Advanced Natural Language Processing

QUANTIZATION OF LARGE LANGUAGE MODELS

Hardik Sharma

Introduction

As Large Language Models (LLMs) such as GPT-2 grow in size, the demand for computational resources also increases. Quantization is a technique used to reduce the bit-width of model weights, significantly lowering memory usage and inference latency while attempting to maintain model accuracy. This report explores several quantization approaches applied to a pre-trained GPT-2 model, including manual quantization and techniques using the BitsAndBytes library.

Quantization Methods

Manual Quantization

Manual quantization involves reducing the precision of the model weights without using external libraries. We implemented two types:

- **Whole-Model Quantization (Task 1)**: Converted the entire model from floating-point (FP32) to 8-bit integer (INT8).
- **Selective Component Quantization (Task 2)**: Applied quantization selectively to specific components such as the feed-forward networks (FFNs) and attention layers, leaving other parts of the model at full precision.

The main idea behind these methods was to reduce memory usage and latency. However, aggressive quantization (whole-model) led to significant increases in perplexity, indicating loss of numerical precision.

Hugging Face 8-bit and 4-bit Quantization

For Task 3 and Task 4, we used the Hugging Face integration with the BitsAndBytes library:

- **8-bit Quantization (Task 3)**: Applied dynamic 8-bit quantization using the Hugging Face API, which provided a balance between efficiency and accuracy.
- **4-bit Quantization (Task 4)**: Utilized the Hugging Face 4-bit quantization configuration, which offers more aggressive compression but may impact model accuracy slightly.

These methods leveraged efficient low-bit representations for weights, resulting in significant reductions in memory usage while maintaining acceptable accuracy.

NF4 Quantization (Task 5)

NF4 (Normalized Float 4-bit) quantization is a non-linear quantization technique provided by the BitsAndBytes library. Unlike linear quantization, NF4 uses normalized floating-point values, which helps in retaining more numerical information. This method aims to minimize the loss of precision typically associated with low-bit quantization, making it well-suited for sensitive layers in transformer models.

Results and Analysis

Table 1 presents a comprehensive comparison of the different quantization approaches applied to the GPT-2 model.

Discussion

The results show that:

- **Manual Quantization** provided the highest reduction in memory but at the cost of significant increases in perplexity, especially for whole-model quantization.

Table 1: Comparison of Quantization Methods

Metrics	**Manual Quantization**		**BitsAndBytes Quantization**		
	Whole	**Selective**	**8-bit**	**4-bit**	**NF4**
Latency Before	45.58 ms	42.56 ms	43.41 ms	42.74 ms	42.57 ms
Latency After	29.09 ms	30.41 ms	31.03 ms	31.89 ms	16.91 ms
Memory Before	474.70 MB	474.70 MB	474.70 MB	474.70 MB	474.70 MB
Memory After	118.97 MB	158.97 MB	156.35 MB	115.85 MB	115.85 MB
Perplexity Before	30.07	30.07	30.07	30.07	30.07
Perplexity After	∞	46,417,899.07	30.19	33.42	32.05

- **8-bit and 4-bit Hugging Face Quantization** offered a balanced approach, achieving memory reduction with minimal impact on perplexity.
- **NF4 Quantization** achieved the best latency improvement and memory reduction while maintaining reasonable perplexity, highlighting its effectiveness for transformer models.

Conclusion

Quantization techniques are effective in reducing the computational and memory requirements of LLMs. Manual quantization methods can lead to significant performance trade-offs if not applied selectively. Using advanced quantization methods like NF4, provided by the BitsAndBytes library, offers a superior balance between efficiency and accuracy, making it a preferred choice for practical deployment.

References

- Datacamp Tutorial on Quantization for LLMs: https://www.datacamp.com/tutorial/quantization-for-large-