清华大学
高级机器学习
2024 年秋季学期

小作业-LLM 的微调与 RAG

杨天行 2024310659 2025 年 1 月 5 日

1 实验任务

本次实验的任务围绕对大规模预训练语言模型（LLM）的微调以及检索增强生成（Retrieval-Augmented Generation, RAG）技术展开，主要包括以下几个方面：

知识库构建。收集与机器学习相关的领域知识，构建一个知识库。知识库应具有一定的专业性，包含一些无法通过预训练语言模型直接回答的问题，以增强其实际应用价值。
检索实现。实现从知识库中检索相关信息的功能。检索系统可使用开源库或训练好的模型，确保问题能够与知识库中的内容进行高效匹配。
问答数据构建。基于知识库内容，利用现有的 LLM API（如 ChatGPT LLM、ChatGLM）生成问答对，构建训练数据。问题应具有一定的专业性，确保 LLM 无法依靠其预训练知识直接回答。答案应结合检索到的 Top-K 知识点生成，答案中需包含引用标记（如 [1], [2]）。
模型微调。使用上一次实验的代码加载 ChatGLM3-6B 权重，对模型进行微调，模型结构与 GLM4-9B 相同。微调过程中需实现相关训练代码，可参考或使用开源工具库（如 Transformers）。设置合理的训练参数，并记录训练过程中的相关信息（如 Loss 曲线）。
性能评估。通过实验对比基于检索的问答生成（RAG）与直接使用预训练语言模型进行问答的性能差异。评估指标包括生成答案的准确性、专业性，以及引用标记的完整性。

2 微调数据构建

在本次实验中我们使用 baseline 代码提供的 ICNP'24 论文 "Luori: Active Probing and Evaluation of Internet-wide IPv6 Fully Responsive Prefixes" 作为知识库数据。我们的目标是微调 ChatGLM3-6B，并使其输出包含引用标记。为此我们需要先生成对应于知识库内容的问题。构建 "问题-知识库信息-答案" 的 pipeline，其中 "问题 + 知识库数据" 作为微调数据的输入，"答案" 作为微调数据的输出。这样在 LLM 输出的答案中，能带上来自知识库信息的引用标记。

2.1 问题生成

首先调用 DeepSeek API，输入知识库与对应的 prompt，以 json 格式批量输出问题：

```
question_prefix = """
Here is a text. Please generate 50 questions based on the information in the text. The
    questions should be answerable using the information from the text. Only output the
    questions.

EXAMPLE JSON OUTPUT:
{
   "questions":[
```

```
        "the first question",
        "the second question",
        "the third question"
    ]
}

"""

def generate_question(client, text):
    response = client.chat.completions.create(
        model="deepseek-chat",
        messages=[
            {"role": "system", "content": "You are a helpful assistant"},
            {"role": "user", "content": question_prefix + text},
        ],
        response_format={
            'type': 'json_object'
        },
        stream=False,
        temperature=0.8,
        max_tokens=8192
    )

    return response.choices[0].message.content
```

一共生成问题 391 条（输入相同，重复生成问题可以出发 input cache hit），生成的问题样例如下：

- What is the significance of IPv6 Fully Responsive Prefixes (FRPs) in network measurements?
- How does Luori propose to address the challenge of probing IPv6 FRPs?
- What is the core idea behind Luori's active probing method?
- How does Luori transform the active probing process into a dynamic search process?
- What are the evaluation results of Luori's Internet-wide probing?

