Quantização de Modelos LLM Usando PyTorch

Quantização é uma técnica de compressão que torna modelos de linguagem mais leves e eficientes, facilitando sua execução com menor custo computacional.

Neste tutorial, você aprenderá:

- O que é quantização e por que usa-lá
- Como quantizar um modelo transformer usando PyTorch
- Como medir ganho de perfomance, redução de tamanho e impacto na acurácia

O Que é Quantização?

Quantização consiste em converter pesos de alta precisão (como FLOATt32) para formatos numéricos menores (como INT8), reduzindo significativamente o uso de memória e acelerando a inferência, com pouca ou nenhuma perda de acurácia

Existem dois tipos de quantização:

- Quantização Pós-Treinamento (PTQ): A quantização é realizada após o treinamento do modelo. É simples de implementar e não exige reentreinamento, mas pode resultar em perda significativa de acurácia.
- Treinamento com Reconhecimento de Quantização (QAT): A quantização é simulada durante o processo de treinamento. Ao final, os pesos são convertidos para tipos numéricos mais compactos, geralmente com perdas mínimas de acurácia em comparação com o PTQ.

> Neste tutorial, focaremos na Quantização Pós-Treinamento (PTQ), que é mais prática para quem já tem um modelo treinado.

Quantização com PyTorch

PyTorch é uma biblioteca de código aberto para aprendizado de máquina. Ela é amplamente usada para construir e treinar modelos de forma flexível e intuitiva.

Pré-Requisitos

```
pip install torch torchvision torchaudio transformers
```

Carregando um Modelo Pré-Treinado

Vamos utilizar o **DistilBERT**, uma versão mais leve do BERT, treinado para análise de sentimentos.

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch

nome_modelo = "distilbert-base-uncased-finetuned-sst-2-english"
```

```
tokenizer = AutoTokenizer.from_pretrained(nome_modelo)
modelo = AutoModelForSequenceClassification.from_pretrained(nome_modelo)
modelo.eval()    # Coloca o modelo em modo de avaliação
```

Testando o Modelo Original

```
texto = "Eu amo a Universidade Federal Fluminense!"

inputs = tokenizer(texto, return_tensors="pt")
with torch.no_grad():
    outputs = modelo(**inputs)

probs = torch.nn.functional.softmax(outputs.logits, dim=-1)
print(probs)
```

Aplicando a Quantização Pós-Treinamento (PTQ)

```
from torch.quantization import quantize_dynamic

modelo_quantizado = quantize_dynamic(
    modelo,
    {torch.nn.Linear},  # Aplica quantização nas camadas lineares
    dtype=torch.qint8   # Usa 8 bits em vez de 32
)
```

Testando o Modelo Quantizado

```
with torch.no_grad():
    outputs = modelo_quantizado(**inputs)

probs = torch.nn.functional.softmax(outputs.logits, dim=-1)
print(probs)
```

Comparando Velocidade, Tamanho e Precisão

1. Tamanho dos Modelos

```
import os

torch.save(modelo.state_dict(), "original.pt")
torch.save(modelo_quantizado.state_dict(), "quantizado.pt")
```

```
print("Tamanho original:", os.path.getsize("original.pt") / 1e6, "MB")
print("Tamanho quantizado:", os.path.getsize("quantizado.pt") / 1e6, "MB")
```

2. Tempo de Inferência

```
import time

# Texto de exemplo para o teste
texto = "Texto para teste de performance sobre os benefícios da quantização."
inputs = tokenizer(texto, return_tensors="pt")

# Modelo original
start_time = time.time()
for _ in range(100):
    with torch.no_grad():
        modelo(**inputs)
tempo_original = (time.time() - start_time) / 100

# Modelo quantizado
start_time = time.time()
for x in range(100):
    with torch.no_grad():
        modelo_quantizado(**inputs)
tempo_quantizado = (time.time() - start_time) / 100

print(f"Tempo de inferência original (média de 100 execuções):
{tempo_original:.6f} segundos")
print(f"Tempo de inferência quantizado (média de 100 execuções):
{tempo_quantizado:.6f} segundos")

speedup = tempo_original / tempo_quantizado
print(f"\nO modelo quantizado é {speedup:.2f}x mais rápido.")
```

3. Teste de Acurácia

```
print("\n--- Comparando a Saída dos Modelos ---")

with torch.no_grad():
    probs_original = torch.nn.functional.softmax(modelo(**inputs).logits, dim=-1)
    probs_quantizado =
torch.nn.functional.softmax(modelo_quantizado(**inputs).logits, dim=-1)

print("Original:", probs_original.numpy())
print("Quantizado:", probs_quantizado.numpy())

diff = torch.abs(probs_original - probs_quantizado).sum().item()
print(f"Erro absoluto total: {diff:.6f}")
```

Conclusão

A quantização é uma técnica simples, porém poderosa, para otimizar modelos de linguagem. Ao reduzir a precisão dos pesos, é possível diminuir significativamente o tamanho do modelo e acelerar o tempo de inferência, com impacto mínimo na acurácia. Neste tutorial, vimos como aplicar a quantização pós-treinamento em um modelo transformer com PyTorch, além de comparar os resultados em termos de desempenho, uso de memória e precisão.

Esse tipo de otimização é especialmente útil em cenários que exigem modelos mais leves e rápidos — como protótipos, validações rápidas e aplicações em larga escala.

Referências

- DistilBERT - HuggingFace
- Tutorial sobre Quantização em PyTorch (oficial)