

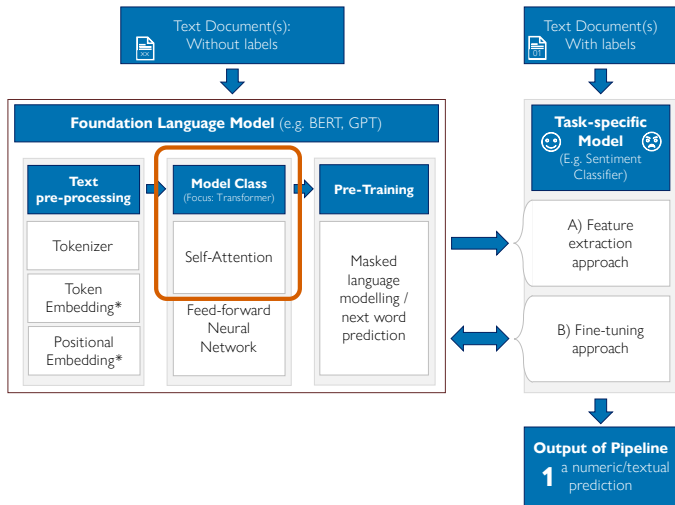
Understanding and Training Language Models: **Self-attention and Transformers**

Erik-Jan Senn

Faculty of Mathematics and Statistics, University of St. Gallen

CSH Autumn School at University of Hohenheim
September/October 2024

Where are we?



Language Modelling Pipeline

Roadmap

Self-attention

Transformers

Self-attention and Transformers

Self-attention

What is attention?

Motivating example:

*Yesterday, my friends went to the cinema because the weather was nasty.
The movie had amazing visuals, but the story was quite boring.*

What is attention?

Motivating example:

*Yesterday, my friends went to the cinema because the weather was **nasty**.*

*The movie had **amazing** visuals, but the story was quite **boring**.*

Possible task:

- ▶ How many adjectives are in this sentence? 3

What is attention?

Motivating example:

Yesterday, my friends went to the cinema because the weather was nasty.

*The **movie** had **amazing visuals**, but the **story was quite boring**.*

Possible task:

- Is this movie review positive, negative, or neutral? *neutral*

What is attention?

Motivating example:

Yesterday, my friends went to the cinema because the weather was nasty.

The movie had amazing visuals, but the story was quite boring.

Possible task:

- ▶ Is it going to rain today? *pretty likely*

What is attention?

Motivating example:

*Yesterday, my friends went to the cinema because the weather was nasty.
The movie had amazing visuals, but the story was quite boring.*

We observe:

- ▶ Some parts of the sequence are more important in general (e.g., the was never highlighted).
- ▶ Each task requires **looking at different tokens** in the sequence.

What is attention?

Motivating example:

*Yesterday, my friends went to the cinema because the weather was nasty.
The movie had amazing visuals, but the story was quite boring.*

We observe:

- ▶ Some parts of the sequence are more important in general (e.g., the was never highlighted).
- ▶ Each task requires looking at different tokens in the sequence.

Idea of attention: Allow tokens to interact (to represent and learn more general concepts, interactions), by paying attention to other tokens.

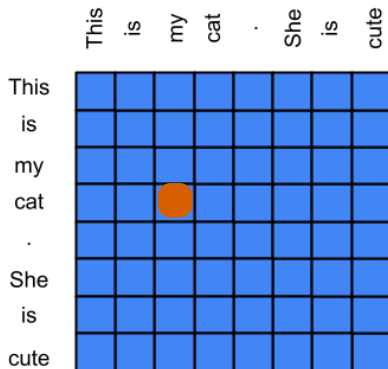
Self-attention Matrix

- **A** is a $l \times l$ **interaction matrix** for tokens within a sequence.

	This	is	my	cat	.	She	is	cute
This								
is								
my								
cat								
.								
She								
is								
cute								

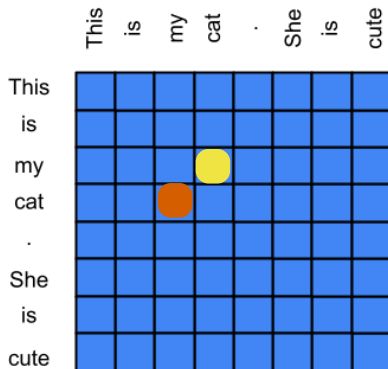
Self-attention Matrix

- ▶ **A** is a $l \times l$ **interaction matrix** for tokens within a sequence.
- ▶ **Red** element **A**_{4,3}:
 - ▶ How much **attention** does token cat pay to token my (one-directional)?
 - ▶ More specific: How much will the **output representation** of the token cat **be influenced** by the representation of token my.



