

# Урок 3

## Применение Transfer learning

### Задание

- Объясняем работу примера style transfer (красивые примеры тут, colab)
- Реализуйте классификацию эмоциональной окрашенности текстов при помощи объекта pipeline:
  1. Найдите тип задач Sentiment Analysis на [huggingface.co](https://huggingface.co)
  2. Найдите модель для русского языка (примеры: rubert-tiny2..., rubert-base...)

### Решение

Предполагаю, что объяснение примеров с "кошками" это опечатка, т.к. мы выполняли это задание в прошлой домашней работе.

В любом случае выбираю более сложное задание по эмоциональной окрашенности текстов.

[Аналогичную работу](#) выполнял на курсе "Разметка данных". Только там использовался чистый ML и на английском языке. Результат был ошеломительный. Машина делала предсказания, лучше меня.

С нейронными сетями такую работы не выполнял. Что, оказалось не простым решением, в особенности если использовать языковую модель BERT, у которой есть ограничение до 512 токенов. Т.е. должны быть тексты короткие, либо их нужно обрезать. Как показывает данная работа, Берта устарела и не годится для современных задач. Но, обо всем по порядку.

Необходимо определиться с датасетами, для уникальности выбрал отзывы тематики онлайн-образования с сайта [IRECOMMEND.RU](http://IRECOMMEND.RU). В данный момент, среди "отзовиков", портал находится на первом месте. Хоть он больше нацелен на отзывы по товарам, но присутствуют отзывы и на услуги, в том числе образовательные.

Выбор пал на irecommend.ru, еще по одной причине: помимо пятибалльной системы оценок, на нем присутствует бинарная классификация. Т.е. любой отзыв имеет помимо количества звезд еще лейбл в виде пальца вверх со словом "рекомендует" или палец вниз "не рекомендует". Данный подход, на других отзовиках не заметил.

### Получение данных

[IRECOMMEND](#) является чисто коммерческим проектом, где отзывы монетизируются и портал за каждый платит деньги. Следовательно, на безвозмездной основе не предоставляет в открытом доступе датасеты, либо публичные API.

Пришлось воспользоваться фреймворком scrapy для реализации скрейпера, способного получить и распарсить данные в нужный формат (CSV). Для последующей загрузки в юпитер-ноутбук.

8 датасетов с отзывами по популярным образовательным площадкам:

1. [Stepik](#)
2. [Geekbrains](#)
3. [SkillBox](#)
4. [SkillFactory](#)
5. [Netologiya](#)
6. [SkyEng](#)
7. [Yandex Practicum](#)
8. [Uch.Ru](#)

Формат датасета табулированный CSV, со следующими полями:

- `header` - заголовок отзыва
- `text` - текст отзыва
- `verdict` - классификация: "рекомендует" или "не рекомендует"
- `link` - ссылка на оригинальный отзыв

`header`

Заголовок отзыва содержит одно или два предложения. Заметил тенденцию, что часто в отрицательных отзывах присутствует только заголовок, т.е. человек настолько зол, что не хочет даже писать текст.

`text`

Самое объемное поле, размер может достигать 20 тысяч символов или примерно 2000 слов. Т.е. по этой причине BERT проигрывает, другим моделям, для современных нужд

`verdict`

1 - рекомендует, 0 - не рекомендует. Имеет только два значения

`link`

Ссылка на оригинальный отзыв. Часто в комментариях люди прикрепляют фотографии или скрины. Картинки из отзывов, по моему мнению в будущем, тоже должны подвергаться анализу. Т.к. отзыв может содержать только одно изображение, говорящее само за себя.

Ссылка на датасеты есть в самом коде, выложил данные на GitHub в открытый доступ.

## Импорт необходимых библиотек

Для того чтобы ноутбук работал, нужно установить необходимые библиотеки. Первая ячейка кода автоматически проверяет их наличие, в случае отсутствия таковых, произведет установку и перезапустит сеанс. Поэтому не следуют обращать внимания на предупреждение: "Ваша среда неожиданной прекратила работу". Нужно лишь второй раз запустить блокнот на исполнение.

```
In [1]: # Если нет нужных пакетов, они будут установлены, сеанс завершится, нужно будет стартовать снова
import importlib
packages = ['torch', 'accelerate', 'transformers']
# Проверка установки каждого пакета
for package in packages:
    try:
        importlib.import_module(package)
    except ImportError:
        if package == 'torch':
            !pip install torch
        if package == 'accelerate':
            !pip install accelerate -U
        if package == 'transformers':
            !pip install transformers[torch]
    # Перезапуск сеанса
    import os
    os.kill(os.getpid(), 9)
```

```
In [24]: import torch
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from transformers import BertForSequenceClassification, BertTokenizer, TrainingArguments, Trainer, TrainerCallback, EarlyStopping
from transformers import LongformerTokenizer, LongformerForSequenceClassification
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]  Unzipping tokenizers/punkt.zip.
```

```
Out[24]: True
```

Как писал ранее, встречаются отзывы только с заголовками, типа "Очень плохой курс" или "Не рекомендую компанию X". Чтобы текстовое поле не было пустым, функция ниже копирует содержимое заголовка в соответствующее пустое поле text.

```
In [3]: def copy_header_to_text(row):
    if pd.isna(row['text']) or row['text'] == '':
        return row['header']
    else:
        return row['text']
```

Загрузка датасета с 32 отзывами о Stepik

Самый маленький датасет, может потому что большинство курсов там бесплатны и студентов не "мотивируют" на написание отзыва :)

```
In [4]: st_dataset = pd.read_csv('https://github.com/allseenn/transfer/raw/main/03.Tasks/sait-stepik.csv', sep='\t', encoding='utf8')
st_dataset['text'] = st_dataset.apply(copy_header_to_text, axis=1)
```

Загрузка датасета с 516 отзывами о GeekBrains

Следующие два датасета самые большие. Это родственные компании, даже количество отзывов у них как близнецы 516.

```
In [5]: gb_dataset = pd.read_csv('https://github.com/allseenn/transfer/raw/main/03.Tasks/sait-geekbrains.csv', sep='\t', encoding='utf8')
gb_dataset['text'] = gb_dataset.apply(copy_header_to_text, axis=1)
```