2.2 RAG Prompt 修改

本次实验在提供的 baseline 代码使用的 LightRAG[1]上修改 RAG 代码。直接运行`run.py`，LightRAG 能够自动在文本上构建 Knowledge Graph 并利用 KG 构建 prompt，从而回答问题，但答案中并不包含引用标记。

我们以上面的问题之一 "What are the evaluation results of Luori's Internet-wide probing" 进行测试，结果如下：

[1]Zirui Guo et al., LightRAG: Simple and Fast Retrieval-Augmented Generation, preprint

```
The evaluation results of Luori's Internet-wide probing demonstrate its
↪ effectiveness and efficiency in detecting and compiling a comprehensive list
↪ of Fully Responsive Prefixes (FRPs) in IPv6 networks. Here's a breakdown of
↪ the key findings:

**Coverage and Scope**:
- **Address Space**: Luori successfully probed a vast address space, covering 1.3
↪ × 10^33 addresses. This is equivalent to the address space covered by 7.1 ×
↪ 10^13 /64 prefixes.
- **Autonomous Systems (ASes)**: The FRPlist compiled by Luori includes
↪ information on 795 ASes, indicating its wide coverage across different network
↪ operators and service providers.
- **Routing Prefixes**: The list covers 9182 routing prefixes, providing a
↪ detailed view of the FRPs present in the IPv6 network.

**Performance Metrics**:
- **Number of Fully Responsive Prefixes (FRPs)**: Luori identified 516,739 FRPs
↪ during the six-month probing period, which is significantly more than the
↪ existing FRPlist.
- **False Routing Responses (FRRPs)**: The FRPlist contains 7,079 FRRPs, which are
↪ responses from network devices that do not correspond to actual network
↪ traffic. These FRRPs are crucial for understanding the impact of FRPs on
↪ network measurements.
- **Fully Responsive Non-Routing Prefixes (FRNRPs)**: The FRPlist also includes
↪ 508,660 FRNRPs, which are responsive prefixes that do not correspond to any
↪ routing prefix.

**Comparison with Existing Methods**:
- **Comparison with MAPD**: Luori outperforms the brute-force probing method MAPD
↪ in terms of coverage and efficiency. While MAPD can only probe fixed-length
↪ FRPs, Luori can probe FRPs of any length, leading to a more comprehensive list
↪ of FRPs.
- **Comparison with Hitlist's FRPlist**: Luori's FRPlist covers significantly more
↪ address space than the Hitlist's FRPlist, despite having a smaller number of
↪ FRPs. This is due to the ability of Luori to probe larger FRPs and the
↪ efficient use of probing resources.

**Long-Term Probing**:
- **Continuous Growth**: The number of FRRPs and FRNRPs in the FRPlist continues
↪ to grow over time, indicating the dynamic nature of FRP patterns in the IPv6
↪ network.
```

```
- **Impact on IPv6 Measurement Campaigns**: The growing number of FRPs highlights
↪  the potential impact of FRPs on IPv6 measurement campaigns, emphasizing the
↪  need for accurate and comprehensive FRP detection.

**Overall, the evaluation results of Luori's Internet-wide probing demonstrate its
↪  ability to effectively detect and compile a comprehensive list of FRPs in IPv6
↪  networks. The resulting FRPlist can be used by researchers and network
↪  operators to mitigate the impact of FRPs on their measurements and to better
↪  understand the characteristics and distribution of FRPs in the IPv6
↪  Internet**.
```

我们可以在 lightrag 框架代码的`lightrag/prompt.py`中定义了进行 rag 的 prompt，原始的 prompt 为：

```
PROMPTS["rag_response"] = """---Role---

You are a helpful assistant responding to questions about data in the tables
↪  provided.


---Goal---

Generate a response of the target length and format that responds to the user's
↪  question, summarizing all information in the input data tables appropriate for
↪  the response length and format, and incorporating any relevant general
↪  knowledge.
If you don't know the answer, just say so. Do not make anything up.
Do not include information where the supporting evidence for it is not provided.

---Target response length and format---

{response_type}

---Data tables---

{context_data}

Add sections and commentary to the response as appropriate for the length and
↪  format. Style the response in markdown.
"""
```

