作业1

公开数据集查找：[简体中文口语情感分类数据集 · 数据集](#)

简体中文多情感对话数据集，7类别，4159条数据，格式如下：

需要对10_BERT脚本进行如下修改：

1、去掉sep和header参数（默认sep=',',header是第一行）

```
# 加载和预处理数据
dataset_df = pd.read_csv("Simplified_Chinese_Multi-Emotion_Dialogue_Dataset.csv")
```

2、动态获取类别数量而不是硬编码，而且标签列索引是'label'而不是1（text列同理）

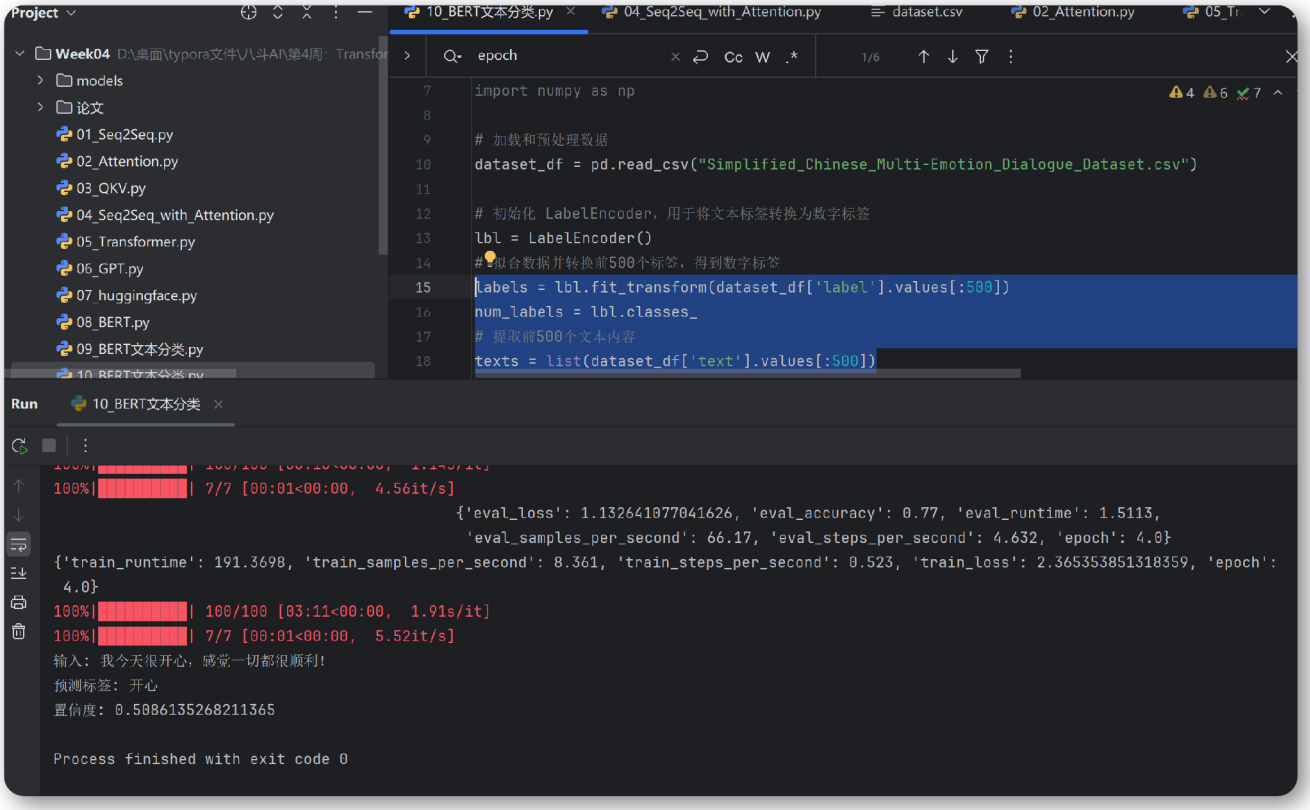
```
# 拟合数据并转换前500个标签，得到数字标签
labels = lbl.fit_transform(dataset_df['label'].values[:500])
num_labels = lbl.classes_
# 提取前500个文本内容
texts = list(dataset_df['text'].values[:500])
```