Загрузка датасета с 516 отзывами о SkillBox

```
In [6]: sb_dataset = pd.read_csv('https://github.com/allseenn/transfer/raw/main/03.Tasks/sait-skillbox-onlain-shkola.csv', sep='\t', en
sb_dataset['text'] = sb_dataset.apply(copy_header_to_text, axis=1)
```

Загрузка датасета с 70 отзывами о SkillFactory

```
In [7]: sf_dataset = pd.read_csv('https://github.com/allseenn/transfer/raw/main/03.Tasks/sait-skillfactoryru.csv', sep='\t', encoding='
sf_dataset['text'] = sf_dataset.apply(copy_header_to_text, axis=1)
```

Загрузка датасета с 240 отзывами о Netologiya

```
In [8]: nt_dataset = pd.read_csv('https://github.com/allseenn/transfer/raw/main/03.Tasks/sait-netologiya.csv', sep='\t', encoding='utf8
nt_dataset['text'] = nt_dataset.apply(copy_header_to_text, axis=1)
```

Загрузка датасета с 364 отзывами о SkyEng

```
In [9]: sk_dataset = pd.read_csv('https://github.com/allseenn/transfer/raw/main/03.Tasks/sait-onlain-shkola-angliiskogo-yazyka-skyeng-1')
sk_dataset['text'] = sk_dataset.apply(copy_header_to_text, axis=1)
```

Загрузка датасета с 70 отзывами YandexPracticum

```
In [10]: ya_dataset = pd.read_csv('https://github.com/allseenn/transfer/raw/main/03.Tasks/kompyuternaya-programma-yandeks-praktikum.csv')
ya_dataset['text'] = ya_dataset.apply(copy_header_to_text, axis=1)
```

Загрузка датасета с 70 отзывами UchiRu

```
In [11]: uchi_dataset = pd.read_csv('https://github.com/allseenn/transfer/raw/main/03.Tasks/sait-uchiru-uchiru-interaktivnaya-obrazovate
uchi_dataset['text'] = uchi_dataset.apply(copy_header_to_text, axis=1)
```

## Предсказание с помощью Longformer

Т.к. мне хотелось сразу сделать предсказания большого поля text, то решил попробовать мультиязычную модель, поддерживающую длинные тексты.

```
In [ ]: # Загрузка предварительно обученной модели Longformer
model_name = 'allenai/longformer-base-4096'
tokenizer = LongformerTokenizer.from_pretrained(model_name)
model = LongformerForSequenceClassification.from_pretrained(model_name, num_labels=2)

# Создание функции для предсказания сентимента
def predict_sentiment(text):
    inputs = tokenizer(text, return_tensors='pt', truncation=True, padding=True, max_length=512)
    outputs = model(**inputs)
    predictions = torch.softmax(outputs.logits, dim=1)
    sentiment_label = 1 if predictions[0][1] > predictions[0][0] else 0 # 1 для положительного, 0 для отрицательного
    return sentiment_label
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
    warnings.warn(
vocab.json:  0%|          0.00/899k [00:00<?, ?B/s]
merges.txt:  0%|          0.00/456k [00:00<?, ?B/s]
tokenizer.json:  0%|          0.00/1.36M [00:00<?, ?B/s]
config.json:  0%|          0.00/694 [00:00<?, ?B/s]
```

```
pytorch_model.bin: 0%|          | 0.00/597M [00:00<?, ?B/s]
```

Some weights of LongformerForSequenceClassification were not initialized from the model checkpoint at allenai/longformer-base-4096 and are newly initialized: ['classifier.dense.bias', 'classifier.dense.weight', 'classifier.out\_proj.bias', 'classifier.out\_proj.weight']  
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

Оценка отзывов с помощью модели типа Longformer датасета Stepik

```
In [ ]: # Применение функции к столбцу текста в датасете  
st_dataset['predicted'] = st_dataset['text'].apply(predict_sentiment)
```

Initializing global attention on CLS token...

Input ids are automatically padded from 41 to 512 to be a multiple of `config.attention\_window`: 512

Вывод датасета Stepik вместе с новым столбцом predicted. Вроде бы Longformer справился?!?

```
In [ ]: st_dataset
```

Out[ ]:

	header	text	verdict	link	predicted
0	Вы все еще сомневаетесь? Начинайте учить Python!	Вот раньше как было? Дед был сапожником, отец ...	1	<a href="https://irecommend.ru/content/vy-vse-eshche-so...">https://irecommend.ru/content/vy-vse-eshche-so...</a>	1
1	Хорошая платформа для онлайн обучения. Много и...	– это образовательная платформа и конструктор ...	1	<a href="https://irecommend.ru/content/khoroshaya-platf...">https://irecommend.ru/content/khoroshaya-platf...</a>	1
2	Даёшь новые нейронные связи?	О таком ресурсе как "Степик" я узнала совершен...	1	<a href="https://irecommend.ru/content/daesh-novye-neir...">https://irecommend.ru/content/daesh-novye-neir...</a>	1
3	Самообразование - легко и бесплатно! Стоит обр...	У нас у всех есть разное хобби, а у меня есть ...	1	<a href="https://irecommend.ru/content/samoobrazovanie-...">https://irecommend.ru/content/samoobrazovanie-...</a>	1
4	Заработка на фотографиях автомобилей.500 или 3...	Всем привет! Меня зовут Света и я мама в декре...	1	<a href="https://irecommend.ru/content/zarabotok-na-fot...">https://irecommend.ru/content/zarabotok-na-fot...</a>	1
5	Подробный отчет о моем обучении, качестве подав...	Всем доброго дня! Я знаю точно, что многие, прежде...	0	<a href="https://irecommend.ru/content/rasskazyvayu-pod...">https://irecommend.ru/content/rasskazyvayu-pod...</a>	1
6	Лучший образовательный портал для тех, кто дей...	Я проходила курсы на разных обучающих порталах...	1	<a href="https://irecommend.ru/content/luchshii-obrazov...">https://irecommend.ru/content/luchshii-obrazov...</a>	1
7	Возможность учиться бесплатно в удобное время.	Нашла случайно в Google Play приложение Stepik...	1	<a href="https://irecommend.ru/content/vozmozhnost-uchi...">https://irecommend.ru/content/vozmozhnost-uchi...</a>	1
8	Двоякое впечатление. Отличные преподаватели. М...	Двоякое впечатление. Мой ребёнок занимался в э...	1	<a href="https://irecommend.ru/content/dvoyakoe-vpechat...">https://irecommend.ru/content/dvoyakoe-vpechat...</a>	1
9	Издала 2 книги при помощи Ridero, расскажу о п...	Здравствуйте! В последнее время я не так часто...	1	<a href="https://irecommend.ru/content/izdala-2-knigi-p...">https://irecommend.ru/content/izdala-2-knigi-p...</a>	1
10	Сдача экзамена потрясла все мыслимое и немыслимое	Мне очень нравился процесс обучения по курсу "...	0	<a href="https://irecommend.ru/content/sdacha-ekzamena-...">https://irecommend.ru/content/sdacha-ekzamena-...</a>	1
11	Как я прошла 5 бесплатных курсов за 5 месяцев...	Может ли бесплатное обучение быть качественным...	1	<a href="https://irecommend.ru/content/kak-ya-proshla-5...">https://irecommend.ru/content/kak-ya-proshla-5...</a>	1
12	Профессиональная завлекаловка	Когда пытаешься написать негативный отзыв об э...	0	<a href="https://irecommend.ru/content/professionalnaya...">https://irecommend.ru/content/professionalnaya...</a>	1
13	Stepik. Обучение на мобильном телефоне. Прошла...	Я очень люблю учиться, зарегистрирована на раз...	1	<a href="https://irecommend.ru/content/stepik-obuchenie...">https://irecommend.ru/content/stepik-obuchenie...</a>	1
14	Онлайн-школа Kodland	Сегодняшний отзыв будет об онлайн-школа Kodlan...	1	<a href="https://irecommend.ru/content/onlain-shkola-ko...">https://irecommend.ru/content/onlain-shkola-ko...</a>	1
15	Отзыв преподавателя на Юдеми, как продаются ку...	На сайт обучающей платформы Udemy я пошла посл...	1	<a href="https://irecommend.ru/content/otzyv-prepodavat...">https://irecommend.ru/content/otzyv-prepodavat...</a>	1

