Xtuner微调LLM：1.8B、多模态、Agent

1. XTuner微调个人小助手认知

1.1 环境安装

```
conda create --name xtuner0.1.17 python=3.10 -y
# 激活环境
conda activate xtuner0.1.17
# 创建版本文件夹并进入，以跟随本教程
mkdir -p xtuner0117 && cd xtuner0117
# 拉取 0.1.17 的版本源码
git clone -b v0.1.17  https://github.com/InternLM/xtuner
# 进入源码目录
cd xtuner
# 从源码安装 XTuner
pip install -e '.[all]'
```

1.2 前期准备

1.2.1 数据集准备

先创建一个文件夹来存放我们这次训练所需要的所有文件。

```
# 前半部分是创建一个文件夹，后半部分是进入该文件夹。
mkdir -p ft && cd ft

# 在ft这个文件夹里再创建一个存放数据的data文件夹
mkdir -p ft/data && cd ft/data
```

之后我们可以在 `data` 目录下新建一个 `generate_data.py` 文件，将以下代码复制进去，然后运行该脚本即可生成数据集。

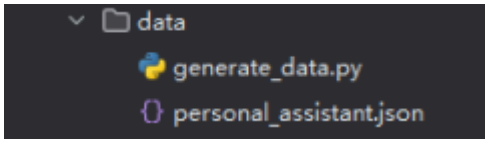
```
# 创建 `generate_data.py` 文件
touch ft/data/generate_data.py
```

```python
import json

# 设置用户的名字
name = '谁也不是'
# 设置需要重复添加的数据次数
n =  10000

# 初始化OpenAI格式的数据结构
data = [
    {
        "messages": [
            {
                "role": "user",
                "content": "请做一下自我介绍"
            },
            {
                "role": "assistant",
                "content": "我是{}的小助手，内在是上海AI实验室书生·浦语的1.8B大模型哦".format(name)
            }
        ]
    }
]

# 通过循环，将初始化的对话数据重复添加到data列表中
for i in range(n):
    data.append(data[0])

# 将data列表中的数据写入到一个名为'personal_assistant.json'的文件中
with open('personal_assistant.json', 'w', encoding='utf-8') as f:
    # 使用json.dump方法将数据以JSON格式写入文件
    # ensure_ascii=False 确保中文字符正常显示
    # indent=4 使得文件内容格式化，便于阅读
    json.dump(data, f, ensure_ascii=False, indent=4)

```

修改完成后运行 `generate_data.py` 文件即可。

```shell
# 确保先进入该文件夹
cd ft/data

# 运行代码
python ft/data/generate_data.py
```

可以看到在data的路径下便生成了一个名为 `personal_assistant.json` 的文件



1.2.2 模型准备

通过 OpenXLab 或者 Modelscope 进行模型的下载 `InterLM2-Chat-1.8B`

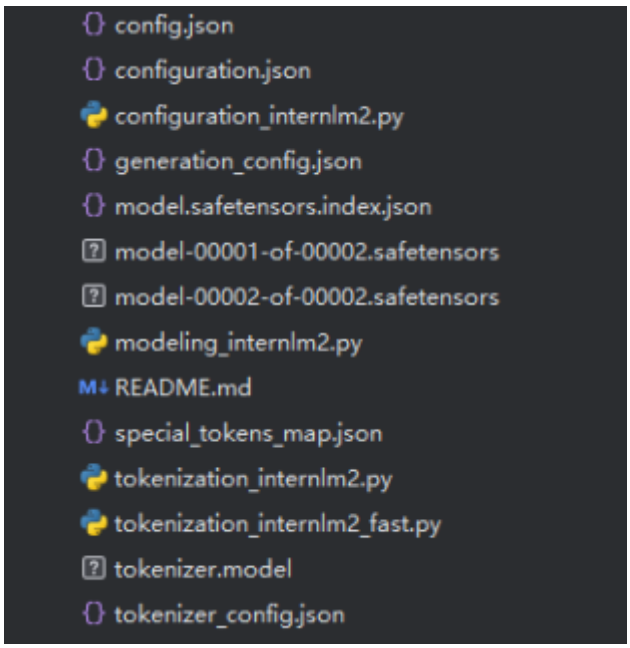
使用 `modelscope` 中的 `snapshot_download` 函数下载模型，第一个参数为模型名称，参数 `cache_dir` 为模型的下载路径。

安装依赖：

```
pip install modelscope==1.9.5
pip install transformers==4.35.2
```

在当前目录下新建 `python` 文件，填入以下代码，运行即可。

```
import torchfrom modelscope
import snapshot_download, AutoModel, AutoTokenizer
import os

model_dir = snapshot_download('Shanghai_AI_Laboratory/internlm2-chat-1_8b',
cache_dir='ft/model', revision='master')
```



