

# 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：



| text                      | label |
|---------------------------|-------|
| 你想不想去吃午饭？                 | 平静    |
| 哦！我被选中了！                  | 开心    |
| 我几大身体好像有点不太舒服，肚子好痛。       | 伤心    |
| 我的小组成员一个都没「活」真后悔跟他一起组队。   | 生气    |
| 他们是不是吵架了？不会打起来吧？          | 平静    |
| 你知道吗？我们班有人交了女朋友！          | 惊讶    |
| 干！他真的有够恶心，卫生习惯有够差。        | 厌恶    |
| 今天你要不要去骑自行车？              | 平静    |
| 我们今天班际篮球赛赢了！！！            | 开心    |
| 今天我和我朋友吵架了。               | 伤心    |
| 有人说我胖！该死！                 | 生气    |
| 我好像被我喜欢的人讨厌了。             | 平静    |
| 我刚刚看到有人眉毛像蜡笔小新！           | 惊讶    |
| 为什么我的便当里全是青椒啊！真的是很讨厌吃青椒的！ | 厌恶    |

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）

```
14     # 拟合数据并转换前500个标签，得到数字标签
15     Labels = lbl.fit_transform(dataset_df['label'].values[:500])
16     num_labels = lbl.classes_
17     # 提取前500个文本内容
18     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

The screenshot shows a PyCharm project titled 'Week04' with several files listed in the left sidebar, including '01\_Seq2Seq.py', '02\_Attention.py', '03\_QKV.py', '04\_Seq2Seq\_with\_Attention.py', '05\_Transformer.py', '06\_GPT.py', '07\_huggingface.py', '08\_BERT.py', '09\_BERT文本分类.py', and '10\_BERT文本分类.py'. The '10\_BERT文本分类.py' file is currently selected and open in the editor. The code in the editor is as follows:

```

7 import numpy as np
8
9 # 加载和预处理数据
10 dataset_df = pd.read_csv("Simplified_Chinese_Multi-Emotion_Dialogue_Dataset.csv")
11
12 # 初始化 LabelEncoder，用于将文本标签转换为数字标签
13 lbl = LabelEncoder()
14 # 合并数据并转换前500个标签，得到数字标签
15 labels = lbl.fit_transform(dataset_df['label'].values[:500])
16 num_labels = lbl.classes_
17 # 提取前500个文本内容
18 texts = list(dataset_df['text'].values[:500])

```

The terminal window at the bottom shows the execution of the script and its output:

```

100%|██████████| 100/100 [00:10<00:00,  1.14s/it]
100%|██████████| 7/7 [00:01<00:00,  4.56it/s]
{'eval_loss': 1.132641077041626, 'eval_accuracy': 0.77, 'eval_runtime': 1.5113,
 'eval_samples_per_second': 66.17, 'eval_steps_per_second': 4.632, 'epoch': 4.0}
{'train_runtime': 191.3698, 'train_samples_per_second': 8.361, 'train_steps_per_second': 0.523, 'train_loss': 2.365353851318359, 'epoch': 4.0}
100%|██████████| 100/100 [03:11<00:00,  1.91s/it]
100%|██████████| 7/7 [00:01<00:00,  5.52it/s]
输入: 我今天很开心，感觉一切都很顺利!
预测标签: 开心
置信度: 0.5086135268211365
Process finished with exit code 0

```

## 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：

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/v1/text-cls/regex' \
  -H 'Accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "request_id": "1",
    "request_text": "播放甄嬛传"
}'
```

Request URL

`http://127.0.0.1:8000/v1/text-cls/regex`

Server response

Code Details

200 Response body

```
{
  "request_id": "1",
  "request_text": "播放甄嬛传",
  "classify_result": [
    "FilmTele-Play"
  ],
  "classify_time": 0.007,
  "error_msg": "ok"
}
```

Response headers

```
content-length: 126
content-type: application/json
date: Thu, 05 Feb 2026 16:38:42 GMT
server: unicorn
```

Responses

| Code          | Description  | Links    |
|---------------|--|----------|
| 200           | Successful Response  | No links |
| Media type    | <code>application/json</code>  |          |
| Example Value | <a href="#">Schema</a>   |          |
|               | { "request_id": "string", "request_text": "string", "classify_result": "string", "classify_time": 0, "error_msg": "string" } |          |

422 Validation Error

No links

TfIdf分类结果：

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/v1/text-cls/tfidf' \
  -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/tfidf`

Server response

Code Details

200 Response body

```
{
  "request_id": "2",
  "request_text": "播放甄嬛传",
  "classify_result": [
    "FilmTele-Play"
  ],
  "classify_time": 0.009,
  "error_msg": "ok"
}
```

Response headers

```
content-length: 122
content-type: application/json
date: Thu, 05 Feb 2026 16:52:29 GMT
server: unicorn
```

Responses

| Code          | Description  | Links    |
|---------------|--|----------|
| 200           | Successful Response  | No links |
| Media type    | <code>application/json</code>  |          |
| Example Value | <a href="#">Schema</a>   |          |
|               | { "request_id": "string", "request_text": "string", "classify_result": "string", "classify_time": 0, "error_msg": "string" } |          |

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  | Links   |
|------|--|---|
| 200  | <p><b>Response body</b></p> <pre>{   "request_id": "3",   "request_text": "播放甄嬛传",   "classify_result": [     "TVProgram-Play"   ],   "classify_time": 0.045,   "error_msg": "ok" }</pre> <p><b>Response headers</b></p> <pre>content-length: 127 content-type: application/json date: Thu, 05 Feb 2026 17:10:18 GMT server: uvicorn</pre> | <a href="#">Copy</a> <a href="#">Download</a> |

**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   | Links   |
|------|---|---|
| 200  | <p><b>Response body</b></p> <pre>{   "request_id": "2",   "request_text": "播放甄嬛传",   "classify_result": [     "Video-Play"   ],   "classify_time": 1.05,   "error_msg": "ok" }</pre> <p><b>Response headers</b></p> <pre>content-length: 122 content-type: application/json date: Thu, 05 Feb 2026 16:57:04 GMT server: uvicorn</pre> | <a href="#">Copy</a> <a href="#">Download</a> |

**Responses**

| Code | Description         | Links    |
|------|---------------------|----------|
| 200  | Successful Response | No links |

**Media type**

**application/json**

Controls Accept header.

[Example Value](#) | [Schema](#)