	header	text	verdict	link	predicted
16	5 пройденных курсов — ожидание и реальность	Чтобы не было мучительно больно за деградацию ...	1	<a href="https://irecommend.ru/content/5-proidennykh-kursov--o">https://irecommend.ru/content/5-proidennykh-kursov--o</a>	1
17	Как получить американский диплом, не выходя из...	Для начала отмечу, что - это , которая предлаг...	1	<a href="https://irecommend.ru/content/kak-poluchit-ame">https://irecommend.ru/content/kak-poluchit-ame</a>	1
18	Мнение психолога. Инфоцыгане ☺ Стоит ли тратить...	Современная действительность всё чаще, толкает...	0	<a href="https://irecommend.ru/content/mnenie-psikholog">https://irecommend.ru/content/mnenie-psikholog</a>	1
19	Code.org - сайт для обучения программированию	Искал я сайты для обучения программированию. И...	1	<a href="https://irecommend.ru/content/codeorg-sait-dly">https://irecommend.ru/content/codeorg-sait-dly</a>	1
20	Не советую школу по многим причинам	Не советую школу по многим причинам	0	<a href="https://irecommend.ru/content/ne-sovetuyu-shko">https://irecommend.ru/content/ne-sovetuyu-shko</a>	1
21	Мечтаете о дипломе Стенфорда, да ещё не выходя...	Всем доброго времени суток) Если вы, также как...	1	<a href="https://irecommend.ru/content/mechtaete-o-dipl">https://irecommend.ru/content/mechtaete-o-dipl</a>	1
22	Отличный сайт для самообразования в сфере комп...	Всем привет! Решила немного разбавить в свои о...	1	<a href="https://irecommend.ru/content/otlichnyi-sait-d">https://irecommend.ru/content/otlichnyi-sait-d</a>	1
23	Заграничное образование? Легко! Действительно ...	Подарили мне два месяца назад мой добрый муж на...	0	<a href="https://irecommend.ru/content/zagranichnoe-obr">https://irecommend.ru/content/zagranichnoe-obr</a>	1
24	Для тех, кто любит учиться ☺: искренне, от ду...	Со Степиком я познакомилась четыре года назад,...	1	<a href="https://irecommend.ru/content/dlya-tekh-kto-ly">https://irecommend.ru/content/dlya-tekh-kto-ly</a>	1
25	Нечем заняться во время карантина? Пройдите БЕ...	Доброго времени суток, дорогие читатели! Сегодня...	1	<a href="https://irecommend.ru/content/nechem-zanyatsya">https://irecommend.ru/content/nechem-zanyatsya</a>	1
26	Неплохой образовательный ресурс, выдаются серт...	Открыла для себя Stepik в апреле этого года. П...	1	<a href="https://irecommend.ru/content/neplokhoi-obrazo">https://irecommend.ru/content/neplokhoi-obrazo</a>	1
27	После прохождения бесплатного курса заработала...	Сборник онлайн платных и бесплатных курсов. Я ...	1	<a href="https://irecommend.ru/content/posle-prokhozhde">https://irecommend.ru/content/posle-prokhozhde</a>	1
28	Учим итальянский язык по песням: мой опыт рабо...	Сегодня хочу поделиться своим опытом работы на...	1	<a href="https://irecommend.ru/content/uchim-italyanski">https://irecommend.ru/content/uchim-italyanski</a>	1
29	Невозможное возможно - я разместила курс на St...	От момента как я узнала о платформе Stepik до ...	1	<a href="https://irecommend.ru/content/nevozmozhnoe-voz">https://irecommend.ru/content/nevozmozhnoe-voz</a>	1
30	Хотите учиться бесплатно? Тогда вам сюда. Огро...	Всем привет. Я все продолжаю вас знакомить с о...	1	<a href="https://irecommend.ru/content/khotite-uchitsya">https://irecommend.ru/content/khotite-uchitsya</a>	1

	header	text verdict	link	predicted
31	БЕСПЛАТНЫЕ обучающие курсы на любой вкус, в лю...	Всем привет! Я люблю учиться. Вот честно, очен...	1	<a href="https://irecommend.ru/content/besplatnye-obuch...">https://irecommend.ru/content/besplatnye-obuch...</a>