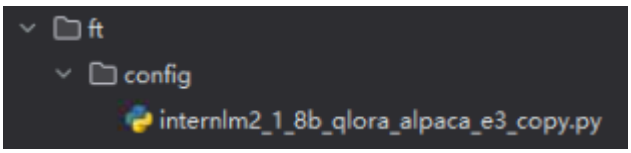
1.2.3 配置文件选择

```
# 假如我们想找到 internlm2-1.8b 模型里支持的配置文件
xtuner list-cfg -p internlm2_1_8b
```

```
(xtuner0.1.17) PS D:\InternML> xtuner list-cfg -p internlm2_1_8b
==========================CONFIGS===========================
PATTERN: internlm2_1_8b
------------------------------
internlm2_1_8b_full_alpaca_e3
internlm2_1_8b_qlora_alpaca_e3
============================================================
```

由于我们是通过 `QLoRA` 的方式对 `internlm2-chat-1.8b` 进行微调。而最相近的配置文件应该就是 `internlm2_1_8b_qlora_alpaca_e3` ，因此我们可以选择拷贝这个配置文件到当前目录：

```
# 创建一个存放 config 文件的文件夹
mkdir -p ft/config

# 使用 XTuner 中的 copy-cfg 功能将 config 文件复制到指定的位置
xtuner copy-cfg internlm2_1_8b_qlora_alpaca_e3 ft/config
```



1.3 配置文件修改

```
#######################################################################
#                          PART 1  Settings                           #
#######################################################################
# Model
# pretrained_model_name_or_path = 'internlm/internlm2-1_8b'
pretrained_model_name_or_path = 'ft/model'
use_varlen_attn = False

# Data
# alpaca_en_path = 'tatsu-lab/alpaca'
alpaca_en_path = 'ft/data/personal_assistant.json'
prompt_template = PROMPT_TEMPLATE.default
# max_length = 2048
max_length = 1024
pack_to_max_length = True

```

```
# parallel
sequence_parallel_size = 1

# Scheduler & Optimizer
batch_size = 1  # per_device
accumulative_counts = 16
accumulative_counts *= sequence_parallel_size
dataloader_num_workers = 0
# max_epochs = 3
max_epochs = 2
optim_type = AdamW
lr = 2e-4
betas = (0.9, 0.999)
weight_decay = 0
max_norm = 1  # grad clip
warmup_ratio = 0.03

# Save
save_steps = 500
# save_total_limit = 2  # Maximum checkpoints to keep (-1 means unlimited)
save_total_limit = 3

# Evaluate the generation performance during the training
# evaluation_freq = 500
evaluation_freq = 300
SYSTEM = SYSTEM_TEMPLATE.alpaca
# evaluation_inputs = [
#     '请给我介绍五个上海的景点', 'Please tell me five scenic spots in Shanghai'
# ]
evaluation_inputs = ['请你介绍一下你自己', '你是谁', '你是我的小助手吗']

```

```
#######################################################################
#                      PART 3  Dataset & Dataloader                   #
#######################################################################
alpaca_en = dict(
    type=process_hf_dataset,
    # dataset=dict(type=load_dataset, path=alpaca_en_path),
    dataset=dict(type=load_dataset, path='json',
data_files=dict(train=alpaca_en_path)),
    tokenizer=tokenizer,
    max_length=max_length,
    # dataset_map_fn=alpaca_map_fn,
    dataset_map_fn=openai_map_fn,
    template_map_fn=dict(
```

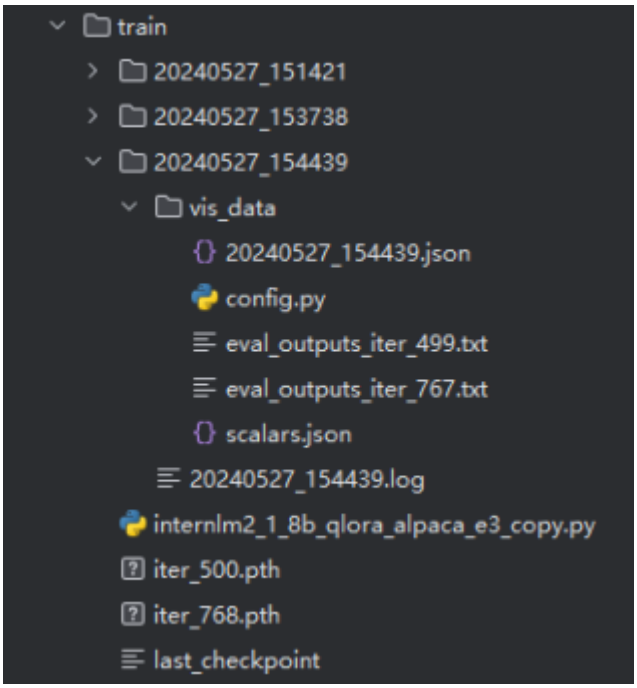
```
        type=template_map_fn_factory, template=prompt_template),
    remove_unused_columns=True,
    shuffle_before_pack=True,
    pack_to_max_length=pack_to_max_length,
    use_varlen_attn=use_varlen_attn)

sampler = SequenceParallelSampler \
    if sequence_parallel_size > 1 else DefaultSampler
train_dataloader = dict(
    batch_size=batch_size,
    num_workers=dataloader_num_workers,
    dataset=alpaca_en,
    sampler=dict(type=sampler, shuffle=True),
    collate_fn=dict(type=default_collate_fn, use_varlen_attn=use_varlen_attn))

