基于ChatGLM3-6B模型进行文本情感分类分析

- 杨君

背景

文本情感分析是自然语言处理领域的一个重要应用，旨在识别和提取文本中的情感信息，包括情感极性、情感强度和情感类型等。例如：某用户在社交媒体上发布了一条评论：“这家餐厅的服务太差了！菜也不好吃。” 我们可以明确地识别出以下情感信息：

- 情感极性：负面。因为评论中使用了“太差了”和“不好吃”这两个否定词汇，表明了作者对餐厅的不满。
- 情感强度：中等。虽然作者表达了不满，但并没有使用过于强烈的词汇，因此可以判断情感强度为中等。
- 情感类型：愤怒。由于作者对餐厅的服务和菜品表示不满，可以判断出作者的情感类型为愤怒。情感分析对于企业或组织来说具有重要的意义，有效分析可以帮助他们及时发现用户的负面评价，并采取相应的措施来改善产品和服务质量。

2022年年底OpenAI发布ChatGPT，将LLM（Large Language Model）带向了一个新的高度，而2023年OpenAI继续放出大招：更强大的GPT-4问世，瞬间引爆了整个互联网圈。LLM(大规模语言模型）技术通过在大量语料之上进行预训练，使得模型能够“记住”语料中的知识，包含常识、语法、句法、语义等，从而能够对语言进行“理解”，进而能应用于下游的文本理解任务（如分类、NER、信息检索），以及文本生成任务（如对话、摘要、故事生成）。

大语言模型（LLM）已经从新兴技术发展为主流技术。而以大模型为核心技术的产品将迎来全新迭代。ChatGLM3是智谱AI和清华大学 KEG 实验室联合发布的新一代对话预训练模型。在语义、数学、推理、代码、知识等不同角度的数据集上测评显示，具有在 10B 以下的基础模型中最强的性能。本项目使用国内开源LLM，探索了LLM技术的在文本情感分类中的应用。

目的

本项目基于ChatGLM3-6B进行"电影影评情感极性分析"任务，并尝试优化prompt来探索LLM的文本分析能力。

环境搭建过程

机器规格

- 使用阿里云弹性加速计算实例EAIS，具体实例规格如下：

安装依赖

```
pip install protobuf 'transformers>=4.30.2' cpm_kernels 'torch>=2.0' gradio mdtex2html sentence
pip install modelscope
```