Вывод метрик тоже внушает оптимизма, по f1 почти 90% эффективность

```
In [ ]: # Подсчет метрик
accuracy = accuracy_score(st_dataset['verdict'], st_dataset['predicted'])
precision = precision_score(st_dataset['verdict'], st_dataset['predicted'])
recall = recall_score(st_dataset['verdict'], st_dataset['predicted'])
f1 = f1_score(st_dataset['verdict'], st_dataset['predicted'])
print(f"accuracy = {accuracy}, precision = {precision}, recall = {recall}, f1 = {f1}")

accuracy = 0.8125, precision = 0.8125, recall = 1.0, f1 = 0.896551724137931
```

Оценим датасет GeekBrains

```
In [ ]: # Применение функции к столбцу текста в датасете
gb_dataset['predicted'] = gb_dataset['text'].apply(predict_sentiment)
```

Метрики, также хорошие

```
In [ ]: accuracy = accuracy_score(gb_dataset['verdict'], gb_dataset['predicted'])
precision = precision_score(gb_dataset['verdict'], gb_dataset['predicted'])
recall = recall_score(gb_dataset['verdict'], gb_dataset['predicted'])
f1 = f1_score(gb_dataset['verdict'], gb_dataset['predicted'])
print(f"accuracy = {accuracy}, precision = {precision}, recall = {recall}, f1 = {f1}")

accuracy = 0.7325581395348837, precision = 0.7325581395348837, recall = 1.0, f1 = 0.8456375838926175
```

Датасет SkillBox

```
In [ ]: # Применение функции к столбцу текста в датасете
sb_dataset['predicted'] = sb_dataset['text'].apply(predict_sentiment)
```

F1 более 90%

```
In [ ]: accuracy = accuracy_score(sb_dataset['verdict'], sb_dataset['predicted'])
precision = precision_score(sb_dataset['verdict'], sb_dataset['predicted'])
recall = recall_score(sb_dataset['verdict'], sb_dataset['predicted'])
f1 = f1_score(sb_dataset['verdict'], sb_dataset['predicted'])
print(f"accuracy = {accuracy}, precision = {precision}, recall = {recall}, f1 = {f1}")

accuracy = 0.896551724137931, precision = 0.896551724137931, recall = 1.0, f1 = 0.9030701754492965
```

```
accuracy = 0.8294573643410853, precision = 0.8294573643410853, recall = 1.0, f1 = 0.9067796610169491
```

А вот и подвох. Longformer, тупо ставит всем отзывам "положительно". Ранее на выводе короткого датасета Stepik это наиболее заметно (

```
In [ ]: sb_dataset['predicted'].value_counts()
```

```
Out[ ]: predicted  
1    516  
Name: count, dtype: int64
```

У ГБ тоже одни единицы

```
In [ ]: gb_dataset['predicted'].value_counts()
```

```
Out[ ]: predicted  
1    516  
Name: count, dtype: int64
```

Датасет SkillFactory

```
In [ ]: # Применение функции к столбцу текста в датасете  
sf_dataset['predicted'] = sf_dataset['text'].apply(predict_sentiment)
```

Вроде и метрики, больше походят на реальные, F1 61%.

```
In [ ]: accuracy = accuracy_score(sf_dataset['verdict'], sf_dataset['predicted'])  
precision = precision_score(sf_dataset['verdict'], sf_dataset['predicted'])  
recall = recall_score(sf_dataset['verdict'], sf_dataset['predicted'])  
f1 = f1_score(sf_dataset['verdict'], sf_dataset['predicted'])  
print(f"accuracy = {accuracy}, precision = {precision}, recall = {recall}, f1 = {f1}")
```

```
accuracy = 0.44285714285714284, precision = 0.44285714285714284, recall = 1.0, f1 = 0.6138613861386139
```

Но, опять столбец predicted содержит только единицы.

```
In [ ]: sf_dataset['predicted'].value_counts()
```

```
Out[ ]: predicted  
1    70  
Name: count, dtype: int64
```

DeepPavlov RuBert

Попробуем рекомендованную модель RuBert, может быть проблема у LongFormer была в многоязычности.

```
In [ ]: # Загрузка предварительно обученной модели и токенизатора RuBERT
model_name = 'DeepPavlov/rubert-base-cased'
model = BertForSequenceClassification.from_pretrained(model_name)
tokenizer = BertTokenizer.from_pretrained(model_name)

def pred_sentiment(text):
    inputs = tokenizer(text, return_tensors='pt', truncation=True, padding=True)
    outputs = model(**inputs)
    predictions = torch.softmax(outputs.logits, dim=1)
    sentiment_label = 1 if predictions[0][1] > predictions[0][0] else 0 # 1 для положительного, 0 для отрицательного
    return sentiment_label

config.json: 0% | 0.00/642 [00:00<?, ?B/s]
pytorch_model.bin: 0% | 0.00/714M [00:00<?, ?B/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at DeepPavlov/rubert-base-cased and
are newly initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
tokenizer_config.json: 0% | 0.00/24.0 [00:00<?, ?B/s]
vocab.txt: 0% | 0.00/1.65M [00:00<?, ?B/s]
special_tokens_map.json: 0% | 0.00/112 [00:00<?, ?B/s]
```

Т.к. по описанным ранее причинам (до 512 токенов), текстовое поле (text) не может быть обработано Бертом, будем оценивать заголовок, который зачастую содержит по несколько предложений.

Оцениваем заголовки датасета Нетологии

```
In [ ]: nt_dataset['pred'] = nt_dataset['header'].apply(pred_sentiment)
```

Asking to truncate to max\_length but no maximum length is provided and the model has no predefined maximum length. Default to no truncation.

