NLPDL assignment3

2200017853 李长烨

实验环境:NVIDIA GeForce RTX 3090, AMD EPYC 7H12 64-Core Processor

task1

Compare Experiment

本实验对比了文本生成任务种不同缓存策略和量化策略对推理效率的影响，包括基线模型，KV-cache缓存，FP16量化和INT8量化，我们针对三种生成长度(tokens=300, 600, 900)的情况，记录了生成的吞吐量和峰值显存占用，重复5次取平均值，最终得到各个情况下的性能表现结果。

max_new_tokens	Method	Throughput (tokens/second)	Peak Memory Usage (MB)
300	Naive	161.86	508.88
	KV-Cache	141.50	507.63
	FP16 Quantization	158.33	270.94
	INT8 Quantization	34.27	187.43
600	Naive	150.44	557.55
	KV-Cache	156.68	543.04
	FP16 Quantization	158.54	287.03
	INT8 Quantization	33.59	205.03
900	Naive	87.14	675.13
	KV-Cache	156.54	627.74
	FP16 Quantization	157.39	330.32

max_new_tokens	Method	Throughput (tokens/second)	Peak Memory Usage (MB)
	INT8 Quantization	33.56	248.81

Result Analysis

1. **Naive（基线模型）**

- **吞吐量**:
  - 随生成长度增加，吞吐量显著下降（300 → 600：161.86 → 150.44，600 → 900：150.44 → 87.14）。
  - 这是由于生成长度越长，序列依赖性增加，计算复杂度上升。
- **显存占用**:
  - 随生成长度增加显存占用明显上升（300 → 600：508.88 MB → 557.55 MB，600 → 900：557.55 MB → 675.13 MB）。
  - 反映出较高的显存开销，尤其是在长序列生成时。

2. **KV-Cache（缓存优化）**

- **吞吐量**:
  - 对比Naive策略，短序列时吞吐量略低（300: 141.50 vs. 161.86），但生成长度增加后表现更优（600: 156.68 vs. 150.44，900: 156.54 vs. 87.14）。
  - 长序列生成时，缓存减少了重复计算，显著提升效率。
- **显存占用**:
  - 较Naive稍低，但总体变化趋势一致，生成长度增加时显存使用量增大（300 → 900：507.63 MB → 627.74 MB）。

3. **FP16 Quantization（半精度量化）**

- **吞吐量**:
  - 在所有生成长度上均接近或略高于Naive（300: 158.33，600: 158.54，900: 157.39）。
  - 相比Naive和KV-Cache，吞吐量更加稳定，受生成长度影响较小。
- **显存占用**:
  - 显著降低，峰值显存相比Naive减少约一半（300: 270.94 MB vs. 508.88 MB，600: 287.03 MB vs. 557.55 MB，900: 330.32 MB vs. 675.13 MB）。
  - FP16量化对显存的优化效果突出，尤其适合显存受限的环境。

4. **INT8 Quantization（8位量化）**

- **吞吐量**:
  - 显著降低，所有生成长度的吞吐量均远低于其他方法（300: 34.27，600: 33.59，900: 33.56）。

    - 量化过于激进，导致性能损失，尤其是生成速度的严重下降。
- **显存占用**：
    - 最低显存需求（300: 187.43 MB，600: 205.03 MB，900: 248.81 MB）。
    - 虽然显存优化效果最佳，但吞吐量的极大下降使其在高效推理任务中不具优势。

Conclusion

- **综合性能推荐**：FP16量化策略在保持较高吞吐量的同时显存占用显著降低，是多数场景的首选。
- **长序列生成场景推荐**：KV-Cache在长序列场景中具有显著优势，可提升效率。
- **显存极限场景推荐**：若显存限制极为严苛（如移动设备），可考虑INT8量化，但需接受吞吐量的极大下降。
- **基线模型（Naive）**：适用于无需优化的简单实验，或硬件资源充足时的参考基准。

Customized KV-Cache

本实验中实现了自定义的KV-Cache，与Golden Greedy Decoding方法在Decode时间上进行了比较，结果如下：

- **Time taken for golden greedy decoding without KV cache**: 12.54s
- **Time taken for customized KV-Cache decoding**: 6.43s

Result Analysis

相比于Golden Greedy Decoding方法，customized KV-Cache方法几乎提升了一倍的decode速度，通过缓存键值对（key-value pairs）减少了重复计算，避免了历史上下文的多次重新计算。该优化确保每一步解码只处理新增的token，显著提升了解码效率，最终使解码时间减少约48.7%。

此外，本实验中还尝试设计了prefix cache机制，但由于前缀获取和存储机制未设计完成，故暂时无法使用。

task2

本报告研究了多种推理技术在GSM8k数据集上的应用效果，评估其生成准确解答的能力。这些技术包括直接回答（Direct Answer）、思维链推理（Chain of Thought, CoT）、反思（Reflexion）、上下文学习（In-Context Learning, ICL）以及它们的组合。实验的目标是分析每种方法的性能，并探讨各技术在特定场景下的表现。

配置与推理技术

1. **Direct Answer（直接回答）**：直接生成答案，无中间推理步骤。由于Direct Answer直接输出最终答案，测试上下文学习（ICL）和反思（Reflexion）没有意义，因此未结合这两种技术。
2. **Chain of Thought（思维链推理，CoT）**：引导模型逐步推理，从而提升准确性。
3. **Reflexion（反思）**：引入自我反思机制以修正和优化生成结果。

4. **In-Context Learning（上下文学习，ICL）**：在提示中提供示例以指导模型的推理。
5. **组合方法**：结合CoT与Reflexion、ICL或两者，以发挥各自优势。

Direct Answer由于其直接生成答案的特点，不适合测试ICL和Reflexion的作用，故未进行单独的ICL和Reflexion实验。

实验结果

以下是各推理技术在GSM8k数据集上的准确率：

推理技术	准确率	正确/总数
Direct Answer	40%	523/1319
CoT	90%	1188/1319
CoT + Reflexion	89%	1180/1319
CoT + ICL	87%	1150/1319
CoT + ICL + Reflexion	90%	1184/1319

结果分析

- **Direct Answer**：作为基准方法，仅达到40%的准确率，缺乏中间推理步骤可能是性能较差的主要原因。由于该方法直接生成最终答案，未测试ICL和Reflexion的组合，因为这两种技术依赖于分步推理提示，不适用于Direct Answer。
- **CoT**：表现最优（90%准确率），通过逐步推理解决了复杂问题，显示出显著优势。
- **CoT + Reflexion**：略低于CoT（89%准确率），可能因反思引入了一定的冗余或噪声。
- **CoT + ICL**：达到87%准确率。虽然上下文示例提供了指导，但与CoT结合的效果不如单独使用CoT。
- **CoT + ICL + Reflexion**：表现与单独使用CoT相当（90%准确率），说明在此任务中，Reflexion和ICL未能显著增强CoT的效果。

推理技术的影响

- **CoT**：对复杂多步骤问题影响最大，是最具普适性的推理方法。
- **Reflexion**：在某些情况下有效，但与CoT结合时效果有限。可能更适合简单问题或独立使用。
- **ICL**：在提供上下文示例时表现良好，但与CoT协同作用不明显。

结论

实验表明，CoT是解决GSM8k数据集中问题的最有效技术。尽管Reflexion和ICL有潜力，但与CoT结合未显著提升性能。