我们对其进行修改，要求先列出所有需要被引用的段落，再带上引用记号回答问题，修改后的 prompt 如

下：

```
PROMPTS["rag_response"] = """---Role---

You are a helpful assistant responding to questions about data in the tables
↪ provided.

---Goal---

Generate a response of the target length and format that responds to the user's
↪ question, summarizing all information in the input data tables appropriate for
↪ the response length and format, and incorporating any relevant general
↪ knowledge.
If you don't know the answer, just say so. Do not make anything up.
Do not include information where the supporting evidence for it is not provided.

---Target response length and format---

{response_type}

---Data tables---

{context_data}

---Instructions---

1. Before generating the main response, list all the referenced content you will
↪ use in your response. Format each reference as follows:
   - [1]: "Referenced content 1 from the data tables."
   - [2]: "Referenced content 2 from the data tables."
   - ...
   Ensure the referenced content comes directly from the provided data tables.

2. After listing all references, generate the response. Use markdown formatting
↪ and include citation markers (e.g., [1], [2]) inline, corresponding to the
↪ previously listed references. The Reference entry should only use a citation
↪ marker at the beginning, and NO UNREALTED CITATION MARKERS SHOULD APPEAR!
↪ Response can only use the citation markers appeared in References.

---Output Example---

### References
```

```
- [1]: Example content from table A.
- [2]: Another example content from table B.

### Response
Based on the data, the trends show an increase in revenue [1] and a decrease in
↪ expenses [2].
"""
```

对上面相同的问题，现在得到了如下，表明可以正确引用内容并带上引用记号：

```
### References
- [1]: "We have implemented a prototype of Luori and deployed it from November
↪ 2023 to April 2024. In this section, we evaluate the performance of Luori
↪ during its deployment." from the Sources table.
- [2]: "Luori found 31.7K FRPs1 in a single Internet-wide probing with 11M
↪ budget2, covering 1.5×10^30 address space3, which is 106 × that of brute-force
↪ probing." from the Sources table.
- [3]: "As of now, Luori has been continuously probing Internet-wide for almost
↪ six months and has found 516k FRPs, which covers 1.3×10^33 address space and
↪ 795 ASes, far exceeding the existing FRPlist, making it the largest publicly
↪ known FRPlist." from the Sources table.

### Response
The evaluation results of Luori's Internet-wide probing demonstrate its
↪ effectiveness and scalability. In a single Internet-wide probing with a budget
↪ of 11 million, Luori found 31.7K Fully Responsive Prefixes (FRPs) and covered
↪ 1.5×10^30 address space, which is 106 times more than brute-force probing [2].
↪ Over a period of almost six months of continuous probing, Luori identified
↪ 516k FRPs, covering 1.3×10^33 address space and 795 Autonomous Systems (ASes),
↪ making it the largest publicly known FRPlist [3].
```

2.3 微调数据构造

我们可以写一个如下的脚本，使用上面修改后的 rag 进行回答，问题库是上面构造的 391 条问题。脚本如下，将 QA 对保存为 OpenAI SFT 格式的 json 数据：

```python
import os
import json
import logging
from lightrag import LightRAG, QueryParam
from lightrag.llm import zhipu_complete, zhipu_embedding
from lightrag.utils import EmbeddingFunc
from tqdm import tqdm

# 设置工作目录和日志
```

```
WORKING_DIR = "./working_dir"
DATA_OUTPUT = "./data/fine_tuning_data.json"
QUESTION_FILE = "./data/questions.json"

logging.basicConfig(format="%(levelname)s:%(message)s", level=logging.INFO)

if not os.path.exists(WORKING_DIR):
    os.mkdir(WORKING_DIR)

if not os.path.exists("./output"):
    os.mkdir("./output")

# 初始化LightRAG
rag = LightRAG(
    working_dir=WORKING_DIR,
    llm_model_func=zhipu_complete,
    llm_model_name="glm-4-flash",
    llm_model_max_async=4,
    llm_model_max_token_size=32768,
    embedding_func=EmbeddingFunc(
        embedding_dim=2048,
        max_token_size=8192,
        func=lambda texts: zhipu_embedding(texts),
    ),
)

# 加载文本数据到RAG
with open("./data/book.txt", "r", encoding="utf-8") as f:
    rag.insert(f.read())

# 加载问题库
with open(QUESTION_FILE, "r", encoding="utf-8") as f:
    questions = json.load(f)["questions"]

# 收集对话数据
fine_tuning_data = []

for question in tqdm(questions):
    try:
        logging.info(f"Processing question: {question}")
        # Hybrid search
        response = rag.query(question, param=QueryParam(mode="hybrid"))

        # 构造单轮对话数据
        dialog = {
            "messages": [
                {"role": "system", "content": "You are a helpful assistant."},
                {"role": "user", "content": question},
```

```
                {"role": "assistant", "content": response.strip()}
            ]
        }
        fine_tuning_data.append(dialog)
    except Exception as e:
        logging.error(f"Error processing question '{question}': {e}")

# 保存结果为JSON文件
with open(DATA_OUTPUT, "w", encoding="utf-8") as f:
    json.dump(fine_tuning_data, f, ensure_ascii=False, indent=4)

logging.info(f"Fine-tuning data saved to {DATA_OUTPUT}")
```