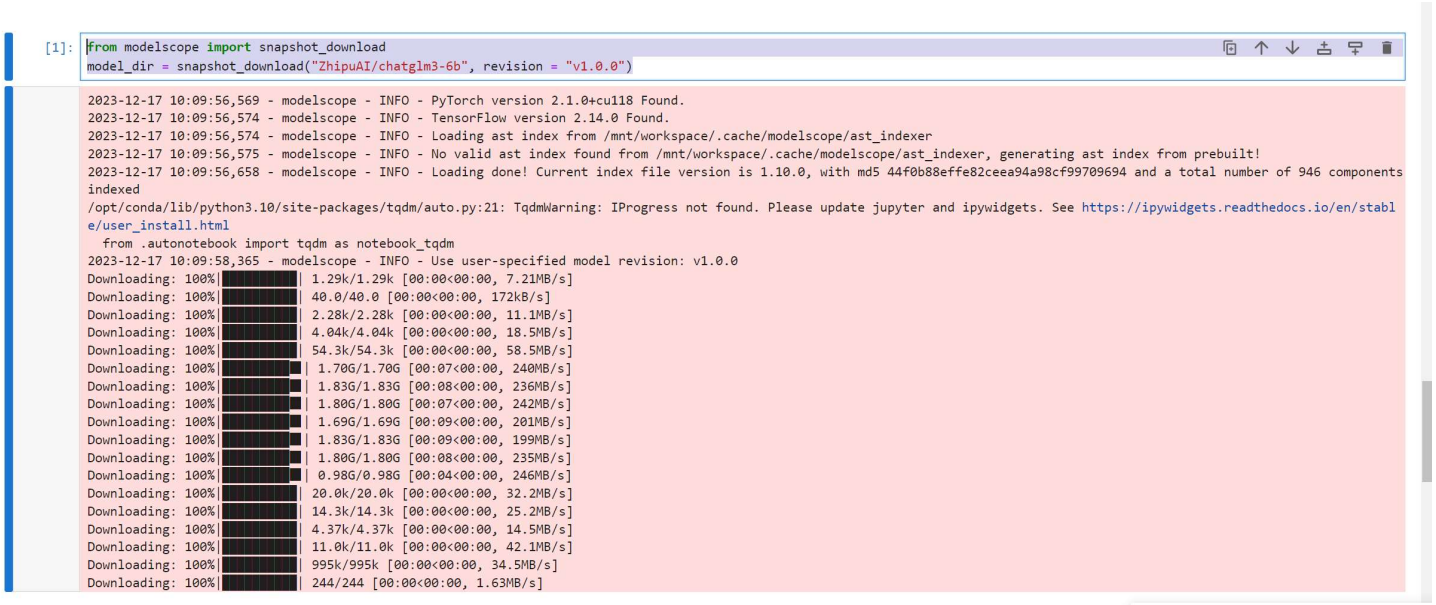
```
[1]: pip install protobuf 'transformers>=4.30.2' cpm_kernels 'torch>=2.0' gradio mdtex2html sentencepiece accelerate

Looking in indexes: https://mirrors.cloud.aliyuncs.com/pypi/simple
Requirement already satisfied: protobuf in /opt/conda/lib/python3.10/site-packages (3.20.3)
Requirement already satisfied: transformers>=4.30.2 in /opt/conda/lib/python3.10/site-packages (4.34.1)
Collecting cpm_kernels
  Downloading https://mirrors.cloud.aliyuncs.com/pypi/packages/af/84/1831ce6ffa87b8fd4d9673c3595d0fc4e6631c0691eb43f406d3bf89b951/cpm_kernels-1.0.11-py3-none-any.whl (416 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 416.6/416.6 kB 18.9 MB/s eta 0:00:00
Requirement already satisfied: torch>=2.0 in /opt/conda/lib/python3.10/site-packages (2.1.0+cu118)
Collecting gradio
  Downloading https://mirrors.cloud.aliyuncs.com/pypi/packages/16/9b/46c017d6bbc60ccb13ef51af4b07dbb288b9080ea3170b8d47beee7842b6/gradio-4.10.0-py3-none-any.whl (16.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.6/16.6 MB 77.6 MB/s eta 0:00:0000:0100:01
Collecting mdtex2html
  Downloading https://mirrors.cloud.aliyuncs.com/pypi/packages/47/fa/5156a032ad68f6c32ae0dc3aaf8b3d690004b42497d4735a08bb4cea6ec3/mdtex2html-1.2.0-py3-none-any.whl (13 kB)
Requirement already satisfied: sentencepiece in /opt/conda/lib/python3.10/site-packages (0.1.99)
Requirement already satisfied: accelerate in /opt/conda/lib/python3.10/site-packages (0.24.1)
Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (3.13.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (0.19.4)
Requirement already satisfied: numpy>=1.17 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (1.26.1)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (2023.10.3)
Requirement already satisfied: requests in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (2.31.0)
Requirement already satisfied: tokenizers<0.15,>=0.14 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (0.14.1)
Requirement already satisfied: safetensors>=0.3.1 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (0.4.0)
Requirement already satisfied: tqdm>=4.27 in /opt/conda/lib/python3.10/site-packages (from transformers>=4.30.2) (4.65.0)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.10/site-packages (from torch>=2.0) (4.8.0)
Requirement already satisfied: sympy in /opt/conda/lib/python3.10/site-packages (from torch>=2.0) (1.12)
Requirement already satisfied: networkx in /opt/conda/lib/python3.10/site-packages (from torch>=2.0) (3.2.1)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/site-packages (from torch>=2.0) (3.1.2)
Requirement already satisfied: fsspec in /opt/conda/lib/python3.10/site-packages (from torch>=2.0) (2023.10.0)
Requirement already satisfied: triton==2.1.0 in /opt/conda/lib/python3.10/site-packages (from torch>=2.0) (2.1.0)
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading https://mirrors.cloud.aliyuncs.com/pypi/packages/c5/19/5af6804c4cc0fed83f47bff6e413a98a36618e7d40185cd36e69737f3b0e/aiofiles-23.2.1-py3-none-any.whl (15 kB)
Collecting altair<6.0,>=4.2.0 (from gradio)
```