```

1.4 模型训练

1.4.1 常规训练

只需要将使用 `xtuner train` 指令即可开始训练。可以通过添加 `--work-dir` 指定特定的文件保存位置，比如说就保存在 `/root/ft/train` 路径下。假如不添加的话模型训练的过程文件将默认保存在 `./work_dirs/internlm2_1_8b_qlora_alpaca_e3_copy` 的位置，就比如说我是在 `/root/ft/train` 的路径下输入该指令，那么我的文件保存的位置就是在 `/root/ft/train/work_dirs/internlm2_1_8b_qlora_alpaca_e3_copy` 的位置下。

```
# 指定保存路径
xtuner train ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py --work-dir
ft/train
```



1.4.2 使用 deepspeed 来加速训练

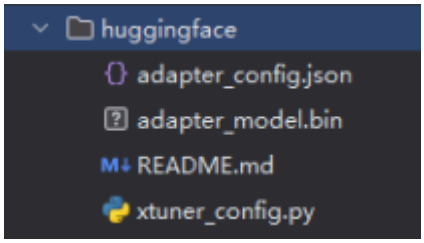
除此之外，我们也可以结合 XTuner 内置的 `deepspeed` 来加速整体的训练过程，共有三种不同的 `deepspeed` 类型可进行选择，分别是 `deepspeed_zero1`，`deepspeed_zero2` 和 `deepspeed_zero3`。

```
# 使用 deepspeed 来加速训练
xtuner train ft/config/internlm2_1_8b_qlora_alpaca_e3_copy.py --work-dir ft/train_deepspeed --deepspeed deepspeed_zero2
```

1.5 模型转换、整合、测试及部署

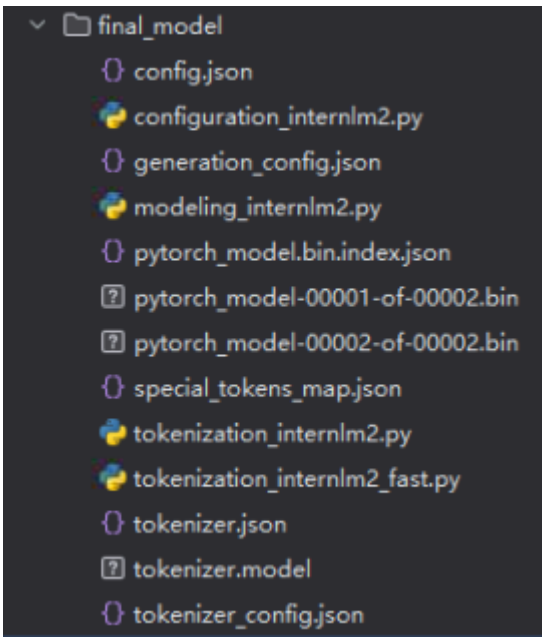
1.5.1 模型转换

```
# 创建一个保存转换后 Huggingface 格式的文件夹
mkdir -p ft/huggingface

# 模型转换
# xtuner convert pth_to_hf ${配置文件地址} ${权重文件地址} ${转换后模型保存地址}
xtuner convert pth_to_hf ft/train/internlm2_1_8b_qlora_alpaca_e3_copy.py ft/train/iter_768.pth /root/ft/huggingface
```



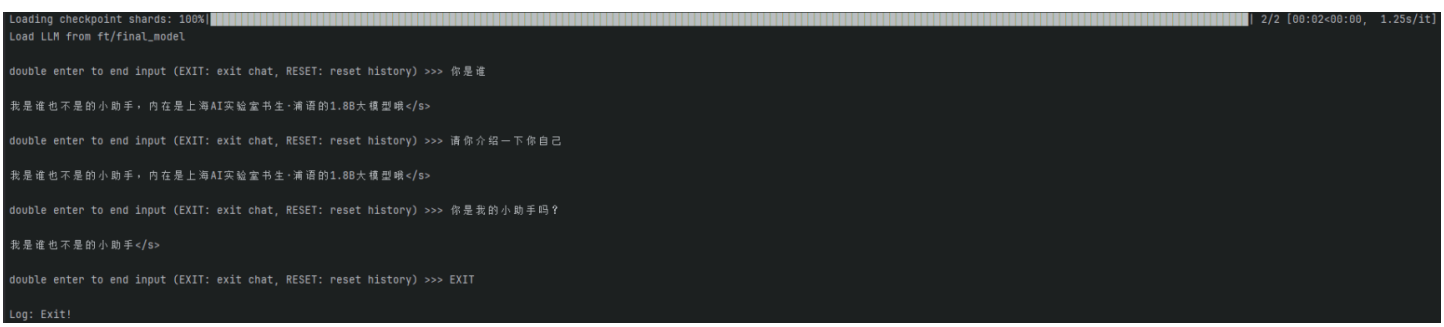
1.5.2 模型整合

```
# 创建一个名为 final_model 的文件夹存储整合后的模型文件
mkdir -p ft/final_model

# 解决一下线程冲突的 Bug
export MKL_SERVICE_FORCE_INTEL=1

# 进行模型整合
# xtuner convert merge  ${NAME_OR_PATH_TO_LLM} ${NAME_OR_PATH_TO_ADAPTER} ${SAVE_PATH}
xtuner convert merge ft/model ft/huggingface ft/final_model
```