我们一共可以构造出构造出 391 条 SFT 数据，用时约 50 分钟。下面是构造出的微调数据示例：


```
[
    {
        "messages": [
            {
                "role": "system",
                "content": "You are a helpful assistant."
            },
            {
                "role": "user",
                "content": "What is the significance of IPv6 Fully Responsive
                ↪  Prefixes (FRPs) in network measurements?"
            },
            {
                "role": "assistant",
```

```
"content": "### References\n- [1]: \"FRPs are a category of issues
↪ within IPv6 routing prefixes that the system is designed to
↪ detect and address. FRPs are a category of network protocols
↪ that are implemented in various ways and are identified
↪ through probing methods like Luori. FRPs refer to Forwarding
↪ and Remote Procedure Calls, which are used to map a local IP
↪ address and port to a remote IP address and port. FRPs refer
↪ to the Free Range Prefixes that are used in various network
↪ scenarios, including load balancing and proxy services.\"
↪ [1]\n- [2]: \"FRPs are a category of protocols that are part
↪ of the IPv6 network, addressing the challenges of sparse
↪ distribution and diverse patterns.\" [3]\n- [3]: \"The
↪ presence of FRPs can significantly impact IPv6 Measurement
↪ Campaigns, as they represent responsive prefixes in the IPv6
↪ address space.\" [10]\n- [4]: \"FRPs are associated with
↪ ICMPv6 for probing IPv6 addresses.\" [5]\n- [5]: \"The FRPlist
↪ includes information about IPv6, showing a relationship
↪ between the list and the technology.\" [15]\n- [6]: \"The
↪ FRPlist covers a significant portion of the address space,
↪ demonstrating its comprehensive coverage.\" [21]\n\n###
↪ Response\nIPv6 Fully Responsive Prefixes (FRPs) are
↪ significant in network measurements due to several factors.
↪ Firstly, FRPs are a category of network protocols implemented
↪ in various ways, which are essential for addressing challenges
↪ such as sparse distribution and diverse patterns within the
↪ IPv6 network [1][2]. These prefixes represent responsive
↪ prefixes in the IPv6 address space [3], which is crucial for
↪ accurate network measurements.\n\nThe presence of FRPs can
↪ significantly impact IPv6 Measurement Campaigns as they are
↪ responsive prefixes that need to be identified and addressed
↪ [3]. They are often associated with ICMPv6 for probing IPv6
↪ addresses, which is a protocol used for sending error messages
↪ and operational information in an IPv6 network [4][5]. The
↪ FRPlist, which includes information about IPv6, demonstrates
↪ the relationship between the list and the technology,
↪ indicating the importance of FRPs in the context of IPv6
↪ network measurements [5][6]. Overall, FRPs are significant in
↪ network measurements due to their role in responsive prefix
↪ detection and the challenges they pose in accurately measuring
↪ IPv6 network performance."
}
```

```
            ]
        }
    ]
```

