Описание процесса работы с моделью

Основные положения

Перед нами стоит `задача классификации` отзывов на фильмы на две категории: положительные и отрицательные, а также присвоение отзыву рейтинга от одного до десяти. С целью повышения точности работы модели будем использовать трансферное обучение, то есть проведем дообучение на датасете imdb редварительно обученной модели.

В качестве основы была выбрана модель `Bert`, которая использует трансформер. В отличие от направленных моделей, которые считывают вводимый текст последовательно, кодировщик трансформера считывает сразу всю последовательность слов, поэтому он считается двунаправленным. Эта особенность позволяет модели изучать контекст слова на основе всего его окружения, что повышает точность предсказаний модели.

Будем производить дообучение на `два класса` - позитивные и негативные отзывы. `Рейтинг` отзыва возьмем из параметра уверенности модели в своем предсказании. Например, если на некоторые входные данные ответ модели "отзыв позитивный с вероятностью 78%", то будем считать, что рейтинг этого отзыва равен 8-и из-за округления.

Подготовка данных

Для удобства дальнейшей работы с моделью, а именно ее интегрированием в веб-приложение, будем пользоваться инструментами платформы "Hugging face", позволяющими, например, загружать свои модели на удаленный сервер и с легкостью выгружать их обратно на любое устройство. Разработка решения данной задачи велась на двух устройствах: на более производительном стационарном компьютере производилась разработка модели и ее обучение, а на ноутбуке с операционной системой Ubuntu производилась работа с веб-приложением и его развертывание на удаленном сервере. Поэтому платформа "Hugging face" стала хорошим подспорьем в решении этой задачи.

Будем использовать датасет imdb. Для удобства так же воспользуемся платформой "Hugging face" и скачаем датасет, аналогичный оригиналу, который доступен по ссылке в условии задачи.

Логинимся на платформе:

```
from huggingface_hub import notebook_login

notebook_login()
```

```
{"model_id":"4e93c8ab04cc4b74bd6ea92760e62061","version_major":2,"vers
ion_minor":0}
```

Загружаем датасет

```
from datasets import load_dataset

imdb = load_dataset("imdb")
```

```
{"model_id":"1fbf489663e041ccb45359304530cb07","version_major":2,"vers
ion_minor":0}
```

```
C:\Users\User\AppData\Local\Programs\Python\Python311\Lib\site-
packages\huggingface_hub\file_download.py:147: UserWarning:
`huggingface_hub` cache-system uses symlinks by default to efficiently
store duplicated files but your machine does not support them in C:\
Users\User\.cache\huggingface\hub\datasets--imdb. Caching files will
still work but in a degraded version that might require more space on
your disk. This warning can be disabled by setting the
`HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more
details, see https://huggingface.co/docs/huggingface_hub/how-to-
cache#limitations.
To support symlinks on Windows, you either need to activate Developer
Mode or to run Python as an administrator. In order to activate
developer mode, see this article:
https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-
device-for-development
  warnings.warn(message)
```

```
{"model_id":"cfd3414b95414cc2b86d04e29ff08933","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"7c0ed0303af64ff28a0886378cfa25a3","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"4dca3b1e77044e9ea2d0e427ace35d81","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"7ba6da8d46094b35aa10eb03b8f457d7","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"e3d8cf75e03f460db3cb0c93ab48ab83","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"3b2f956f8f2b4c089f6102dcb2030808","version_major":2,"vers
ion_minor":0}
```

Пример экземпляра данных

```
imdb["test"][0]
```

```
{'text': 'I love sci-fi and am willing to put up with a lot. Sci-fi
movies/TV are usually underfunded, under-appreciated and
misunderstood. I tried to like this, I really did, but it is to good
TV sci-fi as Babylon 5 is to Star Trek (the original). Silly
```

```
prosthetics, cheap cardboard sets, stilted dialogues, CG that doesn\'t
match the background, and painfully one-dimensional characters cannot
be overcome with a \'sci-fi\' setting. (I\'m sure there are those of
you out there who think Babylon 5 is good sci-fi TV. It\'s not. It\'s
clichéd and uninspiring.) While US viewers might like emotion and
character development, sci-fi is a genre that does not take itself
seriously (cf. Star Trek). It may treat important issues, yet not as a
serious philosophy. It\'s really difficult to care about the
characters here as they are not simply foolish, just missing a spark
of life. Their actions and reactions are wooden and predictable, often
painful to watch. The makers of Earth KNOW it\'s rubbish as they have
to always say "Gene Roddenberry\'s Earth..." otherwise people would
not continue watching. Roddenberry\'s ashes must be turning in their
orbit as this dull, cheap, poorly edited (watching it without advert
breaks really brings this home) trudging Trabant of a show lumbers
into space. Spoiler. So, kill off a main character. And then bring him
back as another actor. Jeeez! Dallas all over again.',
 'label': 0}
```