下载ChatGLM3-6B模型模型

- 从modelscope上下载

```
from modelscope import snapshot_download
model_dir = snapshot_download("ZhipuAI/chatglm3-6b", revision = "v1.0.0")
```



调用 ChatGLM3-6B 模型来生成对话

```
from modelscope import AutoTokenizer, AutoModel, snapshot_download
model_dir = snapshot_download("ZhipuAI/chatglm3-6b", revision = "v1.0.0")
tokenizer = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)
model = AutoModel.from_pretrained(model_dir, trust_remote_code=True).half().cuda()
model = model.eval()
response, history = model.chat(tokenizer, "你好", history=[])
print(response)
response, history = model.chat(tokenizer, "晚上睡不着应该怎么办", history=history)
print(response)
```

[2]:
```
from modelscope import AutoTokenizer, AutoModel, snapshot_download
model_dir = snapshot_download("ZhipuAI/chatglm3-6b", revision = "v1.0.0")
tokenizer = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)
model = AutoModel.from_pretrained(model_dir, trust_remote_code=True).half().cuda()
model = model.eval()
response, history = model.chat(tokenizer, "你好", history=[])
print(response)
response, history = model.chat(tokenizer, "晚上睡不着应该怎么办", history=history)
print(response)
```

```
2023-12-17 10:17:55.486080: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN w
hen one has already been registered
2023-12-17 10:17:55.486130: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT wh
en one has already been registered
2023-12-17 10:17:55.486158: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLA
S when one has already been registered
2023-12-17 10:17:55.495418: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical o
perations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-12-17 10:17:56.782644: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2023-12-17 10:17:57,829 - modelscope - INFO - Use user-specified model revision: v1.0.0
Loading checkpoint shards: 100%|██████████| 7/7 [00:10<00:00,  1.54s/it]
```

