NLPDL-Assignment3 Report

📖 Introduction

The assignment is divided into two parts, one is the testing of the quantization and kv_cache of the large language model and the manual implementation of kv_cache, and the other is the improvement and testing of LLM reasoning.

🔵 Task1

🔬 *Task1.1*

We first compare the implementation of the transformers library of kv-cache and quantization. As for kv-cache we use the `generate` method of `transformers` library, and we set the argument `use_cache` to `True` for using kv-cache. For quantization of the kv-cache we set the argument `cache_implementation` to `True` for applying quantization of the kv-cache. Here we present our experiment result of 10 round experiments:

	Naive	KV_Cache	KV_Cache with Quantization
Time Cost (on avg second)	11.70580622	6.751685917	***5.857653189***
Memory Cost (on avg MB)	17,239	1,269	***1,239***

💡 *Task1.1 Analysis*

From the result we see that KV_Cache indeed help the Language Model to inference much faster, that's mainly because we avoid the full multiply of two large KV matrix. And quantization of KV_Cache slightly help the inference to become faster, that is because quantization decrease the time cost of any calculation that involves the KV_Cache.

What's not meeting our expectation is that the memory cost of KV_Cache with Quantization does not decrease severely compared to the KV_Cache setting. After discussion, we believe that the cause of this is that GPT2 is a rather small model, and only applying quantizaion at the KV_Cache cannot decrease the many memory taken by other calculation components. So the decrease of the memory cost is really small. And the GPU memory decrease of the KV_Cache compared to the Naive setting is mainly because the parameter of matrix multiplication that are involved in inference is much smaller

compared to the naive setting (only need a small amount of parameter to refresh the attention score matrix).

🔬 *Task1.2*

This task mainly involve the implementation of the KV_Cache not using the transformers library. We present our code in our github repo and here we only present our results of a ten rounds experiments:

	w/o KV_Cache	w/ KV_Cache
Time Cost (on avg sec)	9.012378	***7.505301***
Memory Cost (on avg MB)	24,190	***16,018***

🔴 Task2

In this part, we are using the GSM8K benchmark to verify the performance of different reasoning techniques. The resoning techniques include:

- naive (directly output the result)
- CoT (let's think step by step)
- ICL (provide reasoning example)
- CoT + Reflexion (reflect the first attempt)

The prompt of each setting is:

- **Naive**

```
naive_prompt = "Here is a math question for you, directly output the answer: \nQ: "
```

- **CoT Prompt**

```
cot_prompt = "Here is a math question for you, let's think step by step: \nQ: "
```

- **ICL Prompt**

```
icl_prompt = "Q: A car travels 60 miles in 2 hours. What is the speed?\nA: To find the speed, d
```

- **CoT + Reflexion**

```
refine_prompt1 = "Here is a math question for you, let's think step by step: \nQ: "

refine_prompt2_1 = "I think your previous answer may have errors, the question and the previous

refine_prompt2_2 = "Refine your previous answer and output the answer in a new line after the st
```

🔬 *Experiments Result:*

	Naive	CoT	ICL	CoT + Reflexion
Accuracy	37.15%	77.47%	***89.98%***	75.66%

💡 *Result Analysis:*

Here we first present an example where the naive implementation failed to produce the right answer while advanced method have a better performance:

```
question = "Question: Josh decides to try flipping a house.  He buys a house for $80,000 and the

cot_answer is comments below
"""
To determine Josh's profit, we need to follow these steps:

1. **Calculate the total cost:**
   - Purchase price of the house: $80,000
   - Repairs: $50,000
   - Total cost = $80,000 + $50,000 = $130,000

2. **Determine the increased value of the house:**
   - The value of the house increased by 150%.
   - Original value (before repairs) = $80,000
   - Increased value = Original value + (150% of original value)
   - Increased value = $80,000 + (1.5 * $80,000) = $80,000 + $120,000 = $200,000

3. **Calculate the profit:**
   - Profit = Increased value - Total cost
   - Profit = $200,000 - $130,000 = $70,000

#### $70,000
"""
icl answer is comments below

"""
A: First, calculate the total cost of the house after repairs. This is $80,000 (initial cost) +

Next, determine the new value of the house after the 150% increase. Since the value increased by

Calculate 250% of the original value: 250% of $80,000 = (250/100) * $80,000 = 2.5 * $80,000 = $2

Finally, calculate the profit by subtracting the total cost from the new value: $200,000 (new va

#### 70000
"""

reflexion answer is comments below

"""
Apologies for the confusion. Let's carefully go through the steps again to ensure accuracy.

1. **Calculate the total investment:**
```

```
   - Purchase price of the house: $80,000
   - Repairs: $50,000
   - Total investment = $80,000 + $50,000 = $130,000

2. **Determine the increased value of the house:**
   - The value of the house increased by 150%.
   - Original value of the house before repairs: $80,000
   - Increased value = Original value + (150% of original value)
   - Increased value = $80,000 + (1.50 × $80,000) = $80,000 + $120,000 = $200,000

3. **Calculate the profit:**
   - Profit = Increased value - Total investment
   - Profit = $200,000 - $130,000 = $70,000

#### $70,000
"""
```

We see that all three advanced reasoning methods outperforms the naive method, indicating that LMs do have the ability to benefit from test-time inference. Also, to our surprise we see that the reflexion + CoT setting has weaker performance compared to the pure CoT setting. This indicates that in a lot of circumstances the reflexion process modified the right answer into the wrong answer. Thinking deeper we realise that in a lot of condition, the CoT process maybe using the wrong reasoning path to produce the correct result. So it is explainable that the reflexion + CoT setting has weaker performance since if the reasoning path is wrong, the LM will definitely try to fix it, but the fixing process will still lead to the wrong answer.