3、为了验证分类效果是否正确，需要添加以下函数来测试一个输入

```
def predict_sentence(sentence: str, model, tokenizer, lbl, max_length: int = 64, device: str =
None):
    model.eval()
    if device is None:
        device = "cuda" if torch.cuda.is_available() else "cpu"
    model.to(device)

    # 编码
    inputs = tokenizer(
        sentence,
        truncation=True,
        padding=True,
        max_length=max_length,
        return_tensors="pt"
    )
```

```
    inputs = {k: v.to(device) for k, v in inputs.items()}

    # 推理
    with torch.no_grad():
        outputs = model(**inputs)
        logits = outputs.logits  # [batch, num_labels]
        probs = torch.softmax(logits, dim=-1).cpu().numpy()[0]
        pred_id = int(np.argmax(probs))
        pred_label = lbl.inverse_transform([pred_id])[0]
        confidence = float(probs[pred_id])

    return pred_label, confidence, probs

# ═══ 测试一句话 ═══
test_sentence = "我今天很开心，感觉一切都很顺利！"
pred_label, conf, probs = predict_sentence(test_sentence, model, tokenizer, lbl)

print("输入:", test_sentence)
print("预测标签:", pred_label)
print("置信度:", conf)
```

最终脚本运行结果如下，训练4轮后的准确率为0.77，而且对于句子“我今天很开心，感觉一切都很顺利！"结果正确



作业2

项目部署：

1、首先使用如下命令将bert-base-chinese模型下载到本地

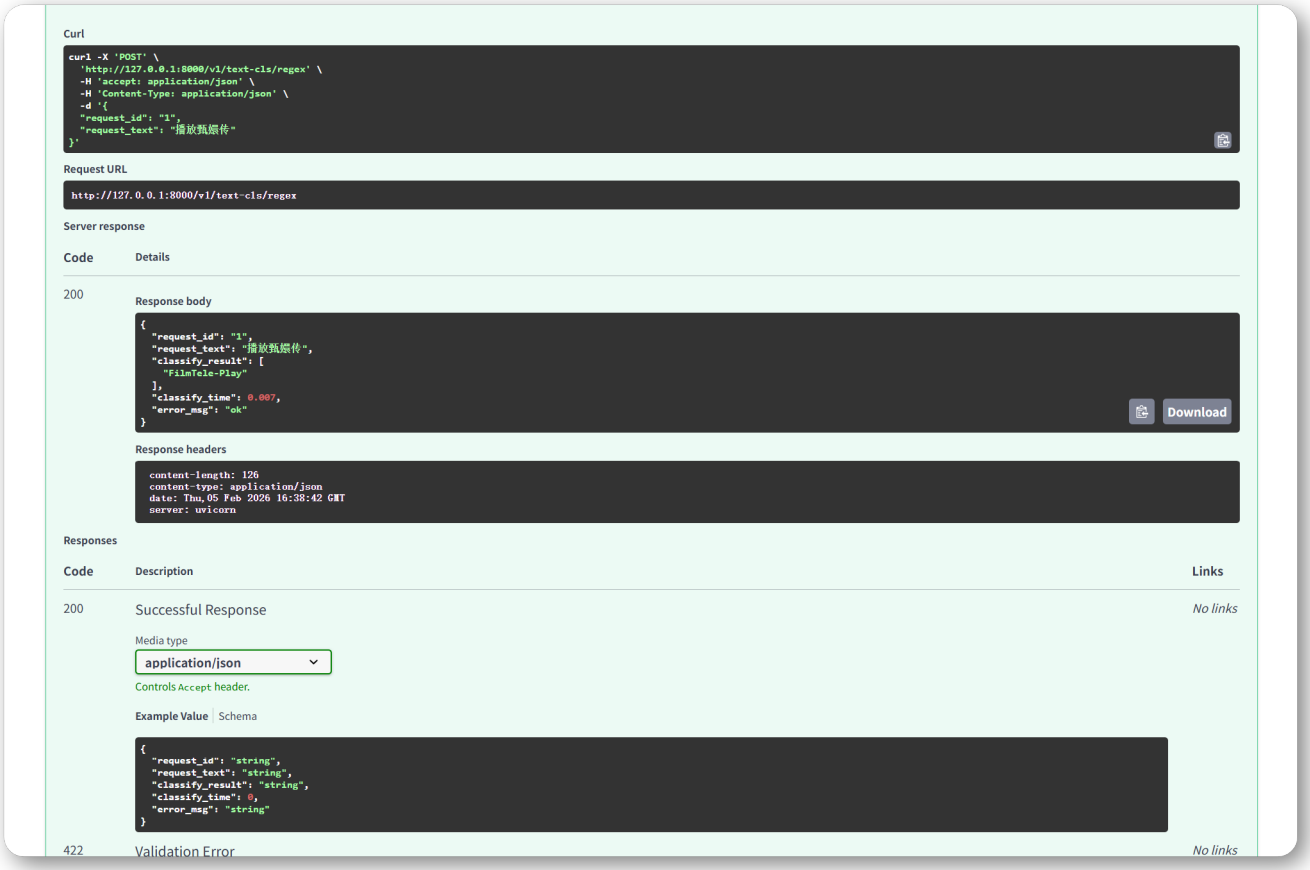
```
modelscope download --model google-bert/bert-base-chinese --local_dir models/google-bert/bert-base-chinese
```