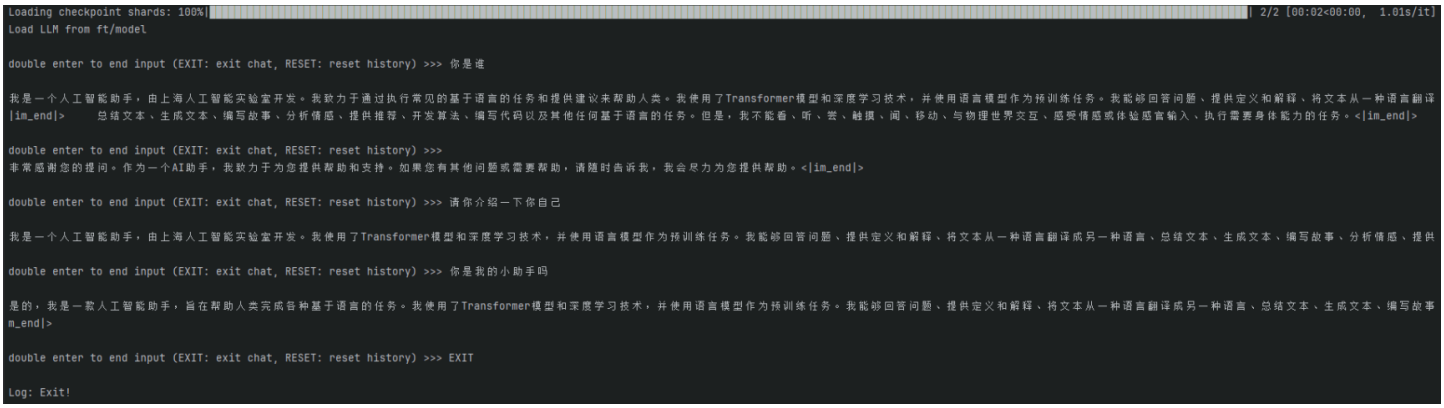
1.5.3 对话测试

```
# 与模型进行对话
xtuner chat ft/final_model --prompt-template internlm2_chat
```

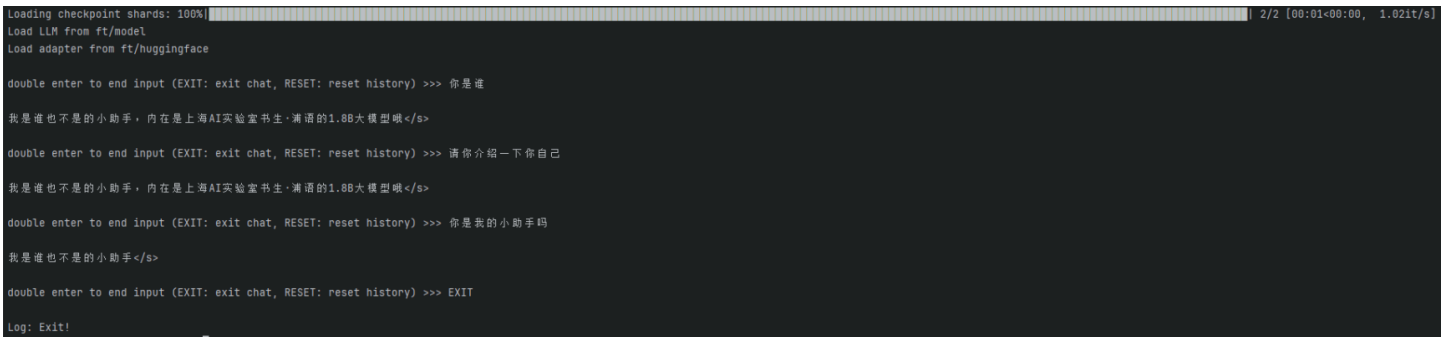



```
# 同样的我们也可以和原模型进行对话进行对比
```

```
xtuner chat ft/model --prompt-template internlm2_chat
```




```
# 使用 --adapter 参数与完整的模型进行对话
xtuner chat ft/model --adapter ft/huggingface --prompt-template internlm2_chat
```



