Библиотеки

```
!pip install datasets
```

```
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-packages (from datasets) (4.67.1)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.12.0,>=2023.1.0 (from fsspec[http]<=2024.12.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.12.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets) (3.11.14)
Requirement already satisfied: huggingface-hub>=0.24.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (0.29.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.6.1
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.2.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.18.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.24.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datase
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets) (2
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->datasets
Downloading datasets-3.5.0-py3-none-any.whl (491 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.2/491.2 kB 11.0 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 9.4 MB/s eta 0:00:00
Downloading fsspec-2024.12.0-py3-none-any.whl (183 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 183.9/183.9 kB 11.7 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 9.2 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 5.6 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2025.3.0
    Uninstalling fsspec-2025.3.0:
      Successfully uninstalled fsspec-2025.3.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the
torch 2.6.0+cu124 requires nvidia-cublas-cu12==12.4.5.8; platform_system == "Linux" and platform_machine == "x86_64", but you hav
torch 2.6.0+cu124 requires nvidia-cuda-cupti-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you
torch 2.6.0+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you
torch 2.6.0+cu124 requires nvidia-cuda-runtime-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but y
torch 2.6.0+cu124 requires nvidia-cudnn-cu12==9.1.0.70; platform_system == "Linux" and platform_machine == "x86_64", but you have
torch 2.6.0+cu124 requires nvidia-cufft-cu12==11.2.1.3; platform_system == "Linux" and platform_machine == "x86_64", but you have
torch 2.6.0+cu124 requires nvidia-curand-cu12==10.3.5.147; platform_system == "Linux" and platform_machine == "x86_64", but you h
torch 2.6.0+cu124 requires nvidia-cusolver-cu12==11.6.1.9; platform_system == "Linux" and platform_machine == "x86_64", but you h
torch 2.6.0+cu124 requires nvidia-cusparse-cu12==12.3.1.170; platform_system == "Linux" and platform_machine == "x86_64", but you
torch 2.6.0+cu124 requires nvidia-nvjitlink-cu12==12.4.127; platform_system == "Linux" and platform_machine == "x86_64", but you
```

```
!pip install matplotlib scikit-learn transformers
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.50.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.18.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.26.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.29.3)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
```

```
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.21.1)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.26.0->trans
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.2
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2025.1.3
```

```
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

import pandas as pd
import re
import numpy as np
import matplotlib.pyplot as plt
import random

from tqdm.auto import tqdm
from datasets import load_dataset, Dataset
from sklearn.model_selection import train_test_split

from transformers import BertTokenizer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from transformers import BertModel
from transformers.modeling_outputs import SequenceClassifierOutput
import copy
from tqdm.auto import tqdm
from datasets import DatasetDict
from transformers import TrainingArguments, Trainer, TrainerCallback
from transformers import DataCollatorWithPadding
from transformers import EarlyStoppingCallback
```

```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
```

```
'cuda'
```

```
RANDOM_STATE = 42
random.seed(RANDOM_STATE)
np.random.seed(RANDOM_STATE)
torch.manual_seed(RANDOM_STATE)
torch.cuda.manual_seed_all(RANDOM_STATE)
```

Работа с train датасетом

```
from google.colab import drive
import shutil
```

```
drive.mount('/content/drive/', force_remount=True)

dataset = load_dataset(
    'csv',
    data_files=f'/content/train.csv',
    column_names=['id','keyword','location', 'text', 'labels'],
)
```

```
Mounted at /content/drive/
Generating train split: 7614/0 [00:00<00:00, 58233.30 examples/s]
```

```
dataset
```

```
DatasetDict({
    train: Dataset({
        features: ['id', 'keyword', 'location', 'text', 'labels'],
        num_rows: 7614
    })
})
```

