Positional Encoding

In embedding layer, same token ID gets mapped to the same vector representation, regardless of where the token ID is positioned in the input sentence

* The cat sat on the mat → [5, 1, 5] from embedding layer,
* On the mat the cat sat

(current vector embedding approach has no idea of position of 'cat'

Same token IDs result in the same embedding vectors → [-123 456 789; 101 212 345; -123 456 789]

(3) It is helpful to inject additional position info to LLM

(4) there are 2 types of positional embeddings

↓ absolute

↓

For each position in input sequence, unique embedding is added to the token's embedding to convey its exact location

The positional vectors have same dimension as the original token embeddings (for easy addition matrix purposes)

↓ relative

↓

The emphasis is on relative position or distance between tokens. Model learns the relationships in terms of 'how far apart' rather than at which exact position

↓

Advantage: model can generalize better to sequence of varying lengths, even if it has not seen such lengths during training

⑤ Both types of positional encodings enable LLMs to understand the order and relationship between tokens, ensuring more accurate and context aware predictions

⑥ The choice between the 2 depends on specific application and nature of data being processed

Absolute: Suitable when fixed order of tokens is crucial, such as sequence generation

Relative: Suitable for tasks like language modeling over long sequences, where the same phrase can appear in different parts of the sequence

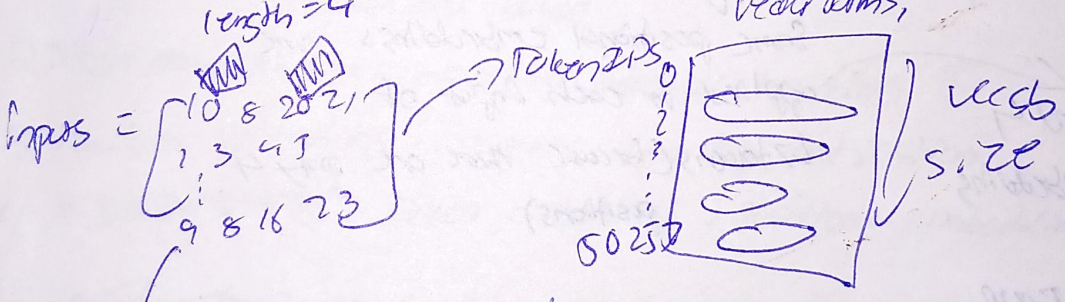
⑦ OpenAI models (GPT) use absolute positional embeddings that are optimized during training process. This optimization is part of the model training itself.

⑧ Get input data from dataloader

inputs = [ ] batch size = 8

↳ 8 text samples w/ 4 tokens size,

context length = 4

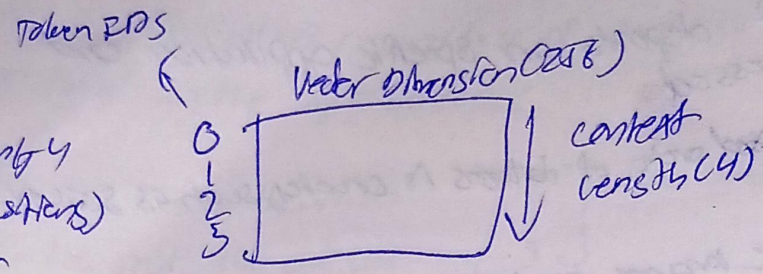


One embedding vector of size 256 length is generated for each token in input

[(256)(256)(256)(256)] 8×4×256

for each of these 8×4 values, we have a 256 dimension tensor

(11) Create another embedding layer to get token embeddings



Only 4 positions) for max_length=4 context length

(12) Input

Inputs = batch_size=8 [4 256]

Context length = 4

Locate one input token embeddings (256 (256) (256) (256)

[10 & 20 21] only 4 bc context_size >& 4

Need to add one positive vector to each of these 4 token embeddings

Same positional embeddings are applied to each input of 4 tokens (because there are only 4 positions)

Hence, only need 4 positional embedding vectors

only create on FOUR positions, since context is 4.

(13) Generate 4 positional embedding vectors from positional embedding matrix

(13) Generate 4 positional embeddings vectors from the positional embeddings matrix

[4, 256]

(14) Add the token embeddings to positional embeddings

$$\begin{bmatrix} (256)(256)\times(256)(256) \\ \vdots \\ (256)(256)(256)(256) \end{bmatrix} + \begin{bmatrix} (256)(256)(256)(256) \end{bmatrix}$$

8 × 4 × 256 (token embeddings)

4 × 256 (positional embeddings)

= Input Embeddings

(8 × 4 × 256)

Per position, each of 256 vector broadcasts 4 × 256 to all 8 samples in batch to support the matrix addition