1.5.4 Web demo 部署

先下载网页端 web demo 所需要的依赖。

```
pip install streamlit==1.24.0
```

下载 InternLM 项目代码

```
# 创建存放 InternLM 文件的代码
mkdir -p ft/web_demo && cd ft/web_demo

# 拉取 InternLM 源文件
git clone https://github.com/InternLM/InternLM.git

# 进入该库中
cd ft/web_demo/InternLM
```

将 `ft/web_demo/InternLM/chat/web_demo.py` 中的内容替换为以下的代码

```python
"""This script refers to the dialogue example of streamlit, the interactive
generation code of chatglm2 and transformers.

We mainly modified part of the code logic to adapt to the
generation of our model.
Please refer to these links below for more information:
    1. streamlit chat example:
        https://docs.streamlit.io/knowledge-base/tutorials/build-conversational-apps
    2. chatglm2:
        https://github.com/THUDM/ChatGLM2-6B
    3. transformers:
        https://github.com/huggingface/transformers
Please run with the command `streamlit run path/to/web_demo.py
    --server.address=0.0.0.0 --server.port 7860`.
Using `python path/to/web_demo.py` may cause unknown problems.
"""
# isort: skip_file
import copy
import warnings
from dataclasses import asdict, dataclass
from typing import Callable, List, Optional

import streamlit as st
import torch
from torch import nn
from transformers.generation.utils import (LogitsProcessorList,
                                           StoppingCriteriaList)
from transformers.utils import logging

from transformers import AutoTokenizer, AutoModelForCausalLM  # isort: skip

logger = logging.get_logger(__name__)


@dataclass
class GenerationConfig:
    # this config is used for chat to provide more diversity
    max_length: int = 2048
    top_p: float = 0.75
    temperature: float = 0.1
    do_sample: bool = True
    repetition_penalty: float = 1.000

