组会汇报

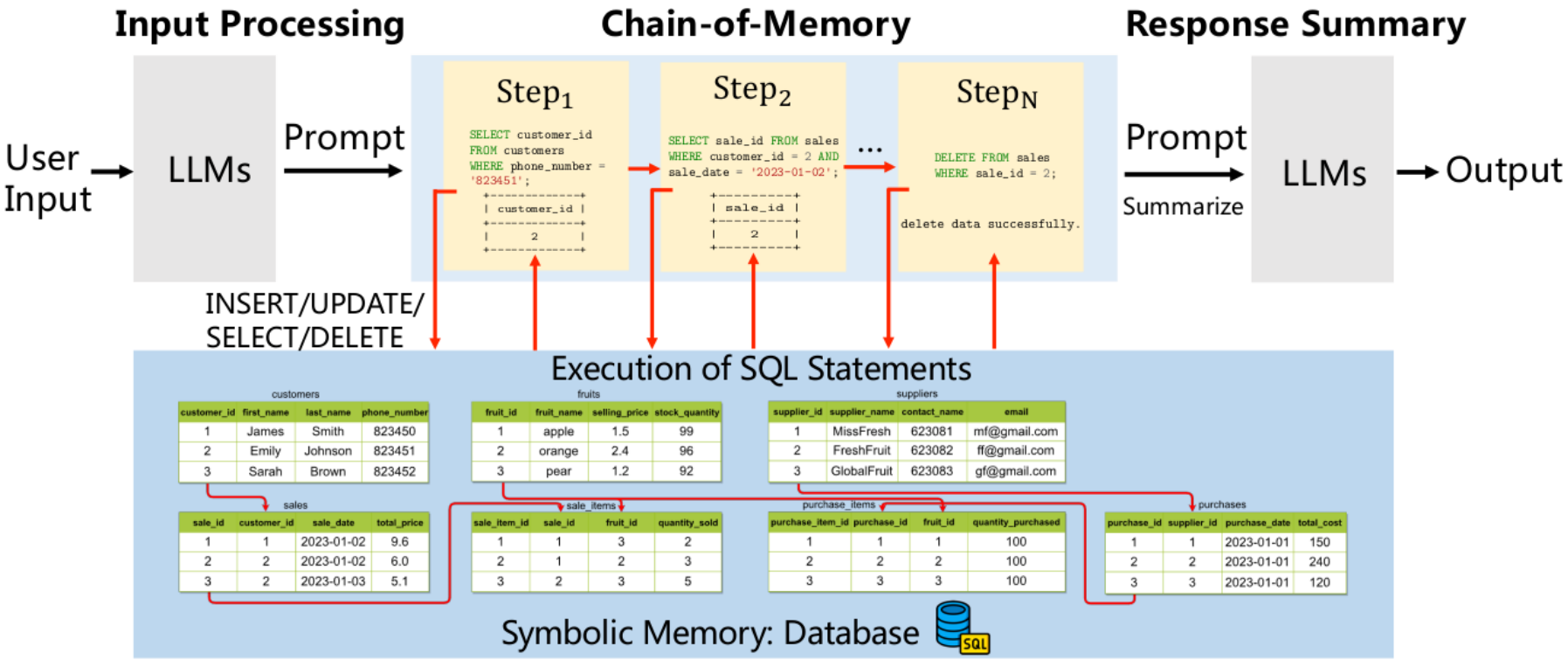
陈钶杰
专业:计算数学

June 13, 2023

目录

文章主要内容

- 传统的神经记忆机制无法支持LLM 模拟复杂推理，该论文提议使用数据库作为LLM 的新型符号存储器，以增强其推理能力。提出了一种名为ChatDB的框架，使用内存链方法来更有效地操作外部符号存储器，从而可以有效地操作内存和处理复杂的多表数据库交互，从而提高准确性和稳定性。
- 本文的重点是用数据库形式的符号记忆来增强LLM，以增强复杂的推理。



- input:手机号为823451的顾客在2023年1月2日买了什么东西?
- output:买了几个梨
- 说明：输入这段话以后，第一步模型尝试从所有顾客中根据手机号823451找出顾客的购物id号，第二步在销售号码中根据顾客的购物id号和购买日期得到销售号，...,通过一步一步有逻辑的推导得到最后的结果具体买了什么水果。