3 LLaMA-3-8B-instruct 微调

3.1 Xtuner 配置

我们使用 LLaMA-3-8B-instruct 作为微调的基座模型，使用 Xtuner 作为微调框架。按照 Xtuner 的文档，使用余弦退火学习率，其中关键的参数包括：

- batch_size=1
- n_procs=8
- max_epochs=50
- lr=2e-5
- max_length=4096

我们可以写出如下的参数配置文档，保存为xtuner_config.py:

```python
# Copyright (c) OpenMMLab. All rights reserved.
from datasets import load_dataset
from mmengine.dataset import DefaultSampler
from mmengine.hooks import (CheckpointHook, DistSamplerSeedHook, IterTimerHook,
                            LoggerHook, ParamSchedulerHook)
from mmengine.optim import AmpOptimWrapper, CosineAnnealingLR, LinearLR
from torch.optim import AdamW
from transformers import AutoModelForCausalLM, AutoTokenizer
from mmengine.visualization import Visualizer, TensorboardVisBackend

from xtuner.dataset.map_fns import template_map_fn_factory, openai_map_fn
from xtuner.dataset import process_hf_dataset
from xtuner.dataset.collate_fns import default_collate_fn
from xtuner.engine.hooks import (DatasetInfoHook, EvaluateChatHook,
                                 VarlenAttnArgsToMessageHubHook)
from xtuner.engine.runner import TrainLoop
from xtuner.model import SupervisedFinetune
from xtuner.parallel.sequence import SequenceParallelSampler
from xtuner.utils import PROMPT_TEMPLATE, SYSTEM_TEMPLATE

#######################################################################
#                          PART 1  Settings                           #
#######################################################################
# Model
pretrained_model_name_or_path = 'meta-llama/Meta-Llama-3-8B-Instruct'
use_varlen_attn = False
```

```
27
28 # Data
29 prompt_template = PROMPT_TEMPLATE.llama3_chat
30 max_length = 4096
31 pack_to_max_length = True
32
33 # parallel
34 sequence_parallel_size = 1
35
36 # Scheduler & Optimizer
37 batch_size = 1  # per_device
38 accumulative_counts = 16
39 accumulative_counts *= sequence_parallel_size
40 dataloader_num_workers = 0
41 max_epochs = 50
42 optim_type = AdamW
43 lr = 2e-5
44 betas = (0.9, 0.999)
45 weight_decay = 0
46 max_norm = 1  # grad clip
47 warmup_ratio = 0.03
48
49 # Save
50 save_steps = 500
51 save_total_limit = 2  # Maximum checkpoints to keep (-1 means unlimited)
52
53 # Evaluate the generation performance during the training
54 evaluation_freq = 500
55 SYSTEM = "You are a helpful assistant."
56 evaluation_inputs = [
57     'What is FRP?'
58 ]
59
60 #######################################################################
61 #                      PART 2  Model & Tokenizer                      #
62 #######################################################################
63 tokenizer = dict(
64     type=AutoTokenizer.from_pretrained,
65     pretrained_model_name_or_path=pretrained_model_name_or_path,
66     trust_remote_code=True,
67     padding_side='right')
68
69 model = dict(
70     type=SupervisedFinetune,
71     use_varlen_attn=use_varlen_attn,
72     llm=dict(
73         type=AutoModelForCausalLM.from_pretrained,
74         pretrained_model_name_or_path=pretrained_model_name_or_path,
```

```
        trust_remote_code=True))

#######################################################################
#                      PART 3  Dataset & Dataloader                   #
#######################################################################
data_files = "data/fine_tuning_data.json"

sampler = SequenceParallelSampler \
    if sequence_parallel_size > 1 else DefaultSampler

train_dataset = dict(