你好👋！我是人工智能助手 ChatGLM3-6B，很高兴见到你，欢迎问我任何问题。
晚上睡不着可以尝试以下方法：

1. 保持安静：尽量避免噪音和光线，营造一个舒适的环境。

2. 放松身心：可以尝试深呼吸、冥想、瑜伽等方式来放松身心。

3. 规律作息：保持每天固定的作息时间，有规律地睡觉和起床。

4. 控制使用电子设备的时间：睡前避免使用电子设备，尤其是带有蓝光的设备，因为蓝光会干扰褪黑激素的分泌，影响睡眠质量。

5. 合理饮食：尽量避免晚餐过量，不要吃辛辣、刺激性的食物，保持饮食清淡。

6. 锻炼身体：适当的锻炼可以帮助入睡，但避免在睡前进行剧烈运动。

7. 洗个热水澡：泡个热水澡可以帮助放松身体，缓解疲劳，有利于入睡。

8. 尝试睡前习惯：可以在睡前喝一杯牛奶、听一些轻柔的音乐、读一本书等方式，建立一个良好的睡前习惯。

如果以上方法不能解决问题，建议咨询专业医生或心理学家，获得更具体的建议和治疗。

电影影评情感数据准备

从互联网上获取IMDb电影评论。从test集中选取了pos 10个，neg 10个。详见data目录

ChatGLM3-6B进行情感分类任务

使用Zero-Shot prompt进行文本情感分类。

- 例子1

```
from modelscope import AutoTokenizer, AutoModel, snapshot_download
model_dir = snapshot_download("ZhipuAI/chatglm3-6b", revision = "v1.0.0")
tokenizer = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)
model = AutoModel.from_pretrained(model_dir, trust_remote_code=True).half().cuda()
model = model.eval()

prompt = """你是一个情感分析专家。请判断以下句子的情感是什么，情感从['正面的', '负面的', '中性的']里

给出的句子是：
弱到爆…剧情、氛围烘托、表演、服化道、特效，统统弱到爆！女主和哥哥转折超级无力，给女主默默烫发来表现阶
转折也是醉了…，除了小迅猛龙看起来像狗以外，其余真是…吐槽无力

回答：
"""

response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

- 例子2

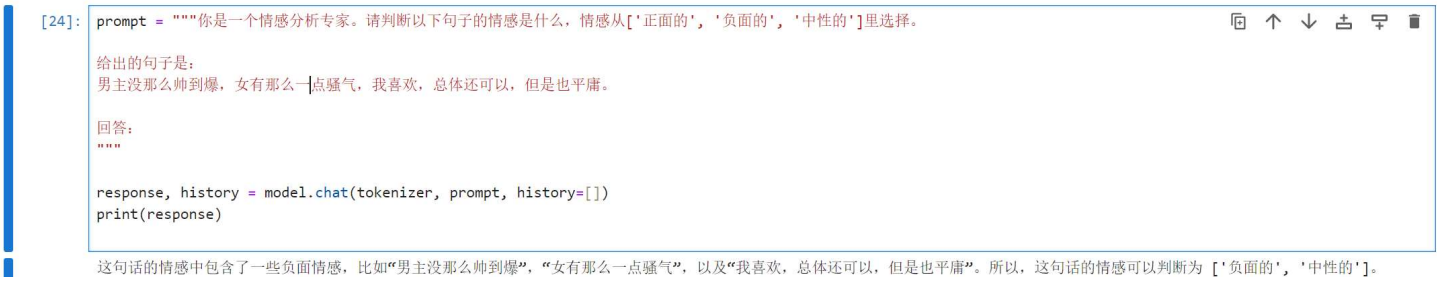
```
from modelscope import AutoTokenizer, AutoModel, snapshot_download
model_dir = snapshot_download("ZhipuAI/chatglm3-6b", revision = "v1.0.0")
tokenizer = AutoTokenizer.from_pretrained(model_dir, trust_remote_code=True)
model = AutoModel.from_pretrained(model_dir, trust_remote_code=True).half().cuda()
model = model.eval()

prompt = """你是一个情感分析专家。请判断以下句子的情感是什么，情感从['正面的', '负面的', '中性的']里

给出的句子是：
男主没那么帅到爆，女有那么一点骚气，我喜欢，总体还可以，但是也平庸。

回答：
"""

response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```



- 例子3，答案错误

```
prompt = """你是一个情感分析专家。请判断以下句子的情感是什么，情感从['正面的', '负面的', '中性的']里

给出的句子是：
我今天中午吃了玉米排骨汤和红烧肉

回答：
"""
response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

[25]:
```
prompt = """你是一个情感分析专家。请判断以下句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。

给出的句子是：
我今天中午吃了玉米排骨汤和红烧肉

回答：
"""
response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

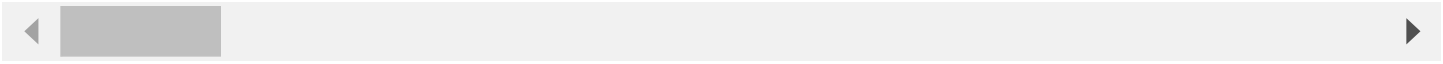
```
['正面的']
```

- 例子4，输出结果不唯一

```
prompt = """你是一个情感分析专家。请判断以下句子的情感是什么，情感从['正面的', '负面的', '中性的']里

给出的句子是：
I went and saw this movie last night after being coaxed to by a few friends of mine. I'll admit

回答：
"""
response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```



[26]:
```
prompt = """你是一个情感分析专家。请判断以下句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。

