Contents

1 Problem: 2

NOTE :- I am running the code on MacBook GPU named 'mps'.

1.1 Part- a

Completed all the TODO sections in the provided code to create a functioning transformer model with the ability to autoregressively generate 'n' tokens. It is done with proper comments in the code, present in the zip file.

1.2 Part- b and c

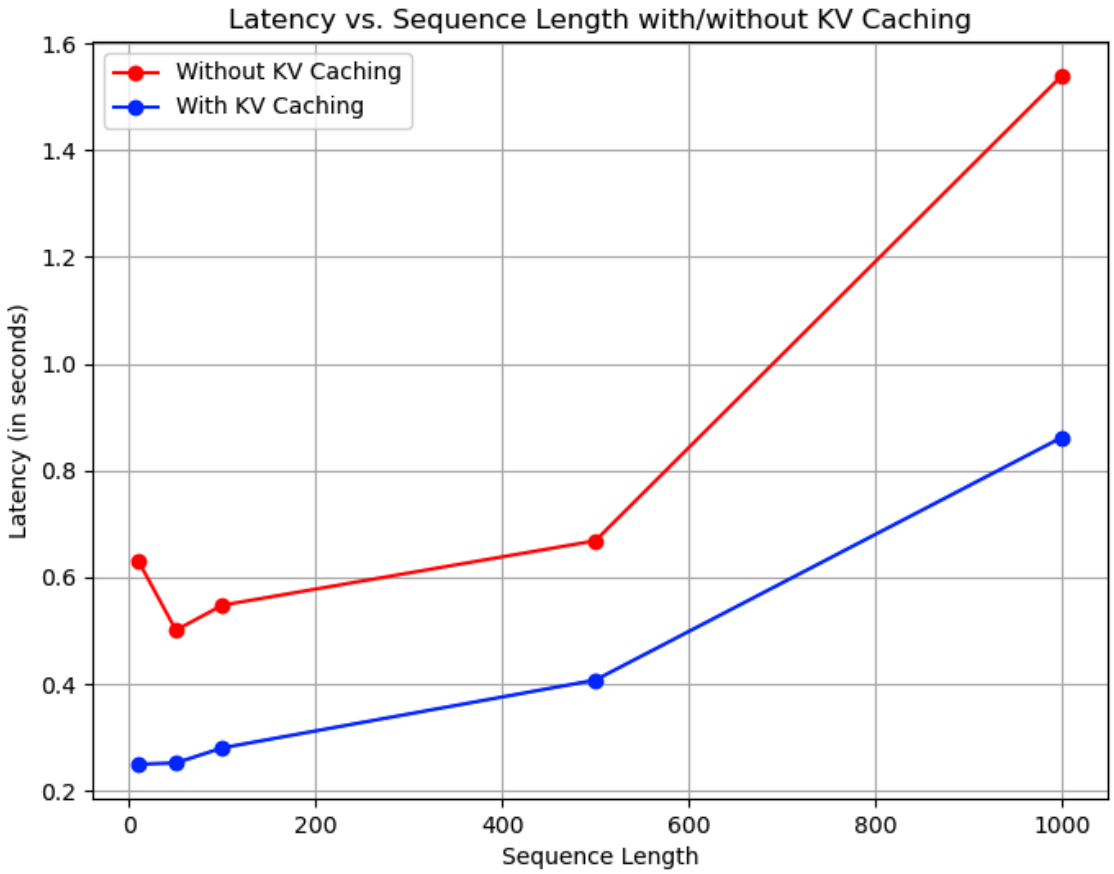
These two parts also well implemented in the code file.

```
(base) neha@Meghas-MacBook-Air sysForML_assign4 % python transformer.py --mode evaluate
Successfully loaded model state dictionary from model_state_dict.pt
Test cases loaded from test_cases.json
Evaluation Results:
  Num test cases: 10
  All tests passed: True
  Pass rate: 100.00% (10/10)
(base) neha@Meghas-MacBook-Air sysForML_assign4 % python transformer.py --mode kv_evaluate
Successfully loaded model state dictionary from model_state_dict.pt
Test cases loaded from test_cases.json
Evaluation Results:
  Num test cases: 10
  All tests passed: True
  Pass rate: 100.00% (10/10)
```

1.3 Part- d

```
(base) neha@Meghas-MacBook-Air sysForML_assign4 % python transformer_skeleton.py --mode benchmark
Seq Len: 10 | No KV: 0.6300s | With KV: 0.2501s
Seq Len: 50 | No KV: 0.5011s | With KV: 0.2526s
Seq Len: 100 | No KV: 0.5480s | With KV: 0.2809s
Seq Len: 500 | No KV: 0.6686s | With KV: 0.4078s
Seq Len: 1000 | No KV: 1.5388s | With KV: 0.8618s
2025-03-28 22:38:22.712 python[49765:1950394] +[IMKClient subclass]: chose IMKClient_Modern
2025-03-28 22:38:22.712 python[49765:1950394] +[IMKInputSession subclass]: chose IMKInputSession_Modern
Successfully loaded model state dictionary from model_state_dict.pt
Benchmarking...
Results:
  Without KV cache: 0.0088 seconds
  With KV cache: 0.0100 seconds
  Speedup: 0.88x
```

- **Faster Generation with KV Caching:** KV caching significantly reduces latency, especially for longer sequences.
- **Minimal Difference for Short Sequences:** For shorter sequences (10, 50), the performance gain is less noticeable.
- **Exponential Growth Without Caching:** Without KV caching, latency increases almost exponentially as sequence length increases.
- **Efficient Handling of Long Sequences:** KV caching eliminates redundant computations, leading to faster token generation in longer sequences.
- **Conclusion:** KV caching greatly enhances the efficiency of decoder-only transformers, particularly in long-sequence scenarios.


Latency vs. Sequence Length with/without KV Caching
Without KV Caching
With KV Caching
Latency (in seconds)
Sequence Length
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000

References

[1] LCS2-IITD, Large Language Models (LLMs): Introduction and Recent Advances. Available: `https://lcs2-iitd.github.io/ELL881-AIL821-2401/`.

[2] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness," 2022. Available: `https://arxiv.org/abs/2205.14135`.

[3] Stable Softmax Implementation. Available: `https://jaykmody.com/blog/stable-softmax/`.

[4] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," *arXiv preprint arXiv:1909.08053*, 2019.