    type=process_hf_dataset,
    dataset=dict(type=load_dataset, path='json', data_files=data_files),
    # dataset=dict(type=load_dataset, path='json', data_files='tmp/dataset_dict.json'),
    tokenizer=tokenizer,
    max_length=max_length,
    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)

train_dataloader = dict(
    batch_size=batch_size,
    num_workers=dataloader_num_workers,
    dataset=train_dataset,
    sampler=dict(type=sampler, shuffle=True),
    collate_fn=dict(type=default_collate_fn, use_varlen_attn=use_varlen_attn))



#######################################################################
#                    PART 4  Scheduler & Optimizer                    #
#######################################################################
# optimizer
optim_wrapper = dict(
    type=AmpOptimWrapper,
    optimizer=dict(
        type=optim_type, lr=lr, betas=betas, weight_decay=weight_decay),
    clip_grad=dict(max_norm=max_norm, error_if_nonfinite=False),
    accumulative_counts=accumulative_counts,
    loss_scale='dynamic',
    dtype='float16')

param_scheduler = [
    dict(
```

```
        type=CosineAnnealingLR,
        eta_min=0.0,
        by_epoch=True,
        begin=0,
        end=max_epochs,
        convert_to_iter_based=True)
]

# train, val, test setting
train_cfg = dict(type=TrainLoop, max_epochs=max_epochs)

#######################################################################
#                          PART 5  Runtime                            #
#######################################################################
# Log the dialogue periodically during the training process, optional
custom_hooks = [
    dict(type=DatasetInfoHook, tokenizer=tokenizer),
    dict(
        type=EvaluateChatHook,
        tokenizer=tokenizer,
        every_n_iters=evaluation_freq,
        evaluation_inputs=evaluation_inputs,
        system=SYSTEM,
        prompt_template=prompt_template)
]

if use_varlen_attn:
    custom_hooks += [dict(type=VarlenAttnArgsToMessageHubHook)]

# configure default hooks
default_hooks = dict(
    # record the time of every iteration.
    timer=dict(type=IterTimerHook),
    # print log every 10 iterations.
    logger=dict(type=LoggerHook, log_metric_by_epoch=False, interval=10),
    # enable the parameter scheduler.
    param_scheduler=dict(type=ParamSchedulerHook),
    # save checkpoint per `save_steps`.
    checkpoint=dict(
        type=CheckpointHook,
        by_epoch=False,
        interval=save_steps,
        max_keep_ckpts=save_total_limit),
    # set sampler seed in distributed evrionment.
    sampler_seed=dict(type=DistSamplerSeedHook),
)

# configure environment
```

```python
env_cfg = dict(
    # whether to enable cudnn benchmark
    cudnn_benchmark=False,
    # set multi process parameters
    mp_cfg=dict(mp_start_method='fork', opencv_num_threads=0),
    # set distributed parameters
    dist_cfg=dict(backend='nccl'),
)

# set visualizer
visualizer = dict(type=Visualizer, vis_backends=[dict(type=TensorboardVisBackend)])

# set log level
log_level = 'INFO'

# load from which checkpoint
load_from = None

# whether to resume training from the loaded checkpoint
resume = False

# Defaults to use random seed and disable `deterministic`
randomness = dict(seed=None, deterministic=False)

# set log processor
log_processor = dict(by_epoch=False)
```

微调命令如下：

```
NPROC_PER_NODE=8 xtuner train ./xtuner_config.py  --deepspeed deepspeed_zero1
```