Токенизация

Токенизатор позволяет разбивать текст на более мелкие единицы (токены). Такое разделение играет роль предобработки данных перед обучением.

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("distilbert/distilbert-base-
uncased")
```

```
The cache for model files in Transformers v4.22.0 has been updated.
Migrating your old cache. This is a one-time only operation. You can
interrupt this and resume the migration later on by calling
`transformers.utils.move_cache()`.
```


```
{"model_id":"c9f88e11c54847baaf63bea749bcb30e","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"7705c418421949069c339c83bdf95c3b","version_major":2,"vers
ion_minor":0}
```


```
C:\Users\User\AppData\Local\Programs\Python\Python311\Lib\site-
packages\huggingface_hub\file_download.py:147: UserWarning:
`huggingface_hub` cache-system uses symlinks by default to efficiently
store duplicated files but your machine does not support them in C:\
Users\User\.cache\huggingface\hub\models--distilbert--distilbert-base-
uncased. Caching files will still work but in a degraded version that
might require more space on your disk. This warning can be disabled by
setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable.
For more details, see https://huggingface.co/docs/huggingface_hub/how-
to-cache#limitations.
```

```
To support symlinks on Windows, you either need to activate Developer
Mode or to run Python as an administrator. In order to activate
developer mode, see this article:
https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-
device-for-development
  warnings.warn(message)
```


```
{"model_id":"e4ca5b1b235c4cfdb887983dac99d65e","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"3c694c8209f94bbebb66d0f410ab63ad","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"0d1dabfd83a14975ad6e15350434ed1b","version_major":2,"vers
ion_minor":0}
```


```
C:\Users\User\AppData\Local\Programs\Python\Python311\Lib\site-
packages\transformers\tokenization_utils_base.py:1617: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True`
by default. This behavior will be deprecated in transformers v4.45,
and will be then set to `False` by default. For more details check
this issue: https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
```

Создадим функцию предварительной токенизации текста.

```
def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True)
```

Произведем токенизацию на всем наборе данных.

```
tokenized_imdb = imdb.map(preprocess_function, batched=True)
```


```
{"model_id":"db8a0f5995e8496a985a7aee2e3a45d4","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"3b3528fc739847029f3e6e47408a8616","version_major":2,"vers
ion_minor":0}
```

```
{"model_id":"030b26cf8499447d863d31ebec36df06","version_major":2,"vers
ion_minor":0}
```


Создадим набор примеров

```
from transformers import DataCollatorWithPadding
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

Импорт метрики для обучения

Для оценки точности во время обучения воспользуемся метрикой `accuracy` (точность), которую предварительно импортируем

```
import evaluate

accuracy = evaluate.load("accuracy")
```

Создадим функцию для рассчета точности.

```
import numpy as np


def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    predictions = np.argmax(predictions, axis=1)
    return accuracy.compute(predictions=predictions,
references=labels)
```

Создадим вспомогательные словари, в которых укажем соответствие меток целевым значениям.

```
id2label = {0: "NEGATIVE", 1: "POSITIVE"}
label2id = {"NEGATIVE": 0, "POSITIVE": 1}
```

Обучение

Загрузим Bert. Обучение будем производить в две эпохи.

```
from transformers import AutoModelForSequenceClassification,
TrainingArguments, Trainer

model = AutoModelForSequenceClassification.from_pretrained(
    "distilbert/distilbert-base-uncased", num_labels=2,
id2label=id2label, label2id=label2id
)
```

```
WARNING:tensorflow:From C:\Users\User\AppData\Local\Programs\Python\
Python311\Lib\site-packages\tf_keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

```
{"model_id":"970c84404e154b3982866804afe95c4e","version_major":2,"vers
ion_minor":0}
```