2、运行 `training_code` 文件夹下的 `train_bert` ，`train_tfidf` 训练两个模型并保存到到项目中

3、在终端运行 `fastapi run main.py` 运行web服务实例，在http://127.0.0.1:8000/docs下进行测试

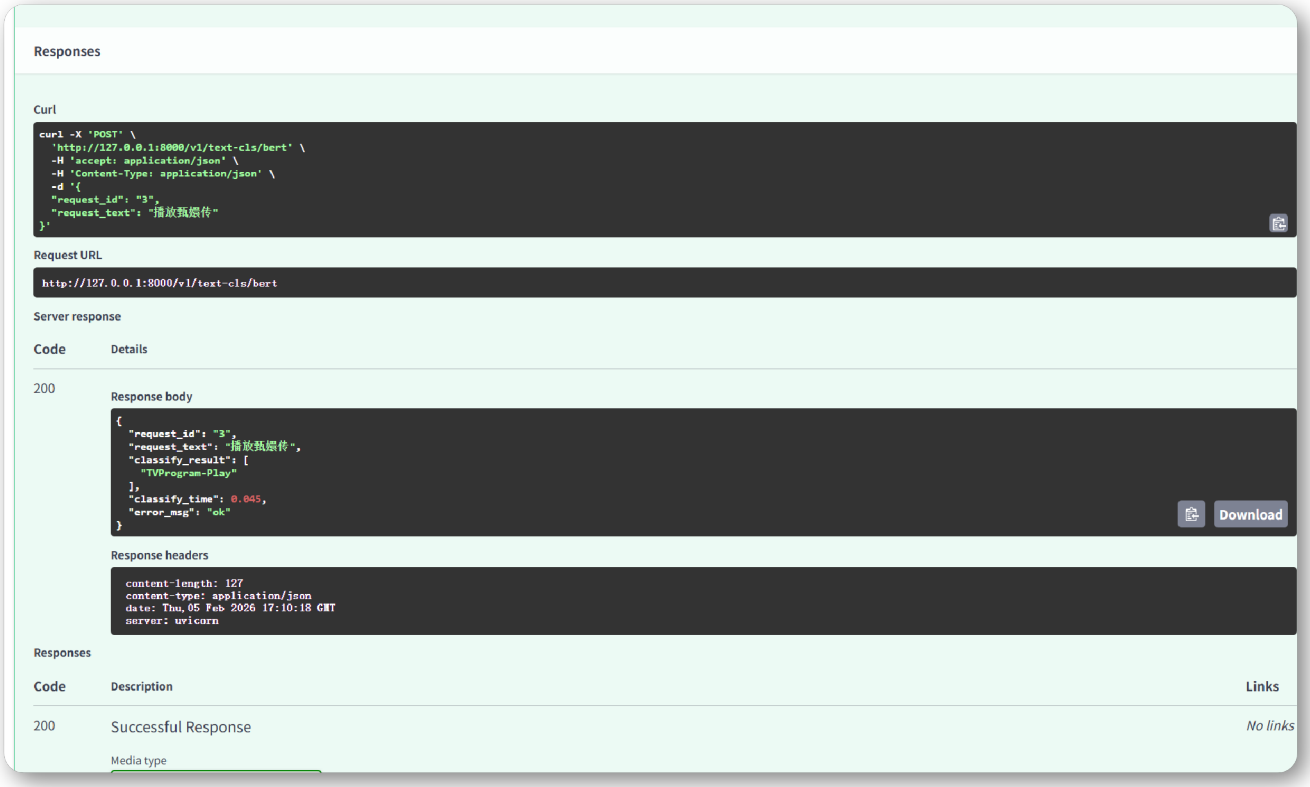
调用测试过程展示，使用句子：“播放甄嬛传”

