DL_05_Transformers: explicação

Esse notebook trabalha **construindo e visualizando a arquitetura de atenção de um Transformer**, usando tanto `Keras` quanto ferramentas como `bertviz`. Vamos passar pelo que está sendo feito em cada parte relevante, com foco na **arquitetura de atenção**:

🔷 1. Criação das embeddings com posição

```
embedding_layer = TokenAndPositionEmbedding(20, 10, 768)(input_layer)
```

- Isso cria embeddings que combinam **token embeddings** com **positional embeddings** (essencial para Transformers captarem ordem dos tokens).
- `20`: vocabulário de 20 palavras
- `10`: sequência de 10 tokens
- `768`: dimensão do embedding (igual ao BERT-base)

🔷 2. Projeções Q, K, V e atenção escalonada

```
query = Dense(d_model)(embedding_layer)
key = Dense(d_model)(embedding_layer)
value = Dense(d_model)(embedding_layer)
```

Essas camadas `Dense` aprendem projeções lineares de Q (query), K (key) e V (value), todas derivadas do mesmo `embedding_layer`.

Depois, temos o bloco de atenção:

```
dot_product = tf.matmul(query, key, transpose_b=True)
scale = dot_product / sqrt(d_model)
```

```
softmaxed = softmax(scale)
z = tf.matmul(softmaxed, value)
```

Esse é o clássico **Scaled Dot-Product Attention**, núcleo do Transformer.

🔷 3. Feed-Forward Network (FFN)

```
ff_1 = Dense(4 * 768, activation='relu')(z)
ff_2 = Dense(768)(ff_1)
ffn_output = LayerNormalization()(ff_2)
```

- Esse bloco representa a **camada feed-forward** após o mecanismo de atenção.
- Primeira `Dense` expande, depois a segunda reduz à dimensão original.
- `LayerNormalization` estabiliza o output.

🔷 4. Visualização de atenção com BERT e bertviz

```
attention = model(**viz_input).attentions
head_view(attention, tokens, starter, heads = [8])
```

Essa parte usa o modelo **pré-treinado BERT-base** da Hugging Face e visualiza as **atenções de head 8**, comparando duas frases semelhantes.

✅ Conclusão (visão geral)

Esse notebook:

- **Implementa um bloco de atenção do Transformer em Keras**, incluindo Q/K/V e atenção escalonada.
- Adiciona **feed-forward com normalização** como no paper "Attention is All You Need".
- Usa `bertviz` para **visualizar como o BERT realmente aprende a "olhar" para os tokens**.