```
Some weights of DistilBertForSequenceClassification were not
initialized from the model checkpoint at distilbert/distilbert-base-
```

```
uncased and are newly initialized: ['classifier.bias',
'classifier.weight', 'pre_classifier.bias', 'pre_classifier.weight']
You should probably TRAIN this model on a down-stream task to be able
to use it for predictions and inference.
```

```
training_args = TrainingArguments(
    output_dir="my_awesome_model",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=2,
    weight_decay=0.01,
    eval_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
    push_to_hub=True,
)

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

trainer.train()
```

```
<IPython.core.display.HTML object>
```

```
TrainOutput(global_step=3126, training_loss=0.20268608588708645,
metrics={'train_runtime': 21327.2348, 'train_samples_per_second':
2.344, 'train_steps_per_second': 0.147, 'total_flos':
6556904415524352.0, 'train_loss': 0.20268608588708645, 'epoch': 2.0})
```

Загрузим модель на сервер в свой личный кабинет. В дальнейшем будем импортировать ее именно оттуда.

```
trainer.push_to_hub()
```

```
CommitInfo(commit_url='https://huggingface.co/gordovaa/
my_awesome_model/commit/e804d8ef602c77aceeee54cbbf5532a942011656',
commit_message='End of training', commit_description='',
oid='e804d8ef602c77aceeee54cbbf5532a942011656', pr_url=None,
repo_url=RepoUrl('https://huggingface.co/gordovaa/my_awesome_model',
endpoint='https://huggingface.co', repo_type='model',
repo_id='gordovaa/my_awesome_model'), pr_revision=None, pr_num=None)
```

Тест модели

Чтобы иметь возможность **протестировать модель в ноутбуке**, не производя обучение перед этим, загрузим модель, которую сохранили в предыдущем пункте.

```
from transformers import pipeline
pipe = pipeline("text-classification",
model="gordovaa/my_awesome_model")

text = 'It is very nice film'
pipe(text)
```

```
[{'label': 'POSITIVE', 'score': 0.9834852814674377}]
```

В веб-приложении будем использовать аналог функции make_predictions, которая определена ниже. Отличие будет состоять в том, что входные данные будем получать из POST-запроса. В этой функции на основе score высчитываем рейтинг отзыва.

```
def make_predictions(text):
    pipe = pipeline("text-classification",
model="gordovaa/my_awesome_model")
    ans = pipe(text)[0]
    if ans['label'] == 'POSITIVE':
        print(ans['label'])
        star = round(ans['score'], 1)
        star *= 10
        star = int(star)
        print(star)
    else:
        print(ans['label'])
        star = round(1-ans['score'], 1)
        star *= 10
        if star == 0.0:
            star = 1.0
        star = int(star)
        print(star)

text = "There is a lot of ambiguity"
make_predictions(text)
```

```
POSITIVE
6
```

Подсчет метрик

Длина входного текста может оказаться больше максимально допустмой, что приведет к ошибкам. Поэтому воспользуемся исключениями try-except.

```
pipe = pipeline("text-classification",
model="gordovaa/my_awesome_model")
t_list, l_list, pred_list = [], [], []
for el in imdb['test']:
    t = el['text']
    l = el['label']
    try:
        p = pipe(t)[0]
        pred = p['score'] if p['label'] == 'POSITIVE' else 1-
p['score']
        t_list.append(t)
        l_list.append(l)
        pred_list.append(pred)
    except Exception:
        pass


data = {'feedback' : t_list, 'label' : l_list, 'prediction' :
pred_list}
df = pd.DataFrame(data)
df
```