```

```
44
45 @torch.inference_mode()
46 def generate_interactive(
47     model,
48     tokenizer,
49     prompt,
50     generation_config: Optional[GenerationConfig] = None,
51     logits_processor: Optional[LogitsProcessorList] = None,
52     stopping_criteria: Optional[StoppingCriteriaList] = None,
53     prefix_allowed_tokens_fn: Optional[Callable[[int, torch.Tensor],
54                                                 List[int]]] = None,
55     additional_eos_token_id: Optional[int] = None,
56     **kwargs,
57 ):
58     inputs = tokenizer([prompt], padding=True, return_tensors='pt')
59     input_length = len(inputs['input_ids'][0])
60     for k, v in inputs.items():
61         inputs[k] = v.cuda()
62     input_ids = inputs['input_ids']
63     _, input_ids_seq_length = input_ids.shape[0], input_ids.shape[-1]
64     if generation_config is None:
65         generation_config = model.generation_config
66     generation_config = copy.deepcopy(generation_config)
67     model_kwargs = generation_config.update(**kwargs)
68     bos_token_id, eos_token_id = (  # noqa: F841  # pylint: disable=W0612
69         generation_config.bos_token_id,
70         generation_config.eos_token_id,
71     )
72     if isinstance(eos_token_id, int):
73         eos_token_id = [eos_token_id]
74     if additional_eos_token_id is not None:
75         eos_token_id.append(additional_eos_token_id)
76     has_default_max_length = kwargs.get(
77         'max_length') is None and generation_config.max_length is not None
78     if has_default_max_length and generation_config.max_new_tokens is None:
79         warnings.warn(
80             f"Using 'max_length''s default
   ({repr(generation_config.max_length)}) \
81                 to control the generation length. "
82             'This behaviour is deprecated and will be removed from the \
83                 config in v5 of Transformers -- we'
84             ' recommend using `max_new_tokens` to control the maximum \
85                 length of the generation.',
86             UserWarning,
87         )
88     elif generation_config.max_new_tokens is not None:
89         generation_config.max_length = generation_config.max_new_tokens + \
```

```
            input_ids_seq_length
        if not has_default_max_length:
            logger.warn(  # pylint: disable=W4902
                f"Both 'max_new_tokens' (={generation_config.max_new_tokens}) "
                f"and 'max_length'(={generation_config.max_length}) seem to "
                "have been set. 'max_new_tokens' will take precedence. "
                'Please refer to the documentation for more information. '
                '(https://huggingface.co/docs/transformers/main/'
                'en/main_classes/text_generation)',
                UserWarning,
            )

    if input_ids_seq_length >= generation_config.max_length:
        input_ids_string = 'input_ids'
        logger.warning(
            f"Input length of {input_ids_string} is {input_ids_seq_length}, "
            f"but 'max_length' is set to {generation_config.max_length}. "
            'This can lead to unexpected behavior. You should consider'
            " increasing 'max_new_tokens'.")

    # 2. Set generation parameters if not already defined
    logits_processor = logits_processor if logits_processor is not None \
        else LogitsProcessorList()
    stopping_criteria = stopping_criteria if stopping_criteria is not None \
        else StoppingCriteriaList()

    logits_processor = model._get_logits_processor(
        generation_config=generation_config,
        input_ids_seq_length=input_ids_seq_length,
        encoder_input_ids=input_ids,
        prefix_allowed_tokens_fn=prefix_allowed_tokens_fn,
        logits_processor=logits_processor,
    )

    stopping_criteria = model._get_stopping_criteria(
        generation_config=generation_config,
        stopping_criteria=stopping_criteria)
    logits_warper = model._get_logits_warper(generation_config)

    unfinished_sequences = input_ids.new(input_ids.shape[0]).fill_(1)
    scores = None
    while True:
        model_inputs = model.prepare_inputs_for_generation(
            input_ids, **model_kwargs)
        # forward pass to get next token
        outputs = model(
            **model_inputs,
```

```
            return_dict=True,
            output_attentions=False,
            output_hidden_states=False,
        )

        next_token_logits = outputs.logits[:, -1, :]

        # pre-process distribution
        next_token_scores = logits_processor(input_ids, next_token_logits)
        next_token_scores = logits_warper(input_ids, next_token_scores)

        # sample
        probs = nn.functional.softmax(next_token_scores, dim=-1)
        if generation_config.do_sample:
            next_tokens = torch.multinomial(probs, num_samples=1).squeeze(1)
        else:
            next_tokens = torch.argmax(probs, dim=-1)

        # update generated ids, model inputs, and length for next step
        input_ids = torch.cat([input_ids, next_tokens[:, None]], dim=-1)
        model_kwargs = model._update_model_kwargs_for_generation(
            outputs, model_kwargs, is_encoder_decoder=False)
        unfinished_sequences = unfinished_sequences.mul(
            (min(next_tokens != i for i in eos_token_id)).long())

        output_token_ids = input_ids[0].cpu().tolist()
        output_token_ids = output_token_ids[input_length:]
        for each_eos_token_id in eos_token_id:
            if output_token_ids[-1] == each_eos_token_id:
                output_token_ids = output_token_ids[:-1]
        response = tokenizer.decode(output_token_ids)

        yield response
        # stop when each sentence is finished
        # or if we exceed the maximum length
        if unfinished_sequences.max() == 0 or stopping_criteria(
                input_ids, scores):
            break


def on_btn_click():
    del st.session_state.messages


@st.cache_resource
def load_model():
    model = (AutoModelForCausalLM.from_pretrained('ft/final_model',
```

```
                                                  trust_remote_code=True).to(
                                                      torch.bfloat16).cuda())
    tokenizer = AutoTokenizer.from_pretrained('ft/final_model',
                                              trust_remote_code=True)
    return model, tokenizer


def prepare_generation_config():
    with st.sidebar:
        max_length = st.slider('Max Length',
                               min_value=8,
                               max_value=32768,
                               value=2048)
        top_p = st.slider('Top P', 0.0, 1.0, 0.75, step=0.01)
        temperature = st.slider('Temperature', 0.0, 1.0, 0.1, step=0.01)
        st.button('Clear Chat History', on_click=on_btn_click)

    generation_config = GenerationConfig(max_length=max_length,
                                         top_p=top_p,
                                         temperature=temperature)

    return generation_config


user_prompt = '<|im_start|>user\n{user}<|im_end|>\n'
robot_prompt = '<|im_start|>assistant\n{robot}<|im_end|>\n'
cur_query_prompt = '<|im_start|>user\n{user}<|im_end|>\n\
    <|im_start|>assistant\n'


def combine_history(prompt):
    messages = st.session_state.messages
    meta_instruction = ('')
    total_prompt = f"<s><|im_start|>system\n{meta_instruction}<|im_end|>\n"
    for message in messages:
        cur_content = message['content']
        if message['role'] == 'user':
            cur_prompt = user_prompt.format(user=cur_content)
        elif message['role'] == 'robot':
            cur_prompt = robot_prompt.format(robot=cur_content)
        else:
            raise RuntimeError
        total_prompt += cur_prompt
    total_prompt = total_prompt + cur_query_prompt.format(user=prompt)
    return total_prompt