给出的句子是：
I went and saw this movie last night after being coaxed to by a few friends of mine. I'll admit that I was reluctant to see it because from what I knew of Asht

回答：
"""
response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

```
['正面的', '积极的']
```

使用few-shot prompting进行优化

few-shot prompting通过给定一些例子，引导LLM按照示例来理解任务，并按指定格式输出结果。few-shot prompting优化如下：

+

- 例子1

```
prompt = """你是一个情感分析专家。你应该判断一整段话的情感是什么，情感从['正面的', '负面的', '中性的
如果不存在，回答：没有。
返回结果为输出列表。

下面是一些例子
"值得去的地方，石头很奇特，景色优美，环境宜人，适合与朋友家人一起游玩"  ->  ["正面的"]
"弱到爆…剧情、氛围烘托、表演、服化道、特效，统统弱到爆！女主和哥哥转折超级无力，给女主默默烫发来表现
转折也是醉了…，除了小迅猛龙看起来像狗以外，其余真是…吐槽无力"  ->  ["负面的"]
"Adrian Pasdar is excellent is this film. He makes a fascinating woman."  ->  ["正面的"]
"This movie is terrible but it has some good effects." ->  ["负面的"]
"我今天中午吃了玉米排骨汤和红烧肉"    ->  没有


你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
输出列表：["正面的"]
如果不存在，回答：没有。
返回结果为输出列表。


现在，我给你一段话：
"I went and saw this movie last night after being coaxed to by a few friends of mine. I'll admi
你应该判断这段话的情感倾向，并以列表的形式返回结果，如果不存在，则回答：没有。
"""

response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

[20]:
```
prompt = """你是一个情感分析专家。你应该判断一整段话的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
如果不存在，回答：没有。
返回结果为输出列表。

下面是一些例子
"值得去的地方，石头很奇特，景色优美，环境宜人，适合与朋友家人一起游玩"  ->  ["正面的"]
"弱到爆…剧情、氛围烘托、表演、服化道、特效，统统弱到爆！女主和哥哥转折超级无力，给女主默默烫发来表现阶段性
转折也是醉了…，除了小迅猛龙看起来像狗以外，其余真是…吐槽无力"  ->  ["负面的"]
"Adrian Pasdar is excellent is this film. He makes a fascinating woman."  ->  ["正面的"]
"This movie is terrible but it has some good effects." ->  ["负面的"]
"我今天中午吃了玉米排骨汤和红烧肉"    ->  没有


你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
输出列表：["正面的"]
如果不存在，回答：没有。
返回结果为输出列表。


现在，我给你一段话：
"I went and saw this movie last night after being coaxed to by a few friends of mine. I'll admit that I was reluctant to
你应该判断这段话的情感倾向，并以列表的形式返回结果，如果不存在，则回答：没有。
"""

response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

输出列表：["正面的"]

- 例子2

```
prompt = """你是一个情感分析专家。你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的'
如果不存在，回答：没有。
返回结果为输出列表。

下面是一些例子
"值得去的地方，石头很奇特，景色优美，环境宜人，适合与朋友家人一起游玩"  ->  ["正面的"]
"弱到爆…剧情、氛围烘托、表演、服化道、特效，统统弱到爆！女主和哥哥转折超级无力，给女主默默烫发来表现
转折也是醉了…，除了小迅猛龙看起来像狗以外，其余真是…吐槽无力"  ->  ["负面的"]
"Adrian Pasdar is excellent is this film. He makes a fascinating woman."  ->  ["正面的"]
"This movie is terrible but it has some good effects." ->  ["负面的"]
"我今天中午吃了玉米排骨汤和红烧肉"    ->  没有


你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
输出列表：["正面的"]
如果不存在，回答：没有。
返回结果为输出列表。


现在，我给你一个句子：
"我今天中午吃了玉米排骨汤和红烧肉"
你应该判断句子的情感倾向，并以列表的形式返回结果，如果不存在，则回答：没有。
"""

response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

[21]:
```
prompt = """你是一个情感分析专家。你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
如果不存在，回答：没有。
返回结果为输出列表。

