Bài tập Llama2

Trần Hồng Đăng MSSV 22022646 | Hồ Tú Minh MSSV 22022674 | Bùi Thế Long MSSV 22022647 | Ngô Văn Kiệt MSSV 22022643

I. KIẾN TRÚC CỦA LLAMA2

LLaMA2 là một mô hình **Transformer Decoder-only**, gồm nhiều lớp Transformer xếp chồng lên nhau. Mỗi khối Transformer bao gồm:

RMSNorm (Root Mean Square Layer Normalization)

RMSNorm thay thế LayerNorm truyền thống, giúp ổn định huấn luyện và giảm độ phức tạp tính toán. Công thức chuẩn hóa RMSNorm:

$$\|\mathbf{x}\|_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \epsilon}$$

trong đó:

- x là tensor đầu vào.
- n là số phần tử của tensor x.
- x_i là thành phần của tensor x.

Grouped Query Attention (GQA)

GQA giúp giảm yêu cầu bộ nhớ bằng cách nhóm Query, sử dụng chung Key-Value. Công thức tính Attention Score:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Rotary Positional Embedding (RoPE)

Thay vì Positional Encoding cố định, RoPE mã hóa vị trí bằng cách xoay ma trận embedding, cụ thể: Query và Key được reshaped để phân tách thành phần chẵn/lẻ, tính toán tần số quay cho mỗi chiều của head:

$$\theta_i = \theta^{-\frac{2i}{d}}$$

Áp dụng phép quay (RoPE Transform)

$$\begin{bmatrix} q_1' \\ q_2' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \tag{2}$$

trong đó θ phụ thuộc vào vị trí token.

II. HÀM TỐI ƯU

AdamW Optimizer

AdamW tối ưu hóa việc cập nhật trọng số, giúp huấn luyện nhanh và giảm overfitting, cụ thể là: Lấy **state** hiện tại của p (các tham số **weight, bias**), nếu **state** chưa được khởi tạo thì khởi tạo **state** với:

- Moment bậc 1: m
- Moment bậc 2: v
- Số bước: $step = 0$

Nếu state đã tồn tại, lấy $m, v, step$ của state hiện tại. Cập nhật gradient bậc 1, bậc 2 theo công thức:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{3}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{4}$$

$$\tag{5}$$

Bias correction để hiệu chỉnh độ lệch nhằm khắc phục sự thiên lệch trong các giá trị moment ban đầu

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Cập nhật tham số theo công thức

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{6}$$

Thêm **weight decay**:

$$\theta_t = \theta_t - \alpha \cdot \lambda \cdot \theta_t$$

Cross Entropy Loss

Hàm mất mát sử dụng trong huấn luyện mô hình:

$$L = -\sum y_i \log(\hat{y}_i) \tag{7}$$

III. TEXT GENERATION

Mô hình sinh văn bản bằng cách lấy mẫu theo nhiệt độ:

1) Tính xác suất softmax sau khi chia logits cho T:

$$P(y_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}} \tag{8}$$

2) Nếu $T = 0.0$, chọn token có xác suất cao nhất.
3) Nếu $T > 0.0$, lấy mẫu từ phân phối xác suất.

IV. KẾT QUẢ VÀ TỐI ƯU THAM SỐ

Text Continuation

- temperature = 0: "I have wanted to see this thriller for a while, and it didn't disappoint. Keanu Reeves, playing the hero John Wick, is this day. He was playing with his toy car, driving it around the living room. Suddenly, he heard a loud crash. He had broken the car and was very sad. John was angry and he shouted at his little brother. He was only three years old and he was only three. He was only three years old. He was very ups" (output ra được đúng ngữ pháp tiếng anh, từ vựng hẹp)
- temperature = 1: "I have wanted to see this thriller for a while, and it didn't disappoint. Keanu Reeves, playing the

hero John Wick, is it!" As John toddled up to the sweet aroma, he held his mum's hand tight, hoping he would finally reach the top step. But as he stepped, he felt so heavy, like an apple. He had made a mistake! His mum apologisedDen handlely leaving, so he started to cry. His m" (output ra được gần đúng ngữ pháp tiếng anh, còn sai chính tả, từ vựng sáng tạo hơn)

