Anlp Ass-4

Quantizing gpt2: Analysis of Time and Memory Predictions

This document outlines various quantization techniques applied to the gpt2 model and analyzes their impact on memory usage, loss, and execution time, focusing on explaining the observed trends in time and memory usage.

Part 1 - Manual Quantization

Baseline Model

- ***Memory Usage (Forward Pass):**** 838.78 MB
- ***Loss:**** 26.38
- ***Time Taken:**** 25.80 seconds

Full Model Quantization

- ***Model Memory Usage:**** 294.25 MB (Significantly reduced due to lower precision weights)
- ***Memory Usage (Forward Pass):**** 1465.78 MB (Increased due to overhead introduced by quantization/dequantization operations)
- ***Loss:**** 26.95 (Slight increase compared to baseline, expected due to reduced precision)
- ***Time Taken:**** 21.86 seconds (Faster than baseline potentially due to smaller memory footprint during some operations, despite the overhead)
- ***Comment:**** While model size decreases, forward pass memory increases, suggesting overhead in managing quantized weights. Time reduction might be due to faster memory access or specific optimizations for quantized operations.

Full Model without lm_head Quantization

- ***Model Memory Usage:**** 255.60 MB (Further reduction compared to full quantization)
- ***Memory Usage (Forward Pass):**** 1269.30 MB (Lower than full model quantization but still higher than baseline)
- ***Loss:**** 26.41 (Closer to baseline, suggesting lm_head might be more sensitive to quantization)
- ***Time Taken:**** 21.58 seconds (Slightly faster than full quantization, consistent with reduced memory footprint)
- ***Comment:**** Excluding lm_head from quantization seems to balance memory usage and accuracy. The forward pass memory still shows overhead, but is lower than fully quantized, indicating lm_head contributes to it.

Only lm_head Quantization

- ***Model Memory Usage:**** 548.99 MB (Larger than other quantizations, lm_head might be a significant portion of the model)
- ***Memory Usage (Forward Pass):**** 1036.32 MB (Lower than full quantization, suggests other parts of the model contribute more to the overhead)
- ***Loss:**** 26.92 (Similar to full quantization, confirms lm_head's sensitivity to quantization)
- ***Time Taken:**** 20.92 seconds (Fastest among manual quantization, possibly due to a better trade-off between quantization and computation)
- ***Comment:**** Quantizing only lm_head provides a faster inference with reasonable loss, but the model size is larger compared to other methods. This indicates that the benefit of quantization depends on the specific part of the model being targeted.

Last 4 Attention Layers Quantization

- ***Model Memory Usage:**** 425.43 MB

- ***Memory Usage (Forward Pass):**** 983.95 MB (Further reduction in forward pass memory)
- ***Loss:**** 26.40 (Minimal impact on loss, suggesting these layers are less sensitive to quantization)
- ***Time Taken:**** 20.91 seconds (Fastest among manual quantization, consistent with lower forward pass memory)
- ***Comment:**** Targeting specific layers like the last attention layers provides significant memory and time improvements with minimal accuracy loss, demonstrating that selective quantization can be effective.

Only q, k, v Matrices Quantization

- ***Model Memory Usage:**** 425.43 MB
- ***Memory Usage (Forward Pass):**** 989.83 MB (Slightly higher than last 4 attention layers, potentially due to different access patterns)
- ***Loss:**** 26.40
- ***Time Taken:**** 21.17 seconds (Slightly slower than last 4 attention layers)
- ***Comment:**** Quantizing only q, k, v matrices shows similar benefits as quantizing last attention layers. The slight difference in time and memory might be due to implementation details and data access patterns.

Part 2 - Quantization using bitsandbytes (bnb)

4-bit Quantization

- ***Model Memory Usage:**** 134.06 MB (Significantly smaller than manual quantization due to 4-bit precision)
- ***Memory Usage (Forward Pass):**** 308.80 MB (Significantly smaller than all manual quantization, bnb is efficient)
- ***Loss:**** 31.30 (Larger increase in loss, indicating a trade-off between memory and accuracy at 4-bit)
- ***Time Taken:**** 16.43 seconds (Fastest among all methods, due to highly optimized bnb implementation and reduced memory footprint)

- ***Note:**** `low_cpu_mem_usage` set to True.
- ***Comment:**** bnb's 4-bit quantization offers substantial memory and time savings, but at the cost of higher loss. This indicates its suitability for scenarios where speed and memory are prioritized over accuracy.

8-bit Quantization

- ***Model Memory Usage:**** 176.53 MB (Larger than 4-bit but still smaller than manual quantization)
- ***Memory Usage (Forward Pass):**** 494.14 MB (Larger than 4-bit but smaller than manual quantization)
- ***Loss:**** 26.56 (Close to baseline, 8-bit offers a good balance between accuracy and memory)
- ***Time Taken:**** 29.28 seconds (Slower than 4-bit but faster than the baseline, indicating 8-bit trade-offs)
- ***Note:**** `low_cpu_mem_usage` set to True.
- ***Comment:**** 8-bit quantization with bnb provides a good balance, reducing memory usage and improving speed compared to the baseline while maintaining accuracy close to the original model.

4-bit NF4 Quantization

- ***Model Memory Usage:**** 134.06 MB (Same as 4-bit quantization, as expected)
- ***Memory Usage (Forward Pass):**** 494.86 MB (Higher than 4-bit, NF4 might introduce different overhead)
- ***Loss:**** 28.38 (Lower loss than 4-bit quantization, indicating NF4's better accuracy)
- ***Time Taken:**** 15.96 seconds (Fastest among all methods, suggesting NF4 is computationally efficient)
- ***Comment:**** NF4 (NormalFloat4) in bnb provides a better accuracy than standard 4-bit quantization while maintaining similar speed benefits, making it a good option for scenarios needing both efficiency and reasonable accuracy.

General Comments on Time and Memory Trends

- ***Quantization Precision:**** Lower precision (e.g., 4-bit) leads to smaller model sizes but can increase loss.
- ***Quantization Overhead:**** Introducing quantization and dequantization operations can increase the memory footprint during the forward pass, even if the model size is reduced.
- ***Implementation Efficiency:**** Libraries like bnb are optimized for quantized operations, leading to significant speed improvements and potentially lower memory usage during inference.
- ***Selective Quantization:**** Targeting specific parts of the model (e.g., certain layers or matrices) can provide a good balance between memory reduction, speed improvement, and accuracy preservation.
- ***Data Movement:**** Reduced memory footprint can lead to faster inference due to reduced data movement between memory and processing units.

By understanding these factors, one can choose the appropriate quantization strategy based on the specific requirements of the application, considering the trade-offs between memory, speed, and accuracy.