Наш датасет очень, очень захламлён. Сам автор написал, что могут быть пустые поля в keyword и location. Давайте объективно глянем на датасет. Локаций тут тьма, кто-то пишет млечный путь, кто-то ничего, в общем это будет мешать модели установить закономерности. Также обратим внимание на ссылки, нам не важны сайты, но они начинаются с http:// и их нужно почистить, они не несут полезной информации. Также мусором можно назвать обращения,хэштеги, инородные символы. Хэштег не скажет точно ли катастрофа или нет. Нам с этим очень помогут регулярные выражения. Тут возникла проблема. Я вместе с ненеужными тегами удаляю такие теги, как #RIP и т.п. Ну всё же иногда теги занимают чуть ли не половину всего текста. Мы можем с одной стороны дать модели хэштеги и позволить ей самой определять нужные и ненужные, но датасет очень маленький (было бы тысяч 20 хотя бы, а тут 6 тысяч без дубликатов). Мы пойдём на компромисс и превратим хэштеги #god в слова -> god

```
def clean_tweet(text):
    text = re.sub(r'http\S+', '', text)  # Удаление ссылок
    text = re.sub(r'@\w+', '', text)      # Удаление упоминаний
    text = re.sub(r'#', '', text)
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^A-Za-z0-9\s.,:;?!\-\']', '', text) #удаление всего not-English алфавита
    text = text.strip()                    # Удаление пробелов в начале и конце
    return text
```

```
train_pd = pd.DataFrame(dataset['train'])
train_pd = train_pd.drop('id', axis=1)
train_pd = train_pd.drop('location', axis=1)
train_pd = train_pd.groupby('text', as_index=False).first()
train_pd['text'] = list(map(clean_tweet, train_pd['text']))
train_pd = train_pd.explode('text')
train_pd = train_pd.groupby('text', as_index=False).first()
train_pd = train_pd.drop_duplicates().reset_index(drop=True)
train_pd['labels'] = pd.to_numeric(train_pd['labels'], errors='coerce')
train_pd = train_pd[train_pd['labels'].isin([0, 1])].reset_index(drop=True)
train_pd['labels'] = train_pd['labels'].astype(int)
train_pd
```

	text	keyword	labels
0	! Residents Return To Destroyed Homes As Washi...	wildfire	1
1	' no pharrell only YOU can prevent forest fire...	forest%20fires	0
2	'...As of right now I'm reopening the X-Files....	fear	0
3	'13 M. Chapoutier Crozes Hermitage so much pur...	crushed	0
4	'54 -9 How do people not know who Kendall Jenn...	screaming	0
...	...	...	...
6906	you're the snowstorm I'm purified. the darkest...	snowstorm	0
6907	you're too busy finishing those weapon designs	weapon	0
6908	your Tweet was quoted by	quarantine	0
6909	your lifetime odds of dying from an airplane a...	airplane%20accident	0
6910	your turn ??	electrocute	0

6911 rows × 3 columns

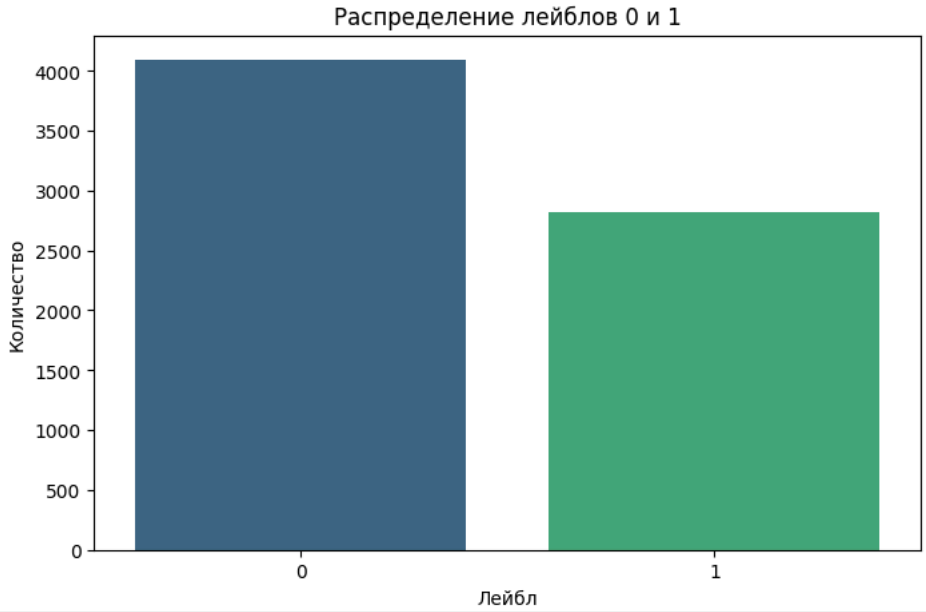
```
import matplotlib.pyplot as plt
import seaborn as sns
label_counts = train_pd['labels'].value_counts()

plt.figure(figsize=(8, 5))
sns.barplot(x=label_counts.index, y=label_counts.values, palette='viridis')
plt.title('Распределение лейблов 0 и 1')
plt.xlabel('Лейбл')
plt.ylabel('Количество')
plt.xticks(ticks=[0, 1], labels=['0', '1'])
plt.show()
```

```
<ipython-input-10-a8f9c9680dd1>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(x=label_counts.index, y=label_counts.values, palette='viridis')
```




```
train = Dataset.from_pandas(train_pd)
train
```

```
Dataset({
    features: ['text', 'keyword', 'labels'],
    num_rows: 6911
})
```

