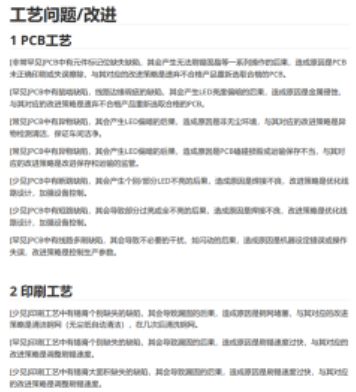
1 数据介绍

初始数据

自创2个缺陷和
可能会出现的异物影响

增加缺陷的出现频率



技术文档

一共57个不完全
相同的
缺陷-原因-措施

技术文档

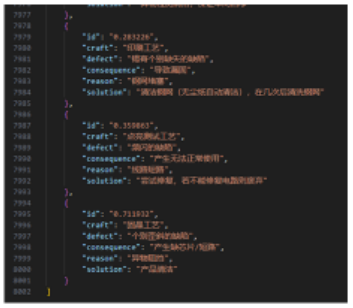
基于RAFT

Craft_QA_miniled

包含4个无关的
context作为干扰项

微调时增强LLM的
RAG能力

基于脚本

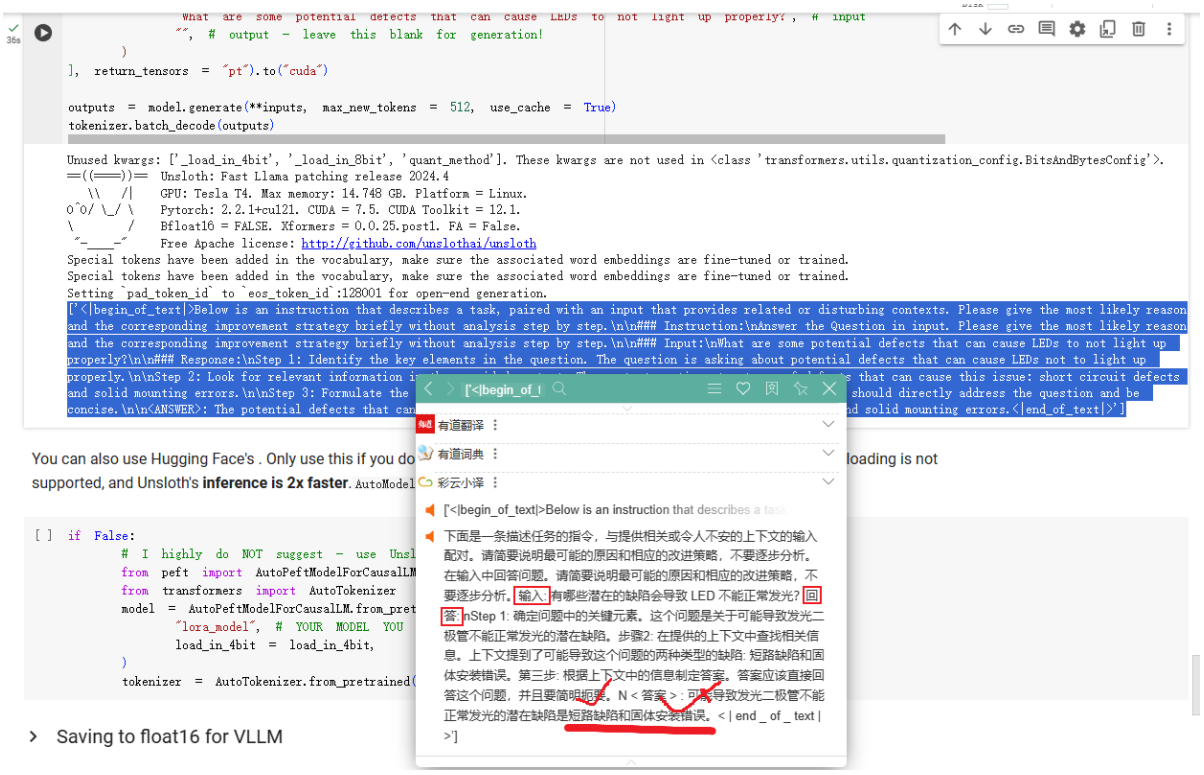


Reports_1000_sample

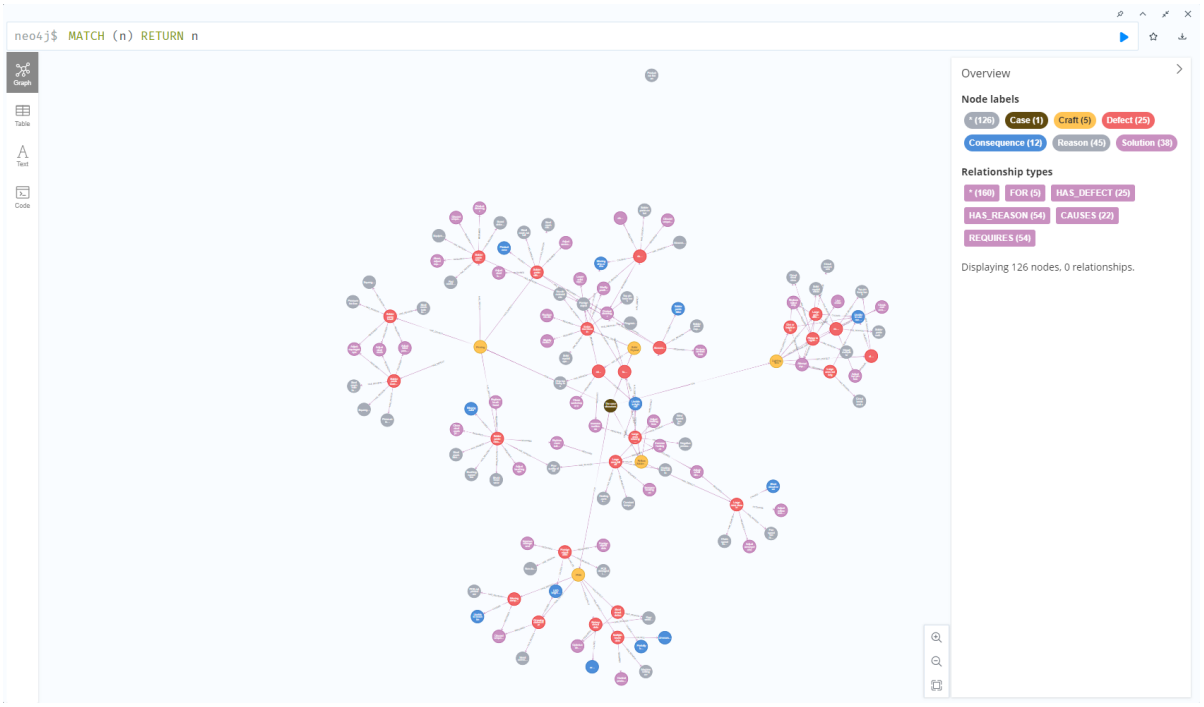
基于频率假设了
1000个出现缺陷的
样品

2 已有工作介绍

- **微调llama**
  - 基于peft库的LoRA完成微调——数据集为RAFT的QA数据



- **RAG构建向量库**
- **Graph知识图谱**
  - 通过技术文档构建知识图谱——neo4j数据库 + gpt-4-32k提取关系



- **调研介绍**
  - llama3大部分工作都是微调，少有缝合或加模块

    llama-pro通过文本修改的方式增加block

```
        output = {}
        for i in range(args.original_layers):
            for k in ckpt:
                if ('layers.' + str(i) + '.') in k:
                    output[k.replace(('layers.' + str(i) + '.'), ('layers.' + str(layer_cnt) + '.'))] = ckpt[k]
            layer_cnt += 1
            if (i+1) % split == 0:
                for k in ckpt:
                    if ('layers.' + str(i) + '.') in k:
                        if 'down_proj' in k or 'o_proj' in k:
                            output[k.replace(('layers.' + str(i) + '.'), ('layers.' + str(layer_cnt) + '.'))] = torch.zeros_l
                        else:
                            output[k.replace(('layers.' + str(i) + '.'), ('layers.' + str(layer_cnt) + '.'))] = ckpt[k]


                layer_cnt += 1

        assert layer_cnt==args.layers
        for k in ckpt:
            if not 'layers' in k:
                output[k] = ckpt[k]

        torch.save(output, args.output_path)
```

- **评估指标**——RAGAs自带的4个检索相关指标，参考论文lawyer-llama是否有相关领域的指标

lawyer llama

其工作与我们相似，微调+RAG，不同的是其做了很多组微调（毕竟数据够），也就是先大微调加入领域知识，再小微调加入推理或检索能力等。

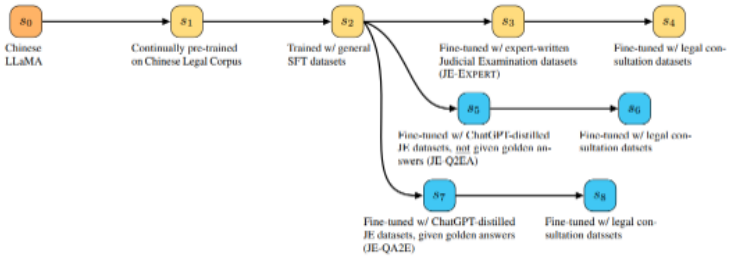


Figure 2: The training process of Lawyer LLaMA, where each node s_i represents the i-th training stage.

论文做了以下实验：

- 对比不同微调阶段的性能——其中涉及领域的**专业评价指标**和普遍的评价指标
- 计算了模型幻觉的占比
- 展示了专家以及其他模型输出的排名(哪个更好)
- 其他就是做了大量的讨论，展示了prompt和response

> 上下文精度、上下文召回率、事实一致性、答案相关性

3 未来计划

- 设计知识图谱检索的实验——展示多跳图谱的检索效果
- 设计其余的LLM实验——时间、幻觉占比……