Всего лишь два нуля, т.е. большинство (238) отзыва, получили опять единицы (

```
In [ ]: nt_dataset['pred'].value_counts()
```

```
Out[ ]: pred
1    238
0     2
Name: count, dtype: int64
```

И метрики хорошие.

```
In [ ]: accuracy = accuracy_score(nt_dataset['verdict'], nt_dataset['pred'])
precision = precision_score(nt_dataset['verdict'], nt_dataset['pred'])
recall = recall_score(nt_dataset['verdict'], nt_dataset['pred'])
f1 = f1_score(nt_dataset['verdict'], nt_dataset['pred'])
print(f"accuracy = {accuracy}, precision = {precision}, recall = {recall}, f1 = {f1}")
```

```
accuracy = 0.7291666666666666, precision = 0.7352941176470589, recall = 0.9887005649717514, f1 = 0.8433734939759037
```

Вывод датасета Нетологии с двумя предсказанными полями predicted (оценка text) и pred (оценка коротких header). Ничего общего с лейблом verdict (

```
In [ ]: nt_dataset.loc[nt_dataset['verdict'] != nt_dataset['pred']]
```

Out[ ]:

	header	text	verdict	link	predicted	pred
1	диGEEKBRAINSзайнер (ЧУДА не будет)	Долго собиралась опубликовать данный отзыв и у...	0	<a href="https://irecommend.ru/content/digeekbrainszain...">https://irecommend.ru/content/digeekbrainszain...</a>	1	1
4	Отвратительно	Месяц назад приобрела курс продакт менеджер на...	0	<a href="https://irecommend.ru/content/otvratitelno-5408">https://irecommend.ru/content/otvratitelno-5408</a>	1	1
5	Разочаровали	Поступил на курс по системной аналитике в конц...	0	<a href="https://irecommend.ru/content/razocharovali-150">https://irecommend.ru/content/razocharovali-150</a>	1	1
6	не рекомендую!	Всем добрый день, училась на курсе иллюстрация...	0	<a href="https://irecommend.ru/content/ne-rekomenduyu-2964">https://irecommend.ru/content/ne-rekomenduyu-2964</a>	1	1
7	Совершенно непригодная контора	Оплатил обучение на курсе "1С-программист: с н...	0	<a href="https://irecommend.ru/content/sovershenno-nepr...">https://irecommend.ru/content/sovershenno-nepr...</a>	1	1
...	...	...	...	...	...	...
206	GeekBrains - много пиара и вранье по трудуоstr...	Предполагаю, что данный отзыв получится больши...	0	<a href="https://irecommend.ru/content/geekbrains-mnogo...">https://irecommend.ru/content/geekbrains-mnogo...</a>	1	1
218	Никому не рекомендую! Я вернула полную стоимос...	Качество курсов низкое В материалах курса мног...	0	<a href="https://irecommend.ru/content/nikomu-ne-rekome...">https://irecommend.ru/content/nikomu-ne-rekome...</a>	1	1
221	Нечестная школа: Не возвращают деньги за обуче...	Вернули только 40% стоимости курса, хотя ни од...	0	<a href="https://irecommend.ru/content/ne-vozvrashchayu...">https://irecommend.ru/content/ne-vozvrashchayu...</a>	1	1
230	Не стоит своих денег	Училась на курсе UI UX дизайна. Курс купила за...	0	<a href="https://irecommend.ru/content/ne-stoit-svoikh-...">https://irecommend.ru/content/ne-stoit-svoikh-...</a>	1	1
231	Никогда не покупайте обучение в нетологии, побе...	Я являюсь действующим менеджером по вайлдбериз...	0	<a href="https://irecommend.ru/content/nikogda-ne-pokup...">https://irecommend.ru/content/nikogda-ne-pokup...</a>	1	1

65 rows x 6 columns

Может быть 2 нуля потому, что датасет не очень большой

Оценим заголовки самого большого датасета GeekBrain

```
In [ ]: gb_dataset['pred'] = gb_dataset['header'].apply(pred_sentiment)
```

И опять, всего два нуля (

```
In [ ]: gb_dataset['pred'].value_counts()
```

```
Out[ ]: pred
1    514
0     2
Name: count, dtype: int64
```

## Файнтюнинг

Собственно, возвращаемся к теме курса. Будем использовать не готовые модели обученные, на непонятных отзывах, может это были базы IMDB или оценка кулинарных рецептов, пробуем загрузить предобученную модель RuBert и дообучить ее на целевых датасетах.

Т.к. количество данных является весомым фактором, то объединю несколько датасетов (ГБ + Яндекс + Учи + Нетология + Скилбокс и Skylearn) в один тренировочный, размером 1776 - записей.

В качестве валидационных данных использую датасет SkillBox. Единственным не тронутым датасетом останется Stepik

```
In [12]: # Загрузка данных
train_df = pd.concat([gb_dataset[['header', 'verdict']], ya_dataset[['header', 'verdict']], uchi_dataset[['header', 'verdict']]]
train_df.columns = ['text', 'label']
train_df = train_df.dropna()
train_df = train_df.reset_index(drop=True)

eval_df = sf_dataset[['header', 'verdict']].copy()
eval_df.columns = ['text', 'label']
eval_df = eval_df.dropna()
eval_df = eval_df.reset_index(drop=True)

train_df.shape, eval_df.shape
```

```
Out[12]: ((1776, 2), (70, 2))
```

Выставляю 10 эпох обучения. Чтобы не было переобучения служит класс EarlyStoppingCallback.

Чтобы наблюдать процесс обучения после каждой эпохи предназначен класс LogMetricsCallback