Я зашёл на hugging face и взял BERT

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: 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 s
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(
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 766B/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 3.40MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 6.71MB/s]
config.json: 100%  570/570 [00:00<00:00, 23.7kB/s]
```

Существует несколько способов токенизации, когда у нас несколько колонок. Объединим два столбца и разделим их служебным токеном

```
def tokenize(dataset: Dataset, tokenizer: BertTokenizer):
    combined_text = [f"{text} [KEYWORD] {keyword}" for text, keyword in zip(dataset["text"], dataset["keyword"])]

    tokenized_text = tokenizer(
        combined_text,
        truncation=True,
        padding='max_length',
        max_length=512
    )

    return tokenized_text
```

```
train = train.map(tokenize, batched=True, fn_kwargs={"tokenizer": tokenizer})
train = train.remove_columns(["keyword"])
train
```

```
Map: 100%                                    6911/6911 [00:12<00:00, 582.00 examples/s]
Dataset({
    features: ['text', 'labels', 'input_ids', 'token_type_ids', 'attention_mask'],
    num_rows: 6911
})
```

```
train = train.train_test_split(test_size=0.10).shuffle(seed=42)
train
```

```
DatasetDict({
    train: Dataset({
        features: ['text', 'labels', 'input_ids', 'token_type_ids', 'attention_mask'],
        num_rows: 6219
    })
    test: Dataset({
        features: ['text', 'labels', 'input_ids', 'token_type_ref', 'attention_mask'],
        num_rows: 692
    })
})
```

Вот и выполнен Препроцессинг данных. Мы избавились от мусора, как смогли, и токенизировали текст.

Модель

```
from transformers import PreTrainedModel, PretrainedConfig
from transformers.modeling_outputs import SequenceClassifierOutput
from transformers import BertModel
import torch.nn as nn


class TransformerConfig(PretrainedConfig):
    model_type = "transformer_clf"

    def __init__(self, base_transformer_model="bert-base-uncased", num_labels=2, **kwargs):
        self.base_transformer_model = base_transformer_model
        self.num_labels = num_labels
        super().__init__(**kwargs)

class TransformerClassificationModel(PreTrainedModel):
    config_class = TransformerConfig

    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels
        self.backbone = BertModel.from_pretrained(config.base_transformer_model)

        for param in self.backbone.parameters():
            param.requires_grad = False


        self.classifier = nn.Linear(self.backbone.config.hidden_size, config.num_labels)

    def forward(
        self,
        input_ids=None,
        attention_mask=None,
        token_type_ids=None,
        position_ids=None,
        head_mask=None,
        inputs_embeds=None,
        labels=None,
        output_attentions=None,
        output_hidden_states=None,
        return_dict=None,
    ):
        outputs = self.backbone(input_ids, attention_mask)
        embeddings = outputs.last_hidden_state[:, 0, :]
        logits = self.classifier(embeddings)

        loss = None
        if labels is not None:
            loss_fn = nn.CrossEntropyLoss()
            loss = loss_fn(logits.view(-1, self.num_labels), labels.view(-1))

        return SequenceClassifierOutput(
            loss=loss,
            logits=logits,
            hidden_states=outputs.hidden_states,
```

```
            attentions=outputs.attentions,
        )
```

```
from transformers import BertForSequenceClassification

bert = BertForSequenceClassification.from_pretrained('bert-base-uncased')
```

```
model.safetensors: 100%                                    440M/440M [00:05<00:00, 138MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly init
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)

    accuracy = accuracy_score(labels, preds)
    precision = precision_score(labels, preds, average='weighted')
    recall = recall_score(labels, preds, average='weighted')
    f1 = f1_score(labels, preds, average='weighted')

    return {
        'eval_accuracy': accuracy,
        'eval_precision': precision,
        'eval_recall': recall,
        'eval_f1': f1
    }
```

Обучение с замороженным backbone

```
config = TransformerConfig(base_transformer_model="bert-base-uncased", num_labels=2)
bert = TransformerClassificationModel(config)
```