使用 8 卡 H100，用时约 4 分钟，微调 loss 曲线如下：

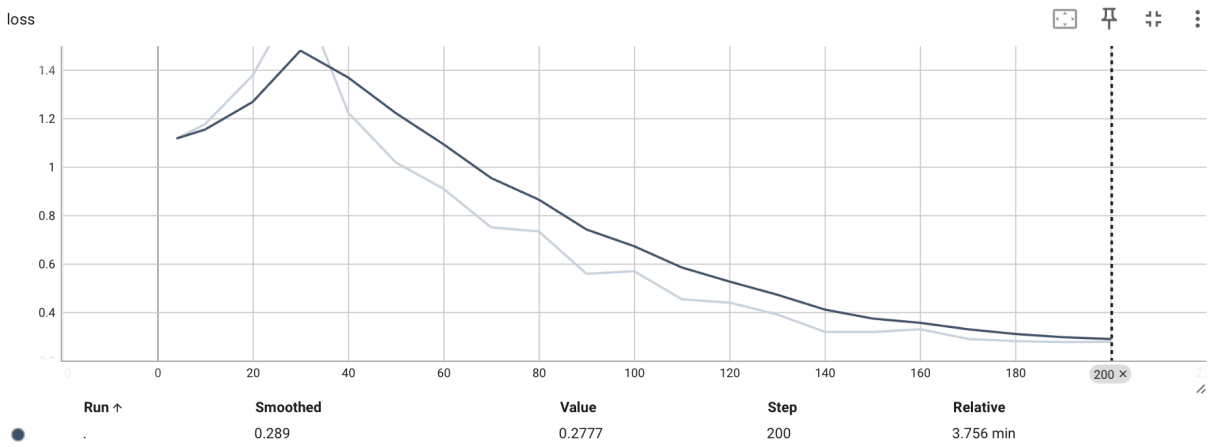

图 1: 微调 loss 曲线

3.2 模型权重转换及推理

我们使用如下命令将微调保存的 pytorch 格式模型转为 hugging face 格式模型，用于后续推理

```
xtuner convert pth_to_hf ./xtuner_config.py work_dirs/xtuner_config/iter_200.pth
    work_dirs/xtuner_config/iter_200_hf
```

之后用给定的标准测试问题“What is FRP?”进行测试，可以看到微调后模型能够给出引用以及回答。需要注意的是这个问题并不在微调的数据中，这表明了模型的泛化性。

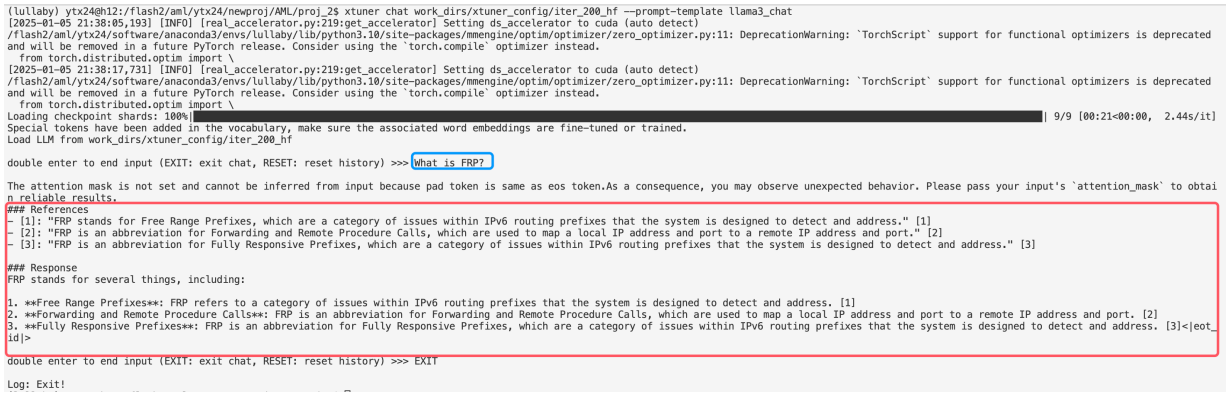

图 2: 微调后模型推理示例

4 RAG/微调模型效果测试

最后我们比较在“What is FRP”这个问题上，不同方法的回答效果。
Lightrag-原始版本输出（使用 hybrid mode）:

> Fully Responsive Prefix (FRP) refers to a type of IPv6 prefix where all addresses under the prefix respond to scans or probes. This means that when a network is scanned, all the addresses within that prefix w ill respond, typically indicating that they are active or accessible. However, the presence of FRPs can be problematic in network measurement campaigns and security assessments.
>
> Key Characteristics of FRPs:
>
> - **Responsive Addresses**: All addresses within an FRP prefix respond to scans, which can lead to incorrect conclusions about the actual number of active hosts on a network. - **Large Address Space**: An FRP can cover a large range of addresses, such as a /48 prefix, which can include 280 responsive addresses. - **Sparse Distribution**: Despite their potential impact, FRPs are relatively sparse in the vast IPv6 address space. - **Diverse Usage Scenarios**: FRPs can be used in various network environments, including cloud computing, content delivery networks (CDNs), and network security applications. - **Implementation Variations**: The implementation of FRPs can vary, with some based on existing functionalities in Linux systems and others using specific technical methods from cloud or CDN vendors.

Implications of FRPs:

- **Misleading Network Measurement**: The presence of FRPs can lead to misleading conclusions in network measurement campaigns, as resources may be wasted probing addresses that are not actual hosts. - **Security Concerns**: FRPs can pose security concerns, as they may hide malicious activities by providing a large number of responsive addresses. - **Efficiency Challenges**: Detecting and avoiding FRPs can be challenging due to their sparse distribution and diverse implementation patterns.

FRP Detection and Management:

- **Probing Tools**: Tools like Luori are used to actively probe IPv6 routing prefixes to identify and compile lists of FRPs, known as FRPlists. - **Analysis Algorithms**: Algorithms like the FAPD algorithm can help analyze FRPs and determine their mapping to hosts. - **Research Community**: The research community uses FRPlists and tools like Luori to study and understand FRPs, their implications for network security, and their practical applications in the network.

In summary, FRPs are a significant consideration in IPv6 networks due to their potential impact on network measurements and security. Understanding their characteristics and managing their presence is essential for accurate network assessments and effective security practices.

Lightrag-修改 prompt 后输出（使用 hybrid mode）:

References - [1]: "FRPs are a category of issues within IPv6 routing prefixes that the system is designed to detect and address. FRPs are a category of network protocols that are implemented in various ways and are identified through probing methods like Luori. FRPs refer to Forwarding and Remote Procedure Calls, which are used to map a local IP address and port to a remote IP address and port. FRPs refer to the Free Range Prefixes that are used in various network scenarios, including load balancing and proxy services." [1]

Response FRP, which stands for Fully Responsive Prefix, is a category of issues within IPv6 routing prefixes. It encompasses a variety of network protocols implemented in various ways and identified through probing methods like Luori. FRPs are used for mapping local IP addresses and ports to remote IP addresses and ports, and they are utilized in various network scenarios, including load balancing and proxy services [1].

LLaMA3-8b 微调输出:

References - [1]: "FRP stands for Free Range Prefixes, which are a category of issues within IPv6 routing prefixes that the system is designed to detect and address." [1] - [2]: "FRP is an abbreviation for Forwarding and Remote Procedure Calls, which are used to map a local IP address and port to a remote IP address and port." [2] - [3]: "FRP is an abbreviation for Fully Responsive Prefixes, which are a category of issues within IPv6 routing prefixes that the system is designed to detect and address." [3]

Response FRP stands for several things, including:

1. **Free Range Prefixes**: FRP refers to a category of issues within IPv6 routing prefixes that the system is designed to detect and address. [1] 2. **Forwarding and Remote Procedure Calls**: FRP is an abbreviation for Forwarding and Remote Procedure Calls, which are used to map a local IP address and port to a remote IP address and port. [2] 3. **Fully Responsive Prefixes**: FRP is an abbreviation for Fully Responsive Prefixes, which are a category of issues within IPv6 routing prefixes that the system is designed to detect and address. [3]