In [13]:

```
class LogMetricsCallback(TrainerCallback):
    def __init__(self):
        self.last_step = 0

    def on_evaluate(self, args, state, control, model, eval_dataloader=None, **kwargs):
        if state.global_step > self.last_step:
            metrics = state.log_history[-1] # Получение метрик из последней записи в истории логов
            print(metrics) # Вывод метрик
            self.last_step = state.global_step

class EarlyStoppingCallback(TrainerCallback):
    def __init__(self, patience=1):
        self.patience = patience
        self.wait = 0
        self.best_accuracy = float('-inf')
        self.best_precision = float('-inf')

    def on_evaluate(self, args, state, control, model, eval_dataloader=None, **kwargs):
        metrics = state.log_history[-1] # Получение метрик из последней записи в истории логов
        accuracy = metrics.get("accuracy", 0.0)
        precision = metrics.get("precision", 0.0)

        if accuracy > self.best_accuracy and precision > self.best_precision:
            self.best_accuracy = accuracy
            self.best_precision = precision
            self.wait = 0
        else:
            self.wait += 1
            if self.wait >= self.patience:
                control.should_training_stop = True

    # Определение модели и токенизатора
model_name = "DeepPavlov/rubert-base-cased"
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=2)
tokenizer = BertTokenizer.from_pretrained(model_name)

    # Подготовка данных для обучения
class CustomDataset(torch.utils.data.Dataset):
    def __init__(self, data, tokenizer, max_length):
        self.data = data
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __getitem__(self, idx):
        text = self.data.loc[idx, "text"]
        label = self.data.loc[idx, "label"]
```

```
encoding = self.tokenizer(text, truncation=True, padding='max_length', max_length=self.max_length, return_tensors='pt')
    return {
        'input_ids': encoding['input_ids'].flatten(),
        'attention_mask': encoding['attention_mask'].flatten(),
        'labels': torch.tensor(label, dtype=torch.long)
    }

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

train_dataset = CustomDataset(train_df, tokenizer, max_length=512)
eval_dataset = CustomDataset(eval_df, tokenizer, max_length=512)

# Параметры обучения
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=10,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    logging_dir='./logs',
    do_train=True,
    do_eval=True,
    logging_steps=100,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True
)

# Создание Trainer с добавлением колбэков ранней остановки и логирования метрик
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    callbacks=[LogMetricsCallback(), EarlyStoppingCallback()]
)

# Обучение модели
trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:  
The secret `HF_TOKEN` does not exist in your Colab secrets.  
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it  
as secret in your Google Colab and restart your session.  
You will be able to reuse this secret in all of your notebooks.  
Please note that authentication is recommended but still optional to access public models or datasets.  
    warnings.warn(  
config.json:  0%|          | 0.00/642 [00:00<?, ?B/s]  
pytorch_model.bin: 0%|          | 0.00/714M [00:00<?, ?B/s]  
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at DeepPavlov/rubert-base-cased and  
are newly initialized: ['classifier.bias', 'classifier.weight']  
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.  
tokenizer_config.json:  0%|          | 0.00/24.0 [00:00<?, ?B/s]  
vocab.txt:  0%|          | 0.00/1.65M [00:00<?, ?B/s]  
special_tokens_map.json:  0%|          | 0.00/112 [00:00<?, ?B/s]  
/usr/local/lib/python3.10/dist-packages/accelerate/accelerator.py:436: FutureWarning: Passing the following arguments to `Accelerator` is deprecated and will be removed in version 1.0 of Accelerate: dict_keys(['dispatch_batches', 'split_batches', 'even_batches', 'use_seedable_sampler']). Please pass an `accelerate.DataLoaderConfiguration` instead:  
dataloader_config = DataLoaderConfiguration(dispatch_batches=None, split_batches=False, even_batches=True, use_seedable_sampler=True)  
    warnings.warn(  
[ 444/2220 05:54 < 23:45, 1.25 it/s, Epoch 2/10]
```

Epoch	Training Loss	Validation Loss
1	0.400300	0.528434
2	0.319700	0.392309

```
{'eval_loss': 0.5284343957901001, 'eval_runtime': 2.1808, 'eval_samples_per_second': 32.098, 'eval_steps_per_second': 4.127, 'epoch': 1.0, 'step': 222}  
{'eval_loss': 0.39230871200561523, 'eval_runtime': 2.1559, 'eval_samples_per_second': 32.47, 'eval_steps_per_second': 4.175, 'epoch': 2.0, 'step': 444}
```

```
Out[13]: TrainOutput(global_step=444, training_loss=0.3846369992505323, metrics={'train_runtime': 358.316, 'train_samples_per_second': 49.565, 'train_steps_per_second': 6.196, 'total_flos': 934570468638720.0, 'train_loss': 0.3846369992505323, 'epoch': 2.0})
```

Как оказалось, достаточно всего двух эпох, далее модель начинает переобучаться, возможно слишком мало тренировочных данных, возможно нужно поиграться с гиперпараметрами. В любом случае проверим результат оценки, но сначала на коротких заголовках header

```
In [14]: # Предположим, что nt_dataset это ваш новый датасет с текстами, которые вы хотите классифицировать  
# Токенизация текстов из столбца "header"  
tokenized_texts = tokenizer(nt_dataset['header'].tolist(), truncation=True, padding=True)  
# Получение предсказаний  
input_ids = torch.tensor(tokenized_texts['input_ids'])
```

```

attention_mask = torch.tensor(tokenized_texts['attention_mask'])
# Предсказание с использованием модели
with torch.no_grad():
    # Перемещаем тензоры на устройство GPU, если они находятся на CPU
    input_ids = input_ids.to(model.device)
    attention_mask = attention_mask.to(model.device)
    outputs = model(input_ids=input_ids, attention_mask=attention_mask)
# Преобразование выходов модели в метки классов
predictions = torch.argmax(outputs.logits, dim=1).tolist()
# Добавление предсказаний в новый столбец 'pred' датасета
nt_dataset['pred'] = predictions
# Фильтрация строк, в которых значения в столбцах 'verdict' и 'pred' отличаются
nt_dataset.loc[nt_dataset['verdict'] != nt_dataset['pred'], ['verdict', '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.

Out[14]:

	verdict	pred
13	1	0
31	1	0
59	1	0
74	0	1
77	0	1
80	0	1
84	0	1
90	0	1

Вот это уже лучше. Результат впечатляет. Наверху вывод содержит только строки отличные от эталонных. Есть и нули и единицы. И количество не большое, даже не нужны метрики  $8/70 = 11\%$  ошибок

Попробуем предсказать заголовки на датасете Stepik, напоминаю он не использовался в обучении и валидации:

```

In [15]: tokenized_texts = tokenizer(st_dataset['header'].tolist(), truncation=True, padding=True)
# Получение предсказаний
input_ids = torch.tensor(tokenized_texts['input_ids'])
attention_mask = torch.tensor(tokenized_texts['attention_mask'])
# Предсказание с использованием модели
with torch.no_grad():
    # Перемещаем тензоры на устройство GPU, если они находятся на CPU

```

```

input_ids = input_ids.to(model.device)
attention_mask = attention_mask.to(model.device)
outputs = model(input_ids=input_ids, attention_mask=attention_mask)
# Преобразование выходов модели в метки классов
predictions = torch.argmax(outputs.logits, dim=1).tolist()
st_dataset['pred'] = predictions
# Фильтрация строк, в которых значения в столбцах 'verdict' и 'pred' отличаются
st_dataset.loc[st_dataset['verdict'] != st_dataset['pred'], ['verdict', 'pred']]

```

Out[15]:

	verdict	pred
<b>18</b>	0	1
<b>23</b>	0	1
<b>27</b>	1	0
<b>29</b>	1	0

Результат аналогичный, т.е. вне зависимости от набора данных имеем примерно 87% точности.

Ну и сравним с валидационным набором SkillFactory. Он в самой тренировке нейросети участия не принимал, он лишь служил для валидацией.

In [16]:

```

tokenized_texts = tokenizer(sf_dataset['header'].tolist(), truncation=True, padding=True)
# Получение предсказаний
input_ids = torch.tensor(tokenized_texts['input_ids'])
attention_mask = torch.tensor(tokenized_texts['attention_mask'])
# Предсказание с использованием модели
with torch.no_grad():
    # Перемещаем тензоры на устройство GPU, если они находятся на CPU
    input_ids = input_ids.to(model.device)
    attention_mask = attention_mask.to(model.device)
    outputs = model(input_ids=input_ids, attention_mask=attention_mask)
# Преобразование выходов модели в метки классов
predictions = torch.argmax(outputs.logits, dim=1).tolist()
sf_dataset['pred'] = predictions
# Фильтрация строк, в которых значения в столбцах 'verdict' и 'pred' отличаются
sf_dataset.loc[sf_dataset['verdict'] != sf_dataset['pred'], ['verdict', 'pred']]

```

Out[16]:

	verdict	pred
3	1	0
24	1	0
36	0	1
37	0	1
53	0	1
58	0	1
67	0	1

7 ошибок на 70 записей - 90% процентная точность

Но, что будет если оценивать не заголовки, а сам текст, посмотрим далее

## Оценка поля text

Т.к. текст огромный, то применю метод обрезки. Сначала делал это с помощью функции, проверяющей каждую строку поля text, позже обнаружил, что это можно делать с помощью параметра max\_length объекта tokenizer.

```
In [19]: #st_dataset['text'] = st_dataset['text'].apply(lambda x: ' '.join(word_tokenize(x)[:512]))
tokenized_texts = tokenizer(st_dataset['text'].tolist(), truncation=True, padding=True, max_length=512)
# Получение предсказаний
input_ids = torch.tensor(tokenized_texts['input_ids'])
attention_mask = torch.tensor(tokenized_texts['attention_mask'])
# Предсказание с использованием модели
with torch.no_grad():
    # Перемещаем тензоры на устройство GPU, если они находятся на CPU
    input_ids = input_ids.to(model.device)
    attention_mask = attention_mask.to(model.device)
    outputs = model(input_ids=input_ids, attention_mask=attention_mask)
# Преобразование выходов модели в метки классов
predictions = torch.argmax(outputs.logits, dim=1).tolist()
st_dataset['predict'] = predictions
# Фильтрация строк, в которых значения в столбцах 'verdict' и 'pred' отличаются
st_dataset.loc[st_dataset['verdict'] != st_dataset['predict'], ['verdict', 'predict']]
```

Out[19]:

	verdict	predict
5	0	1
12	0	1
18	0	1
23	0	1

На самом маленьком датасете Stepik результат оценки поля text имеет такую же точность, как и на поле header

```
In [20]: tokenized_texts = tokenizer(sf_dataset['text'].tolist(), truncation=True, padding=True, max_length=512)
# Получение предсказаний
input_ids = torch.tensor(tokenized_texts['input_ids'])
attention_mask = torch.tensor(tokenized_texts['attention_mask'])
# Предсказание с использованием модели
with torch.no_grad():
    # Перемещаем тензоры на устройство GPU, если они находятся на CPU
    input_ids = input_ids.to(model.device)
    attention_mask = attention_mask.to(model.device)
    outputs = model(input_ids=input_ids, attention_mask=attention_mask)
# Преобразование выходов модели в метки классов
predictions = torch.argmax(outputs.logits, dim=1).tolist()
sf_dataset['predict'] = predictions
# Фильтрация строк, в которых значения в столбцах 'verdict' и 'pred' отличаются
sf_dataset.loc[sf_dataset['verdict'] != sf_dataset['predict'], ['verdict', 'predict']]
```

Out[20]:

	verdict	predict
<b>0</b>	0	1
<b>2</b>	0	1
<b>3</b>	1	0
<b>7</b>	0	1
<b>8</b>	1	0
<b>11</b>	0	1
<b>13</b>	0	1
<b>15</b>	0	1
<b>16</b>	0	1
<b>17</b>	0	1
<b>20</b>	0	1
<b>21</b>	0	1
<b>22</b>	0	1
<b>28</b>	0	1
<b>34</b>	0	1
<b>36</b>	0	1
<b>37</b>	0	1
<b>47</b>	0	1
<b>52</b>	0	1
<b>53</b>	0	1
<b>54</b>	0	1
<b>55</b>	0	1
<b>58</b>	0	1
<b>60</b>	0	1
<b>61</b>	0	1
<b>67</b>	0	1

С датасетом SkillFactory картина не такая радужная (точность 63%), но все же нейросеть работает, хоть и не показывает выдающихся результатов

```
In [40]: # sf_dataset['text'] = sf_dataset['text'].apply(lambda x: ' '.join(word_tokenize(x)[:512]))
tokenized_texts = tokenizer(nt_dataset['text'].tolist(), truncation=True, padding=True, max_length=512)
# Получение предсказаний
input_ids = torch.tensor(tokenized_texts['input_ids'])
attention_mask = torch.tensor(tokenized_texts['attention_mask'])
# Предсказание с использованием модели
with torch.no_grad():
    # Перемещаем тензоры на устройство GPU, если они находятся на CPU
    input_ids = input_ids.to(model.device)
    attention_mask = attention_mask.to(model.device)
    outputs = model(input_ids=input_ids, attention_mask=attention_mask)
# Преобразование выходов модели в метки классов
predictions = torch.argmax(outputs.logits, dim=1).tolist()
nt_dataset['predict'] = predictions
# Фильтрация строк, в которых значения в столбцах 'verdict' и 'pred' отличаются
nt_dataset.loc[nt_dataset['verdict'] != nt_dataset['predict'], ['verdict', 'predict']]
```

	verdict	predict
<b>1</b>	0	1
<b>5</b>	0	1
<b>10</b>	0	1
<b>27</b>	0	1
<b>47</b>	0	1
<b>48</b>	0	1
<b>54</b>	0	1
<b>55</b>	0	1
<b>58</b>	0	1
<b>61</b>	0	1
<b>67</b>	0	1
<b>72</b>	0	1
<b>73</b>	0	1
<b>74</b>	0	1
<b>76</b>	0	1
<b>77</b>	0	1
<b>78</b>	0	1
<b>80</b>	0	1
<b>81</b>	0	1
<b>84</b>	0	1
<b>86</b>	0	1
<b>87</b>	0	1
<b>90</b>	0	1
<b>91</b>	0	1
<b>93</b>	0	1
<b>95</b>	0	1
<b>156</b>	0	1

	<b>verdict</b>	<b>predict</b>
<b>186</b>	0	1
<b>188</b>	0	1
<b>190</b>	0	1
<b>191</b>	0	1
<b>192</b>	0	1
<b>194</b>	0	1
<b>197</b>	0	1
<b>198</b>	0	1
<b>206</b>	0	1
<b>231</b>	0	1

37 ошибок из 240, итого точность предсказания 85%. Не плохо, для небольшого дообучающего датасета.

### Сравнение verdict с pred и predict

Посмотрим на разницу в оценках по заголовку и по тексту. Выведем только отличающиеся от эталона строки.

Датасет Stepik