正则表达式功能测试：

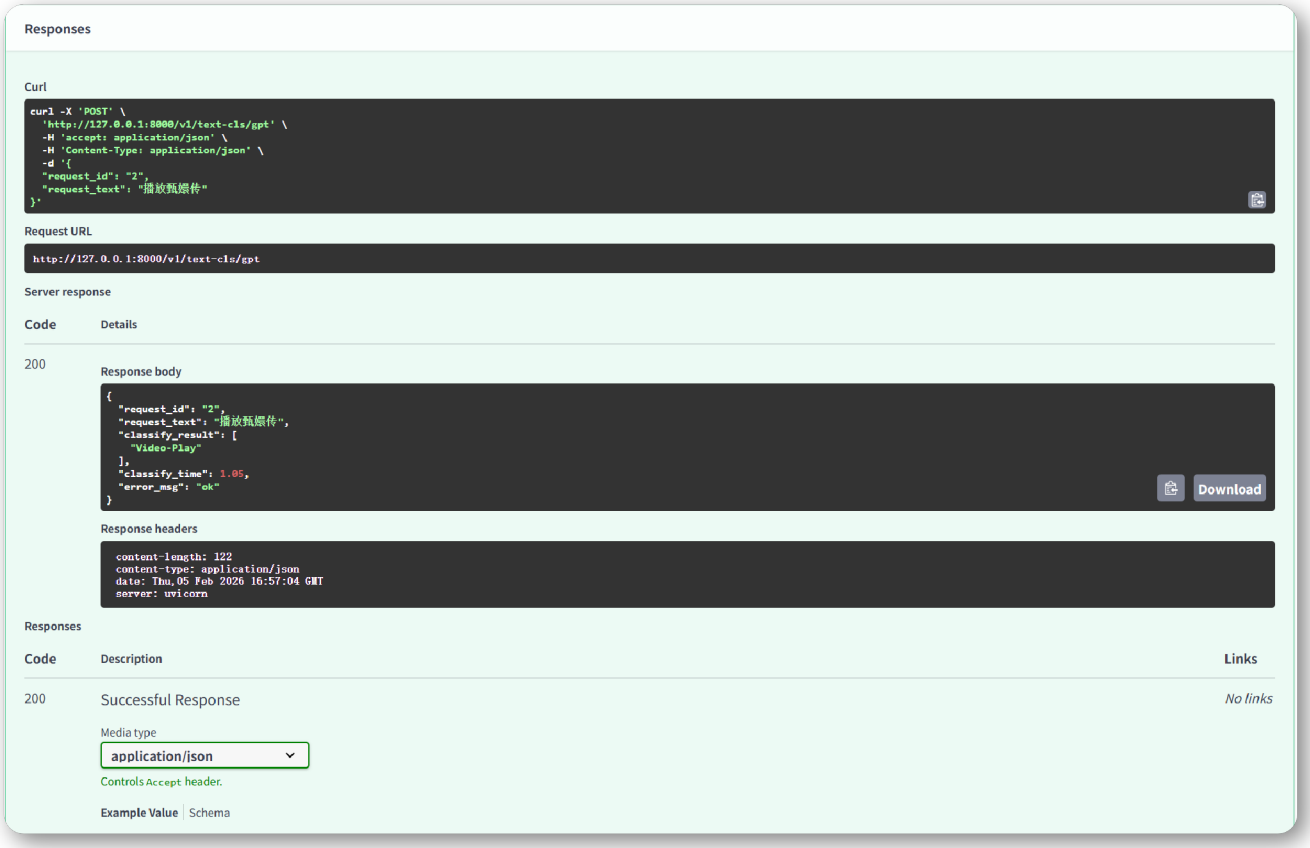


TfIdf分类结果：

bert模型预测结果：


Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/v1/text-cls/bert' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"request_id": "3",
"request_text": "播放甄嬛传"
}'
Request URL
http://127.0.0.1:8000/v1/text-cls/bert
Server response
Code
Details
200
Response body
{
"request_id": "3",
"request_text": "播放甄嬛传",
"classify_result": [
"TVProgram-Play"
],
"classify_time": 0.045,
"error_msg": "ok"
}
Download
Response headers
content-length: 127
content-type: application/json
date: Thu,05 Feb 2026 17:10:18 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type

llm分类结果：


Responses
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/v1/text-cls/gpt' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"request_id": "2",
"request_text": "播放甄嬛传"
}'
Request URL
http://127.0.0.1:8000/v1/text-cls/gpt
Server response
Code
Details
200
Response body
{
"request_id": "2",
"request_text": "播放甄嬛传",
"classify_result": [
"Video-Play"
],
"classify_time": 1.05,
"error_msg": "ok"
}
Download
Response headers
content-length: 122
content-type: application/json
date: Thu,05 Feb 2026 16:57:04 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.
Example Value
Schema