Zero-shot, Prompt-based Sentiment Analysis

- **SST** (epoch=5, batch_size=80, lr=$2e^{-5}$, dropout=0.3)
  - dev acc: 0.411
  - test acc: 0.410
- **CFIMDB** (epoch=5, batch_size=10, lr=$2e^{-5}$, dropout=0.3)
  - dev acc: 0.857
  - test acc: 0.469

Task-specific Fine Tuning

- **Với siêu tham số mặc định**
  - **SST** (epoch=5, batch_size=80, lr=$2e^{-5}$, dropout=0.3)
    - dev acc: 0.411
    - test acc: 0.410
  - **CFIMDB** (epoch=5, batch_size=10, lr=$2e^{-5}$, dropout=0.3)
    - dev acc: 0.857
    - test acc: 0.469
- **Tối ưu hóa siêu tham số (với 5 epoch)**
  - **SST** (kết quả tốt nhất với batch_size=64, lr=$2e^{-5}$) (2 bảng dưới là kết quả thử nghiệm với các tổ hợp learning rate và batch size khác nhau, accuracy được tính trên cả tập test và dev)

dev acc	16	32	64	80
$2e^{-3}$	0.360	0.381	0.344	0.369
$2e^{-4}$	0.391	0.380	0.395	0.391
$2e^{-5}$	0.421	0.415	**0.428**	0.411
$2e^{-6}$	0.374	0.352	0.344	0.327
$2e^{-7}$	0.274	0.263	0.261	0.269

TABLE I
BẢNG KẾT QUẢ DEV ACC

test acc	16	32	64	80
$2e^{-3}$	0.378	0.373	0.362	0.357
$2e^{-4}$	0.414	0.378	0.415	0.385
$2e^{-5}$	0.420	0.421	**0.426**	0.410
$2e^{-6}$	0.383	0.362	0.328	0.324
$2e^{-7}$	0.266	0.256	0.253	0.242

TABLE II
BẢNG KẾT QUẢ TEST ACC

- **CFIMDB** (kết quả tốt nhất với batch_size=10, lr=$2e^{-6}$)

dev acc	4	10
$2e^{-3}$	0.841	0.841
$2e^{-4}$	0.906	0.894
$2e^{-5}$	**0.918**	0.857
$2e^{-6}$	0.784	0.645
$2e^{-7}$	0.596	0.527

test acc	4	10
$2e^{-3}$	0.453	0.488
$2e^{-4}$	0.527	0.486
$2e^{-5}$	0.498	0.469
$2e^{-6}$	0.502	**0.674**
$2e^{-7}$	0.629	0.291

- **Thử nghiệm với các tỉ lệ dropout khác nhau** (default là 0.3)
  - **SST**

dev acc	0	0.3	0.5
batch_size=64, lr=$2e^{-5}$	0.421	**0.428**	0.427

test acc	0	0.3	0.5
batch_size=64, lr=$2e^{-5}$	0.406	**0.426**	0.414

- **CFIMDB**

- **Nhận xét và kết luận**
  - Tham số tốt nhất cho việc fine-tune bộ SST là
    - • Learning rate = $2e^{-5}$
    - • Batch size = 64
    - • Dropout = 0.3
  - Tham số tốt nhất cho việc fine-tune bộ CFIMDB là
    - • Learning rate = $2e^{-6}$
    - • Batch size = 10
    - • Dropout = 0.3
- **Hướng cải thiện**
  - Train trên nhiều epoch hơn

dev acc	**0**	**0.3**	**0.5**
batch_size=64, lr=$2e^{-5}$	0.690	**0.918**	0.600

test acc	**0**	**0.3**	**0.5**
batch_size=64, lr=$2e^{-6}$	0.545	**0.674**	0.193