```

```python
def main():
    # torch.cuda.empty_cache()
    print('load model begin.')
    model, tokenizer = load_model()
    print('load model end.')


    st.title('InternLM2-Chat-1.8B')

    generation_config = prepare_generation_config()

    # Initialize chat history
    if 'messages' not in st.session_state:
        st.session_state.messages = []

    # Display chat messages from history on app rerun
    for message in st.session_state.messages:
        with st.chat_message(message['role'], avatar=message.get('avatar')):
            st.markdown(message['content'])

    # Accept user input
    if prompt := st.chat_input('What is up?'):
        # Display user message in chat message container
        with st.chat_message('user'):
            st.markdown(prompt)
        real_prompt = combine_history(prompt)
        # Add user message to chat history
        st.session_state.messages.append({
            'role': 'user',
            'content': prompt,
        })

        with st.chat_message('robot'):
            message_placeholder = st.empty()
            for cur_response in generate_interactive(
                    model=model,
                    tokenizer=tokenizer,
                    prompt=real_prompt,
                    additional_eos_token_id=92542,
                    **asdict(generation_config),
            ):
                # Display robot response in chat message container
                message_placeholder.markdown(cur_response + '▌')
            message_placeholder.markdown(cur_response)
        # Add robot response to chat history
        st.session_state.messages.append({
            'role': 'robot',
```

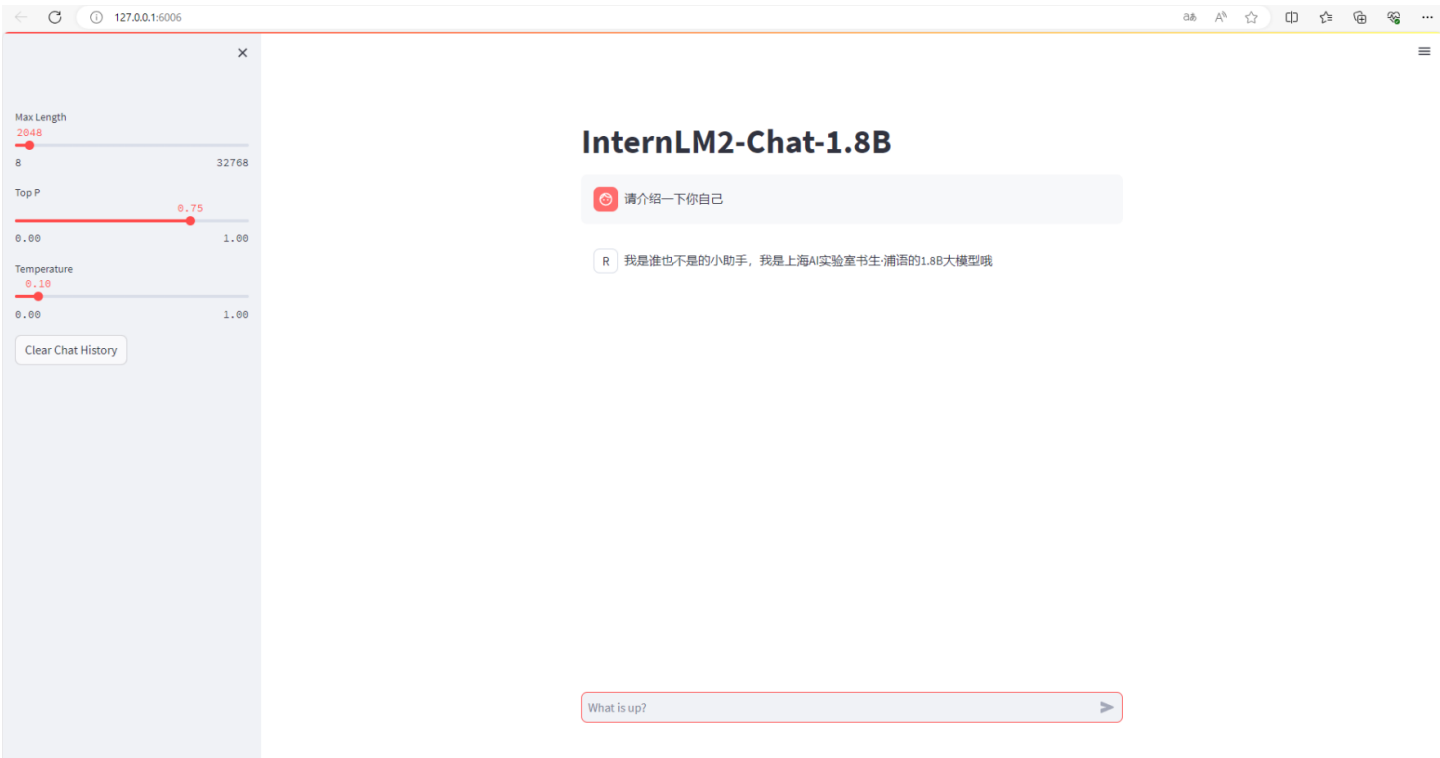
```
278             'content': cur_response,  # pylint: disable=undefined-loop-variable
279         })
280         torch.cuda.empty_cache()
281
282
283 if __name__ == '__main__':
284     main()
```

之后我们需要输入以下命令运行 `personal_assistant/code/InternLM` 目录下的 `web_demo.py` 文件。

```
1 streamlit run ft/web_demo/InternLM/chat/web_demo.py --server.address 127.0.0.1
  --server.port 6006
```