代码任务的的指令微调数据集示例

```
[
    {
        "instruction": "Write a C++ program that calculates the area of a
            rectangle, given its width and height.",
        "input": "width: 5, height: 8",
        "response": "#include <iostream>\nusing namespace std;\n\nint main() {\
            n int width = 5;\n int height = 8;\n int area = width * height;\n
            \n cout << \"Area of rectangle: \" << area << endl;\n return
            0;}",
        "output": "#include <iostream>\nusing namespace std;\n\nint main() {\n
            int width = 5;\n int height = 8;\n int area = width * height;\n\n
             cout << \"Area of rectangle: \" << area << endl;\n return 0;}"
    },
]
```

- Response（回复）：这是指令对应的人工生成的预期输出。回复提供了模型在给定输入下应该生成的参考答案或期望的输出。它可以是文本、图像、标签等，取决于任务的类型。

时间序列预测任务的指令微调数据集示例

```
{
    "Instruction": "For a weather data sequence, from the existing 336 sequences,
        predict the future 96 sequences?",
    "Input": "2020-01-01
        00:10:00,1008.89,0.71,273.18,-1.33,86.1,6.43,5.54,0.89,3.42,5.49,1280.62,1.
        break,2020-01-01 00:20:00,...,0.0,0.0,15.69,56.56,69.3,18.48,425.6,
        break",
    "output":"425.7,425.9,425.8,...,425.8"
}
```

- 指令微调数据样例
  1. instruction: ”For a weather data sequence, from the existing 336 sequences, predict the temperature indicator of the future 96 sequences ?”
  2. input:”2020-01-01 00:10:00,1008.89,...”(336行21维度的数据)
  3. output:”425.7,425.9,425.8,...,425.8”(96行特定维度的值)
- 可以将其中的标红进行修改,来对应不同的数据集.
- 问题:输入,输出序列都太长

Ptuning运行脚本

```
PRE_SEQ_LEN=128
LR=2e-2

CUDA_VISIBLE_DEVICES=0 python3 main.py \
    --do_train \
    --train_file AdvertiseGen/train.json \
    --validation_file AdvertiseGen/dev.json \
    --prompt_column content \
    --response_column summary \
    --overwrite_cache \
    --model_name_or_path ../chatglm-6b \
    --output_dir output/adgen-chatglm-6b-pt-$PRE_SEQ_LEN-$LR \
    --overwrite_output_dir \
    --max_source_length 64 \
    --max_target_length 64 \
    --per_device_train_batch_size 1 \
    --per_device_eval_batch_size 1 \
    --gradient_accumulation_steps 16 \
    --predict_with_generate \
    --max_steps 3000 \
    --logging_steps 10 \
    --save_steps 1000 \
    --learning_rate $LR \
    --pre_seq_len $PRE_SEQ_LEN \
    --quantization_bit 4
```

重要参数

- PRE_SEQ_LEN=128: 这个参数定义了预先截断（pre-truncation）的序列长度。它指定了模型在处理输入和输出序列之前将文本截断或填充到的最大长度。
- –max_source/target_length 64: 这个参数定义了输入/出序列的最大长度。如果输入/出序列超过该长度，将会被截断.
- –prompt_column content: 这个参数指定了输入数据中用作指令（prompt）的列。
- –response_column summary: 这个参数指定了输出数据中用作回复（response）的列。
- –predict_with_generate: 这个参数表示在评估过程中使用生成模式进行预测。在生成模式下，模型将生成回复而不是使用给定的回复
- –gradient_accumulation_steps 16: 这个参数定义了梯度累积的步数。梯度累积可以在训练过程中将多个批次的梯度累积起来，以增加批次大小的效果，从而减少显存的需求。

运行完的结果


```
{
        "epoch": 0.42,
        "train_loss": 4.3601599731445315,
        "train_runtime": 15178.543,
        "train_samples": 114599,
        "train_samples_per_second": 3.162,
        "train_steps_per_second": 0.198
}
```

问题

1. 时间序列这个数据集太长了,一般会有一个阶段的长度,所以这边是不是有点问题.对于每个数据向量,能否进行一定的压缩?能否模仿前面论文的方式进行单独存储数据?
2. 输入数据的方式是否可以改变,接下来打算去看看代码生成繁琐的任务具体是怎么做的?

谢谢老师和同学的聆听!