下面是一些例子
"值得去的地方，石头很奇特，景色优美，环境宜人，适合与朋友家人一起游玩"  ->  ["正面的"]
"弱到爆…剧情、氛围烘托、表演、服化道、特效，统统弱到爆！女主和哥哥转折超级无力，给女主默默烫发来表现阶段性
转折也是醉了…，除了小迅猛龙看起来像狗以外，其余真是…吐槽无力"  ->  ["负面的"]
"Adrian Pasdar is excellent is this film. He makes a fascinating woman."  ->  ["正面的"]
"This movie is terrible but it has some good effects." ->  ["负面的"]
"我今天中午吃了玉米排骨汤和红烧肉"    ->  没有


你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
输出列表：["正面的"]
如果不存在，回答：没有。
返回结果为输出列表。


现在，我给你一个句子：
"我今天中午吃了玉米排骨汤和红烧肉"
你应该判断句子的情感倾向，并以列表的形式返回结果，如果不存在，则回答：没有。
"""

response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

该句子与情感分析无关，因此不需要进行情感分析。

- 例子3

```
prompt = """你是一个情感分析专家。你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的'
如果不存在，回答：没有。
返回结果为输出列表。

下面是一些例子
"值得去的地方，石头很奇特，景色优美，环境宜人，适合与朋友家人一起游玩"  ->  ["正面的"]
"弱到爆…剧情、氛围烘托、表演、服化道、特效，统统弱到爆！女主和哥哥转折超级无力，给女主默默烫发来表现阶
转折也是醉了…，除了小迅猛龙看起来像狗以外，其余真是…吐槽无力"  ->  ["负面的"]
"Adrian Pasdar is excellent is this film. He makes a fascinating woman."  ->  ["正面的"]
"This movie is terrible but it has some good effects." ->  ["负面的"]
"我今天中午吃了玉米排骨汤和红烧肉"    ->  没有


你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
输出列表：["正面的"]
如果不存在，回答：没有。
返回结果为输出列表。


现在，我给你一个句子：
"Widow hires a psychopath as a handyman. Sloppy film noir thriller which doesn't make much of i
你应该判断句子的情感倾向，并以列表的形式返回结果，如果不存在，则回答：没有。
"""

response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

[31]:
```
prompt = """你是一个情感分析专家。你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
如果不存在，回答：没有。
返回结果为输出列表。

下面是一些例子
"值得去的地方，石头很奇特，景色优美，环境宜人，适合与朋友家人一起游玩"  ->  ["正面的"]
"弱到爆…剧情、氛围烘托、表演、服化道、特效，统统弱到爆！女主和哥哥转折超级无力，给女主默默烫发来表现阶段性
转折也是醉了…，除了小迅猛龙看起来像狗以外，其余真是…吐槽无力"  ->  ["负面的"]
"Adrian Pasdar is excellent is this film. He makes a fascinating woman."  ->  ["正面的"]
"This movie is terrible but it has some good effects." ->  ["负面的"]
"我今天中午吃了玉米排骨汤和红烧肉"    ->  没有


你应该判断该句子的情感是什么，情感从['正面的', '负面的', '中性的']里选择。
输出列表：["正面的"]
如果不存在，回答：没有。
返回结果为输出列表。


现在，我给你一个句子：
"Widow hires a psychopath as a handyman. Sloppy film noir thriller which doesn't make much of its tension promising set-up. (3/10)"
你应该判断句子的情感倾向，并以列表的形式返回结果，如果不存在，则回答：没有。
"""

response, history = model.chat(tokenizer, prompt, history=[])
print(response)
```

```
该句子的情感倾向是['负面的']。
```

结论和思考

通过基于ChatGLM3-6B进行"电影影评情感极性分析"任务，探索了大语言模型技术应用；通过prompt优化就可以更好的利用LLM的能力，并且同时支持英文和中文多语言的文本情感分析。
在实践过程中发现多次运行时结果会有随机性，后续可以继续探索temperature、top_p等参数，对LLM模型应用的影响。