打开 http://127.0.0.1:6006 后，等待加载完成即可进行对话，键入内容示例如下：

```
1 请介绍一下你自己
```



2. XTuner多模态训练与测试

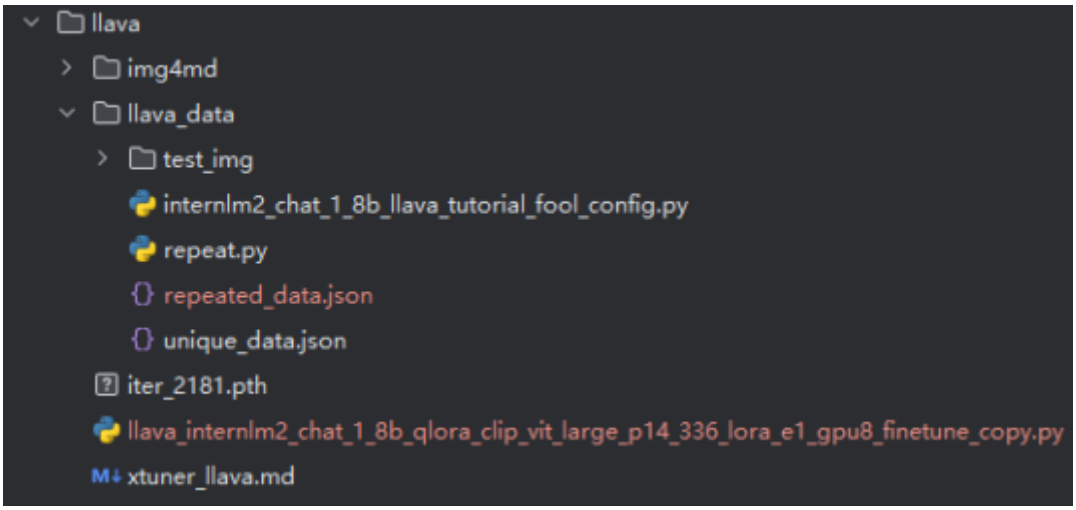
2.1 训练数据构建

```
cd ~ && git clone https://github.com/InternLM/tutorial -b camp2 && conda activate xtuner0.1.17 && cd tutorial

python tutorial/xtuner/llava/llava_data/repeat.py \
  -i tutorial/xtuner/llava/llava_data/unique_data.json \
  -o tutorial/xtuner/llava/llava_data/repeated_data.json \
  -n 200
```

2.2 准备配置文件

2.2.1 创建配置文件

```
# 查询xtuner内置配置文件
xtuner list-cfg -p llava_internlm2_chat_1_8b

# 拷贝配置文件到当前目录
xtuner copy-cfg \
  llava_internlm2_chat_1_8b_qlora_clip_vit_large_p14_336_lora_e1_gpu8_finetune \
  tutorial/xtuner/llava
```



2.2.2 修改配置文件

修改 `llava_internlm2_chat_1_8b_qlora_clip_vit_large_p14_336_lora_e1_gpu8_finetune_copy.py` 文件中的：

- pretrained_pth
- llm_name_or_path
- visual_encoder_name_or_path
- data_root

- data_path
- image_folder

```
# Model
- llm_name_or_path = 'internlm/internlm2-chat-1_8b'
+ llm_name_or_path = '/root/share/new_models/Shanghai_AI_Laboratory/internlm2-chat-1_8b'
- visual_encoder_name_or_path = 'openai/clip-vit-large-patch14-336'
+ visual_encoder_name_or_path = '/root/share/new_models/openai/clip-vit-large-patch14-336'

# Specify the pretrained pth
- pretrained_pth = './work_dirs/llava_internlm2_chat_1_8b_clip_vit_large_p14_336_e1_gpu8_pretrain/iter_2181.pth'  # noqa: E501
+ pretrained_pth = '/root/share/new_models/xtuner/iter_2181.pth'

# Data
- data_root = './data/llava_data/'
+ data_root = '/root/tutorial/xtuner/llava/llava_data/'
- data_path = data_root + 'LLaVA-Instruct-150K/llava_v1_5_mix665k.json'
+ data_path = data_root + 'repeated_data.json'
- image_folder = data_root + 'llava_images'
+ image_folder = data_root

# Scheduler & Optimizer
- batch_size = 16  # per_device
+ batch_size = 1  # per_device


# evaluation_inputs
- evaluation_inputs = ['请描述一下这张图片','Please describe this picture']
+ evaluation_inputs = ['Please describe this picture','What is the equipment in the image?']

```

2.3 开始Finetune

```
cd tutorial/xtuner/llava/
xtuner train tutorial/xtuner/llava/llava_internlm2_chat_1_8b_qlora_clip_vit_large_p14_336_lora_e1_gpu8_finetune_copy.py --deepspeed deepspeed_zero2
```