060163
21684	Сотрудники военной контрразведки ФСБ задержали...	По его словам, к нему в СИЗО пришел следовател...	0.006590