```
In [35]: mask = (st_dataset['verdict'] != st_dataset['pred']) | (st_dataset['verdict'] != st_dataset['predict']) | (st_dataset['pred'] != st_dataset['predict'])
filtered_dataset = st_dataset[mask]
filtered_dataset[['verdict', 'pred', 'predict']]
```

Out[35]:

	<b>verdict</b>	<b>pred</b>	<b>predict</b>
<b>5</b>	0	0	1
<b>12</b>	0	0	1
<b>18</b>	0	1	1
<b>23</b>	0	1	1
<b>27</b>	1	0	1
<b>29</b>	1	0	1

Как видно из вывода выше. Количество ошибок в predict (оценка text) одинаковое с pred (оценка header), скорее всего это связано с тем, что заголовок должен звучать и отражать всю суть происходящего.

В тексте же больше повествования, которое нейросетью воспринимается как положительное, потому что в повествовании меньше негатива. Возможно, весь негатив пытаются поместить в заключение. В любом случае будь то хороший отзыв или плохой, если его надо расписать на несколько предложений или абзацев, то в начале мы будем писать историю того, как мы пришли к конкретной ситуации.

### Датасет SkillFactory

Ситуация похожая на предыдущий датасет, но встречаются и нули. Это скорее всего, либо короткие отзывы, либо резко негативные отзывы.

```
In [36]: mask2 = (sf_dataset['verdict'] != sf_dataset['pred']) | (sf_dataset['verdict'] != sf_dataset['predict']) | (sf_dataset['pred']  
filtered2_dataset = sf_dataset[mask2]  
filtered2_dataset[['verdict', 'pred', 'predict']]
```

Out[36]:

	verdict	pred	predict
<b>0</b>	0	0	1
<b>2</b>	0	0	1
<b>3</b>	1	0	0
<b>7</b>	0	0	1
<b>8</b>	1	1	0
<b>11</b>	0	0	1
<b>13</b>	0	0	1
<b>15</b>	0	0	1
<b>16</b>	0	0	1
<b>17</b>	0	0	1
<b>20</b>	0	0	1
<b>21</b>	0	0	1
<b>22</b>	0	0	1
<b>24</b>	1	0	1
<b>28</b>	0	0	1
<b>34</b>	0	0	1
<b>36</b>	0	1	1
<b>37</b>	0	1	1
<b>47</b>	0	0	1
<b>52</b>	0	0	1
<b>53</b>	0	1	1
<b>54</b>	0	0	1
<b>55</b>	0	0	1
<b>58</b>	0	1	1
<b>60</b>	0	0	1
<b>61</b>	0	0	1
<b>67</b>	0	1	1

Датасет Netology, в данном примере наблюдается минимальные ошибки в оценке заголовка и максимальные с оценкой текста. Связываю это с тем, что датасет использовался для тренировки модели, поэтому у него больше отгаданных заголовков, нежели текстового поля.

```
In [41]: mask3 = (nt_dataset['verdict'] != nt_dataset['pred']) | (nt_dataset['verdict'] != nt_dataset['predict']) | (nt_dataset['pred']  
filtered3_dataset = nt_dataset[mask3]  
filtered3_dataset[['verdict', 'pred', 'predict']]
```

Out[41]:

	verdict	pred	predict
<b>1</b>	0	0	1
<b>5</b>	0	0	1
<b>10</b>	0	0	1
<b>13</b>	1	0	1
<b>27</b>	0	0	1
<b>31</b>	1	0	1
<b>47</b>	0	0	1
<b>48</b>	0	0	1
<b>54</b>	0	0	1
<b>55</b>	0	0	1
<b>58</b>	0	0	1
<b>59</b>	1	0	1
<b>61</b>	0	0	1
<b>67</b>	0	0	1
<b>72</b>	0	0	1
<b>73</b>	0	0	1
<b>74</b>	0	1	1
<b>76</b>	0	0	1
<b>77</b>	0	1	1
<b>78</b>	0	0	1
<b>80</b>	0	1	1
<b>81</b>	0	0	1
<b>84</b>	0	1	1
<b>86</b>	0	0	1
<b>87</b>	0	0	1
<b>90</b>	0	1	1
<b>91</b>	0	0	1

<b>verdict</b>	<b>pred</b>	<b>predict</b>
<b>93</b>	0	0
<b>95</b>	0	0
<b>156</b>	0	0
<b>186</b>	0	0
<b>188</b>	0	0
<b>190</b>	0	0
<b>191</b>	0	0
<b>192</b>	0	0
<b>194</b>	0	0
<b>197</b>	0	0
<b>198</b>	0	0
<b>206</b>	0	0
<b>231</b>	0	0

## Заключение

Работа показала, что доучивать модель стоит под конкретный случай. Также для меня ясно, что для больших текстов нужно использовать более современные языковые модели.

Самым главным выводом для себя сделал тот факт, что на доучивание модели до приемлемого результата не нужны огромные датасеты, и большое количество эпох.

2 эпохи обучения в нашем случае не идут ни в какое сравнение с десятками и даже сотнями эпох на полном обучении.

Во введении приводил ссылку на работу по оценке эмоциональной окрашенности с помощью машинного обучения, там использовался датасет на 50 тысяч отзывов.