Nama: Eduardus Bagus Wicaksono
NIM: 22/493128/TK/53996
Prodi: Teknologi Informasi

Implementasi Transformer

I. Struktur Folder

- layers.py: Berisi implementasi komponen dasar Transformer seperti multi-head attention, feed-forward network, layer normalization, dan positional encoding.
- transformers.py: Menyusun decoder-only Transformer dengan stack DecoderLayer, embedding, causal mask, dan output projection.
- tokenizers.py: Menyediakan tokenizer sederhana untuk mengubah teks menjadi token ID dan sebaliknya dengan special tokens.
- main.py: Menjalankan GUI Tkinter untuk input teks dan menampilkan prediksi token berikutnya dari Transformer NumPy.

II. Desain Arsitektur

Model yang diimplementasikan adalah GPT-style decoder-only Transformer, menggunakan NumPy tanpa library deep learning. Arsitektur terdiri dari:

1. Token Embedding: Token input diubah menjadi vektor berdimensi d_model melalui embedding lookup.
2. Positional Encoding: Ditambahkan ke embedding untuk memberikan informasi posisi token dalam sequence.
3. Stack Decoder Layers (jumlah num_layers):
    - LayerNorm → Multi-Head Self-Attention → Residual Add
        - Multi-Head Attention membagi embedding menjadi num_heads dan melakukan scaled dot-product attention per head.
    - LayerNorm → Position-Wise Feed-Forward Network (FFN) → Residual Add
        - FFN terdiri dari dua lapisan linear dengan ReLU di lapisan pertama.
4. Output Projection: Hasil akhir diproyeksikan ke ukuran vocab menggunakan matriks linear (d_model → vocab_size) untuk menghasilkan logits.
5. Softmax: Hanya untuk token terakhir untuk prediksi token berikutnya.

Struktur ini mengikuti desain GPT yang menekankan autoregressive decoding dan residual connections untuk stabilitas training.

III. Alasan Pemilihan Positional Encoding

Positional encoding sinusoidal dipilih karena:

- Memberikan informasi urutan token tanpa menambah parameter yang harus dilatih.
- Bentuk sinusoidal memungkinkan model membaca pola relatif posisi antar token.

- Lebih stabil dibanding embedding posisi yang di-train untuk implementasi sederhana dengan NumPy.

Fungsi sinus (sin) digunakan untuk indeks genap, dan cosinus (cos) untuk indeks ganjil, memastikan representasi posisi unik untuk setiap token.

IV. Penjelasan Causal Mask

Causal mask digunakan agar self-attention hanya melihat token sebelumnya dan token saat ini, mencegah model mengakses informasi masa depan. Mask berbentuknya: upper-triangular matrix dengan nilai -1e9 di atas diagonal utama. Dalam perhitungan attention: $attention\_score \mathrel{+}= mask$ sehingga softmax mengabaikan posisi yang di-masking.

V. Bukti Uji

```
print("----- Scaled Dot-Product Attention Debug -----")
batch, heads, seq_len, depth = 2, 3, 4, 8
Q = K = V = np.random.randn(batch, heads, seq_len, depth)
mask = create_causal_mask(seq_len)

attn_output, attn_weights = scaled_dot_product_attention(Q, K, V, mask)
print("Output shape:", attn_output.shape)           # (2, 3, 4, 8)
print("Attention weights shape:", attn_weights.shape)  # (2, 3, 4, 4)
print("Sum over last axis (softmax):", np.sum(attn_weights, axis=-1))
print("Masked positions (should be near 0):", attn_weights[0,0,0,1:])
print("-----------------------------------------------")
```

```
-----------------------------
----- Scaled Dot-Product Attention Debug -----
Output shape: (2, 3, 4, 8)
Attention weights shape: (2, 3, 4, 4)
Sum over last axis (softmax): [[[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]

 [[1. 1. 1. 1.]
  [1. 1. 1. 1.]
  [1. 1. 1. 1.]]]
Masked positions (should be near 0): [0. 0. 0.]
-----------------------------------------------
```

Hasil debug menunjukkan bahwa scaled dot-product attention bekerja dengan benar:

- Dimensi output attn_output adalah (2, 3, 4, 8), sesuai dengan (batch_size, num_heads, seq_len, depth), menandakan tensor attention dikembalikan dengan bentuk yang benar.
- Dimensi attention weights adalah (2, 3, 4, 4), yang merepresentasikan bobot perhatian untuk setiap query terhadap semua key dalam sequence.
- Sum softmax pada axis terakhir adalah 1 untuk semua query, membuktikan bahwa softmax menormalisasi bobot attention dengan benar.
- Masked positions (attn_weights[0,0,0,1:]) bernilai 0, menunjukkan bahwa causal mask berhasil memblokir perhatian ke token masa depan.

Dapat disimpulan, uji sederhana ini membuktikan bahwa dimensi tensor, softmax, dan masking telah diterapkan dengan benar pada implementasi attention.