```
import shutil
import os
from transformers import TrainingArguments, Trainer, AutoModelForSequenceClassification, DataCollatorWithPadding

dir = '/content/models'
output_dir = os.path.join(dir, 'Bert')
os.makedirs(output_dir, exist_ok=True)

model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels=2
)

batch_size = 32
num_epochs = 15
learning_rate = 2e-5
weight_decay = 0.001
evaluation_strategy = "epoch"
save_total_limit = 3
remove_unused_columns = True
report_to = "none"
padding = True
logging_steps = 1
metric_for_best_model = "eval_f1"
greater_is_better = True

training_args = TrainingArguments(
    output_dir=output_dir,
    evaluation_strategy=evaluation_strategy,
    learning_rate=learning_rate,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    weight_decay=weight_decay,
    save_total_limit=save_total_limit,
    num_train_epochs=num_epochs,
    remove_unused_columns=remove_unused_columns,
    report_to=report_to,
    logging_steps=logging_steps,
    load_best_model_at_end=True,
    metric_for_best_model=metric_for_best_model,
    save_strategy="epoch",
    fp16=True
)

data_collator = DataCollatorWithPadding(
```

```
    tokenizer=tokenizer,
    padding=padding
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train["train"],
    eval_dataset=train["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics
)

trainer.train()
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly init
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.11/dist-packages/transformers/training_args.py:1611: FutureWarning: `evaluation_strategy` is deprecated and w
  warnings.warn(
<ipython-input-20-b0c74a1780fc>:57: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__ini
  trainer = Trainer(
```

[2925/2925 40:26, Epoch 15/15]

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.490300	0.405906	0.832370	0.832557	0.832370	0.832457
2	0.535700	0.374934	0.842486	0.843239	0.842486	0.840580
3	0.135300	0.464807	0.832370	0.833824	0.832370	0.832844
4	0.029700	0.528716	0.838150	0.837900	0.838150	0.836787
5	0.038400	0.678318	0.786127	0.791591	0.786127	0.787382
6	0.065100	0.812002	0.822254	0.823143	0.822254	0.819628
7	0.026300	0.889621	0.813584	0.812735	0.813584	0.812226
8	0.000800	0.999794	0.815029	0.814139	0.815029	0.814014
9	0.000700	1.076225	0.813584	0.812706	0.813584	0.812751
10	0.000500	1.200043	0.812139	0.811269	0.812139	0.811362
11	0.000900	1.194839	0.823699	0.822943	0.823699	0.822608
12	0.000200	1.279109	0.823699	0.824234	0.823699	0.821335
13	0.000200	1.285382	0.822254	0.821738	0.822254	0.820688
14	0.000500	1.307117	0.822254	0.821634	0.822254	0.820826
15	0.000200	1.317642	0.822254	0.821738	0.822254	0.820688

```
TrainOutput(global_step=2925, training_loss=0.11504595718392804, metrics={'train_runtime': 2428.3231, 'train_samples_per_second':
38.415, 'train_steps_per_second': 1.205, 'total_flos': 2.45443147992576e+16, 'train_loss': 0.11504595718392804, 'epoch': 15.0})
```

```
trainer.save_model(output_dir)
tokenizer.save_pretrained(output_dir)
```

```
('/content/models/Bert/tokenizer_config.json',
 '/content/models/Bert/special_tokens_map.json',
 '/content/models/Bert/vocab.txt',
 '/content/models/Bert/added_tokens.json')
```

```
best_model = BertForSequenceClassification.from_pretrained(output_dir)
test_results = trainer.evaluate(eval_dataset=train["test"])
print("F1 на тестовом наборе:", test_results["eval_f1"])
```

[22/22 00:05]

```
F1 на тестовом наборе: 0.8405799715815363
```

Препроцессинг теста

```
drive.mount('/content/drive/', force_remount=True)

datasett = load_dataset(
    'csv',
    data_files=f'/content/test.csv',
    column_names=['id','keyword','location', 'text'],
)
```

```
Mounted at /content/drive/
```

```
datasett
```

```
DatasetDict({
    train: Dataset({
        features: ['id', 'keyword', 'location', 'text'],
        num_rows: 3264
    })
})
```

```
test_pd = pd.DataFrame(datasett['train'])
test_pd = pd.DataFrame(datasett['train'])
test_pd = test_pd.drop('id', axis=1)
test_pd = test_pd.drop('location', axis=1)
test_pd['text'] = list(map(clean_tweet, test_pd['text']))
```

```
test_pd
```

	keyword	text
0	keyword	text
1	None	Just happened a terrible car crash
2	None	Heard about earthquake is different cities, st...
3	None	there is a forest fire at spot pond, geese are...
4	None	Apocalypse lighting. Spokane wildfires
...	...	...
3259	None	EARTHQUAKE SAFETY LOS ANGELES SAFETY FASTENER...
3260	None	Storm in RI worse than last hurricane. My city...
3261	None	Green Line derailment in Chicago
3262	None	MEG issues Hazardous Weather Outlook HWO
3263	None	CityofCalgary has activated its Municipal Emer...

3264 rows × 2 columns