```
Token indices sequence length is longer than the specified maximum
sequence length for this model (532 > 512). Running this sequence
through the model will result in indexing errors
```

	feedback	label	prediction
0	I love sci-fi and am willing to put up with a ...	0	0.007037
1	Worth the entertainment value of a rental, esp...	0	0.102681
2	its a totally average film with a few semi-alr...	0	0.013824
3	STAR RATING: ***** Saturday Night **** Friday ...	0	0.010641
4	First off let me say, If you haven't enjoyed a...	0	0.971133
...	...	...	...
21482	Two city guys are driving through Hicksville U...	1	0.658370
21483	I got this as part of a competition prize. I w...	1	0.493933
21484	I got Monster Man in a box set of three films ...	1	0.440260
21485	Five minutes in, i started to feel how naff th...	1	0.676139
21486	I caught this movie on the Sci-Fi channel rece...	1	0.914155

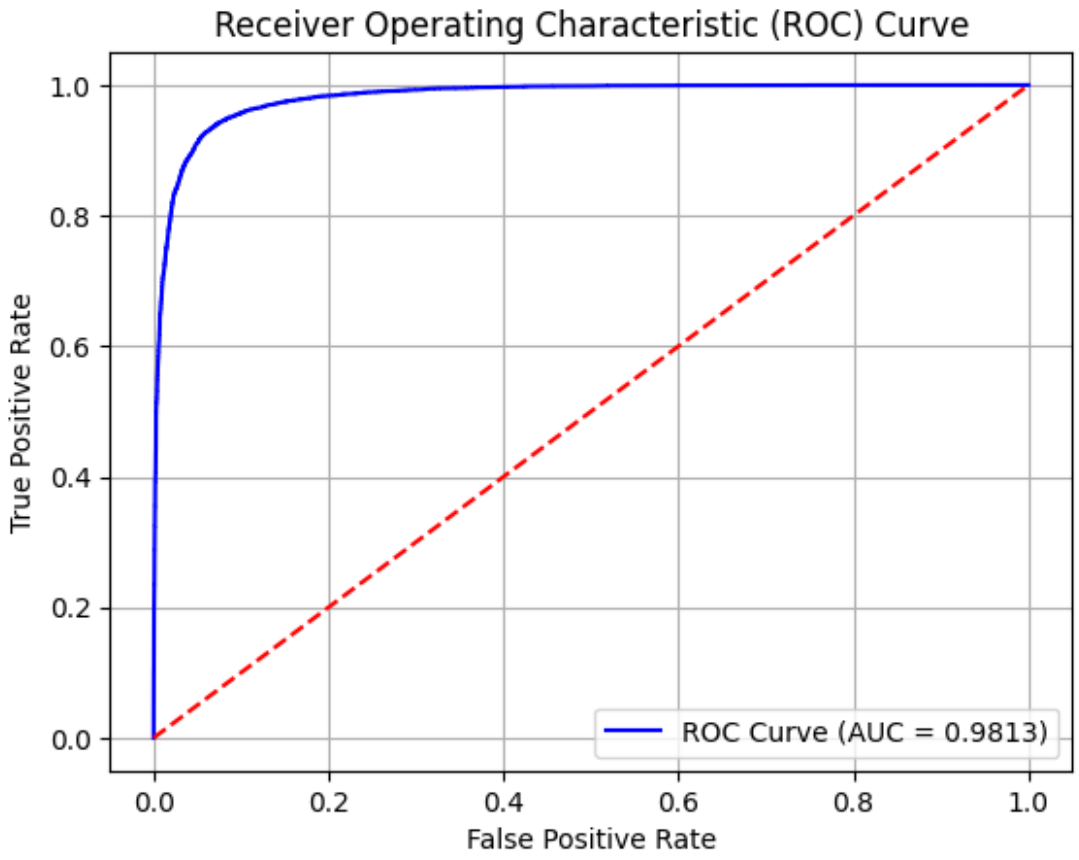
```
[21487 rows x 3 columns]
```

Строим ROC кривую

```
from sklearn.metrics import roc_curve, roc_auc_score,
precision_recall_curve, average_precision_score, f1_score
import seaborn as sns
import matplotlib.pyplot as plt
fpr, tpr, thresholds = roc_curve(df['label'], df['prediction'])
roc_auc = roc_auc_score(df['label'], df['prediction'])

plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC =
{roc_auc:.4f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid()
plt.show()

