第四次实验报告

PB21020651 武宇星

实验过程

数据集构建

使用 huggingface 上的 mehdiiraqui/twitter_disaster 数据集，其特征包含 id, keyword, location, text，依次二分类判断文本是否为灾难的描述。

在加载数据集后进一步分为训练集和验证集，为使数据集能用于微调大模型，需将 text 通过 tokenizer 转换为 token id 和注意力掩码的张量。这里大模型微调使用 huggingface 的库，也需进一步按 huggingface dataset 类要求将数据集标签列名称 target 改为 label；同时为了生成训练时一个 batch 的数据，调用 DataCollatorWithPadding 类将每个 batch 的序列填充至最长序列的长度

```
from transformers import AutoTokenizer, DataCollatorWithPadding
tokenizer = AutoTokenizer.from_pretrained(checkpoint, add_prefix_space=True
tokenizer.pad_token_id = tokenizer.eos_token_id
tokenizer.pad_token = tokenizer.eos_token

def llama_preprocessing_function(examples):
    return tokenizer(examples['text'], truncation=True, max_length=MAX_LEN)

col_to_delete = ['id', 'keyword','location', 'text']
tokenized_datasets = data.map(llama_preprocessing_function, batched=True, r
tokenized_datasets = tokenized_datasets.rename_column("target", "label")
tokenized_datasets.set_format("torch")
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

模型选择

大模型使用预训练的有文本分类头的 llama3-8B-Instruct，而在对序列作填充时由于 llama 没有默认填充 token，故使用 eos_token 结束标志作为填充

```
from transformers import AutoModelForSequenceClassification
import torch
model =  AutoModelForSequenceClassification.from_pretrained(
  pretrained_model_name_or_path=checkpoint,
  num_labels=2,
  device_map="auto",
  offload_folder="offload",
  trust_remote_code=True
)
model.config.pad_token_id = model.config.eos_token_id
```

微调方法

使用 LoRA (Low-Rank Adaptation，低阶适配) 进行微调，该方法会冻结模型的预训练权重，

仅更新一个新增的低秩矩阵。而对于 lora 的参数，主要有 TaskType 设为序列分类，r 为分解矩阵的秩，lora_alpha:为用于对习得权重进行缩放的参数，lora_dropout:为 lora 的 dropout 概率，bias 为偏置，将这些参数按照 lora 论文设置

```
from peft import get_peft_model, LoraConfig, TaskType
peft_config = LoraConfig(
    task_type=TaskType.SEQ_CLS, r=16, lora_alpha=16, lora_dropout=0.05, bia
    target_modules=[
        "q_proj",
        "v_proj",
    ],
)
```

训练

最后定义训练器以及训练参数进行训练，因为是二分类问题，故计算矩阵采用 accuracy，同时为画出 loss 变化图，需定义 trainercallback 类在每一步训练时记录返回 loss

```
def compute_metrics(eval_pred):
    accuracy_metric = evaluate.load("accuracy")
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    accuracy = accuracy_metric.compute(predictions=predictions, references=
    return {'accuracy': accuracy}
```

```
trainer = WeightedCELossTrainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets['train'],
    eval_dataset=tokenized_datasets["val"],
    data_collator=data_collator,
    compute_metrics=compute_metrics
)
```

```
class LogCallback(TrainerCallback):
    def __init__(self):
        self.losses = []
    def on_log(self, args, state, control, logs=None, **kwargs):
        if "loss" in logs:
            self.losses.append(logs["loss"])
```

实验结果

在两张 3090 上训练 30 个 epoch，loss 稳定下降，而在训练前模型预测准确率为 43.66%，差于随机选取，微调后则达到 79.32%，有显著提升

```
100%| | 191/191 [01:06<00:00,  2.85it/s]
Accuracy before fine-tuning: 0.4366
```

```
Accuracy after fine-tuning: 0.7932
```