```
test_pd = test_pd.drop(0)
test_pd
```

	keyword	text
1	None	Just happened a terrible car crash
2	None	Heard about earthquake is different cities, st...
3	None	there is a forest fire at spot pond, geese are...
4	None	Apocalypse lighting. Spokane wildfires
5	None	Typhoon Soudelor kills 28 in China and Taiwan
...	...	...
3259	None	EARTHQUAKE SAFETY LOS ANGELES SAFETY FASTENER...
3260	None	Storm in RI worse than last hurricane. My city...
3261	None	Green Line derailment in Chicago
3262	None	MEG issues Hazardous Weather Outlook HWO
3263	None	CityofCalgary has activated its Municipal Emer...

3263 rows × 2 columns

```
test = Dataset.from_pandas(test_pd)
test
```

```
Dataset({
    features: ['keyword', 'text'],
    num_rows: 3263
})
```

```
test = test.map(tokenize, batched=True, fn_kwargs={"tokenizer": tokenizer})
test = test.remove_columns(["keyword"])
test
```

```
Map: 100%                                   3263/3263 [00:03<00:00, 1063.10 examples/s]
Dataset({
    features: ['text', 'input_ids', 'token_type_ids', 'attention_mask'],
    num_rows: 3263
})
```

```
predictions = trainer.predict(test)
```

```
print(predictions)
```

```
PredictionOutput(predictions=array([[-0.88720703,  0.7192383 ],
       [-1.5361328 ,  1.2070312 ],
       [-1.1582031 ,  0.72265625],
       ...,
       [-2.0566406 ,  1.7255859 ],
       [-1.7119141 ,  1.0732422 ],
       [-1.0927734 ,  0.5288086 ]], dtype=float32), label_ids=None, metrics={'test_runtime': 22.5963, 'test_samples_per_second': 144
```

```
predictions_array = predictions.predictions
```

```
print(predictions_array)
```

```
[[-0.88720703  0.7192383 ]
 [-1.5361328   1.2070312 ]
 [-1.1582031   0.72265625]
 ...
 [-2.0566406   1.7255859 ]
 [-1.7119141   1.0732422 ]
 [-1.0927734   0.5288086 ]]
```

```
predicted_classes = np.argmax(predictions_array, axis=1)
```

```
class_counts = np.bincount(predicted_classes)

print("Количество элементов в классе 0:", class_counts[0])
print("Количество элементов в классе 1:", class_counts[1])
```

```
Количество элементов в классе 0: 2051
Количество элементов в классе 1: 1212
```

```
print("Предсказанные классы:", predicted_classes)
```

```
Предсказанные классы: [1 1 1 ... 1 1 1]
```

```
for pred in predictions:
  print(pred)
```

```
[[-0.88720703  0.7192383 ]
 [-1.5361328   1.2070312 ]
 [-1.1582031   0.72265625]
 ...
 [-2.0566406   1.7255859 ]
 [-1.7119141   1.0732422 ]
 [-1.0927734   0.5288086 ]]
None
{'test_runtime': 22.5963, 'test_samples_per_second': 144.404, 'test_steps_per_second': 4.514}
```

```
sample_submission = pd.read_csv('sample_submission.csv')
sample_submission['target'] = predicted_classes
sample_submission.to_csv('/content/updated_submission.csv', index=False, quoting=1)
```

```
import json
def save_pretrained(self, save_directory):
    torch.save(self.state_dict(), f"{save_directory}/pytorch_model.bin")
    config = {
        "base_transformer_model": "bert-base-uncased",
        "num_labels": self.num_labels
    }
    with open(f"{save_directory}/config.json", "w") as f:
        json.dump(config, f)

TransformerClassificationModel.save_pretrained = save_pretrained
model.save_pretrained("my_model")
```

Submission and Description	Public Score
updated_submission.csv Complete · now	0.81121

```
model = BertForSequenceClassification.from_pretrained(output_dir)
cleaned_text = clean_tweet("@ Train Wreck caused by food accident!!!!!!!!!!!!")
inputs = tokenizer(
    cleaned_text,
    truncation=True,
    padding="max_length",
    max_length=512,
    return_tensors="pt"
)

# Перенос данных на устройство (CPU/GPU)
inputs = {k: v.to(model.device) for k, v in inputs.items()}

model.eval()
with torch.no_grad():
    outputs = model(**inputs)
    logits = outputs.logits
    predicted_class = torch.argmax(logits, dim=1).item()

if predicted_class == 0:
    print("No disaster")
else:
    print("Oh, no, disaster!")
```

```
Oh, no, disaster!
```