print(f'ROC AUC: {roc_auc:.4f}')
```



```
ROC AUC: 0.9813
```

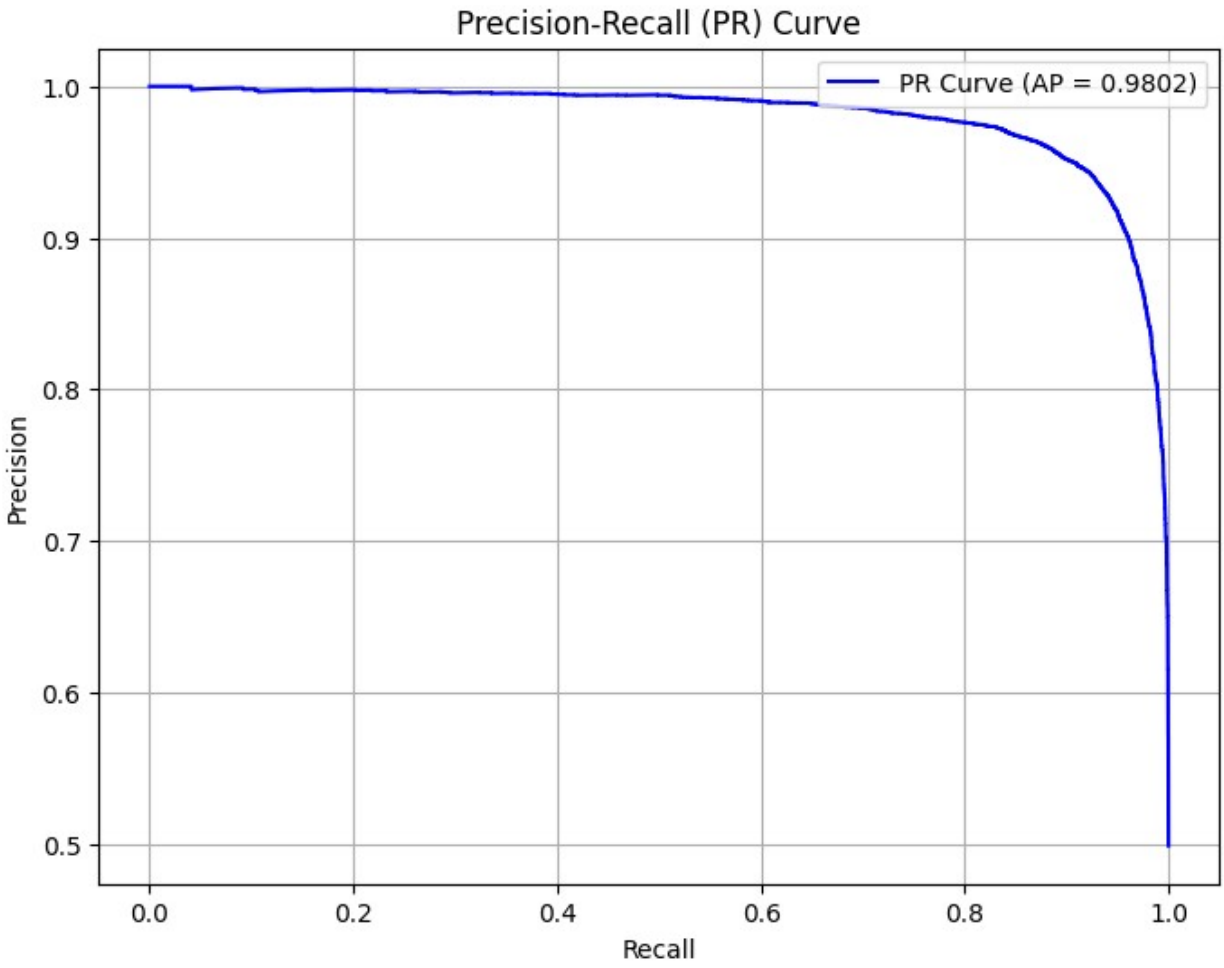
Строим precision-recall кривую

```
precision, recall, thresholds = precision_recall_curve(df['label'],
df['prediction'])

average_precision = average_precision_score(df['label'],
df['prediction'])

plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='blue', label=f'PR Curve (AP =
{average_precision:.4f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall (PR) Curve')
plt.legend(loc='upper right')
plt.grid()
plt.show()

print(f'Average Precision Score: {average_precision:.4f}')
```




```
Average Precision Score: 0.9802
```

f1 мера

```
labels = np.array(df["label"].tolist())
preds = np.array(df["prediction"].tolist())
new_f1 = f1_score(labels, preds)
print(f'F1 мера: {new_f1}')

F1 мера: 0.930494974397876
```

Полученные метрики подтверждают высокую точность результатов работы модели, поэтому будем интегрировать эту модель в веб-приложение на django.