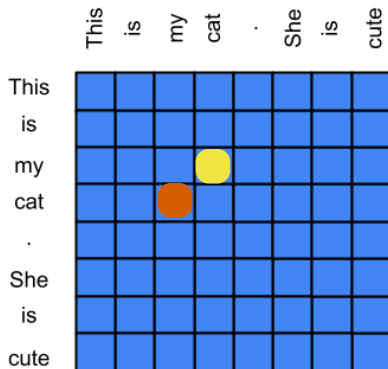
Self-attention Matrix

- ▶ **A** is a $l \times l$ **interaction matrix** for tokens within a sequence.
- ▶ **Red** element **A**_{4,3}:
 - ▶ How much **attention** does token cat pay to token my (one-directional)?
 - ▶ More specific: How much will the **output representation** of the token cat **be influenced** by the representation of token my.
- ▶ **Yellow** element **A**_{3,4}: How much attention does token my pay to token cat (one-directional)?

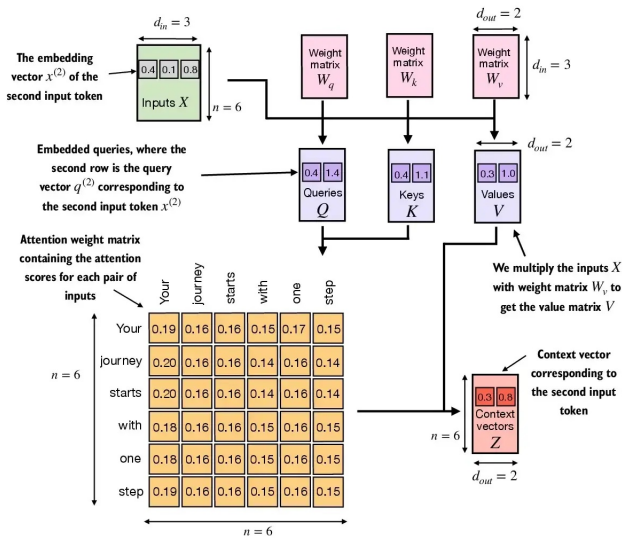


Self-attention Matrix

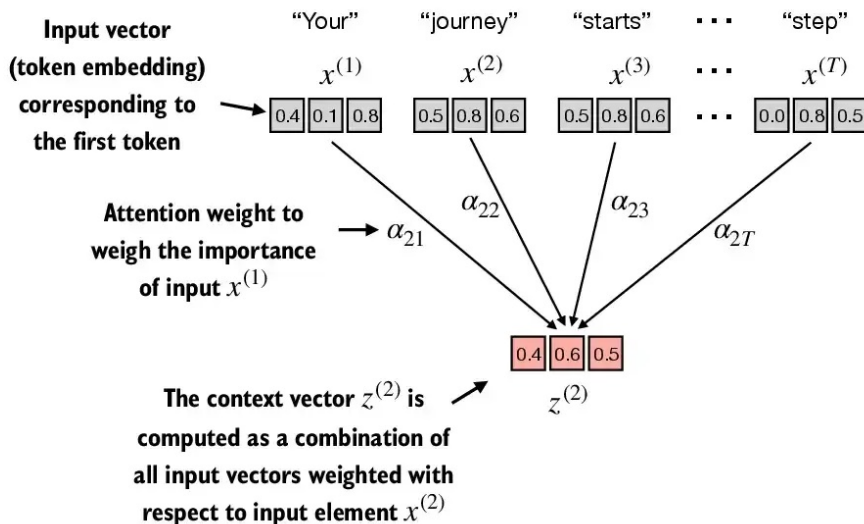
- ▶ **A** is a $l \times l$ **interaction matrix** for tokens within a sequence.
- ▶ **Red** element **A**_{4,3}:
 - ▶ How much **attention** does token **cat** pay to token **my** (one-directional)?
 - ▶ More specific: How much will the **output representation** of the token **cat** **be influenced** by the representation of token **my**.
- ▶ **Yellow** element **A**_{3,4}: How much attention does token **my** pay to token **cat** (one-directional)?
- ▶ **A** is non-symmetric, rows normalized to 1.



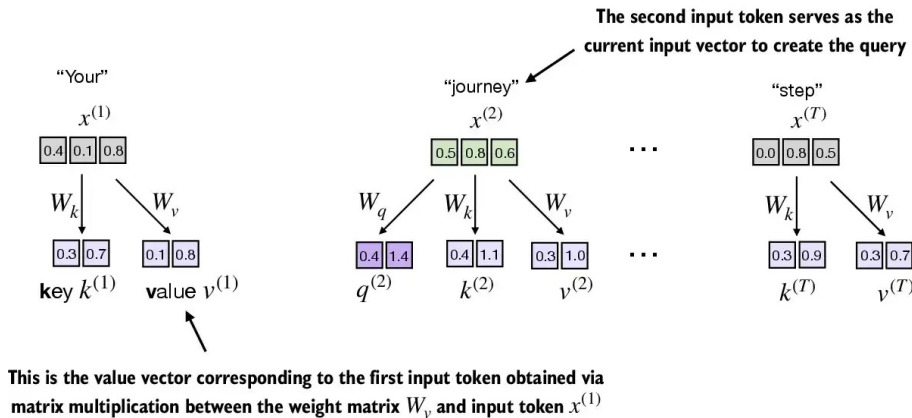
Self-attention Overview



Self-attention Step-by-Step



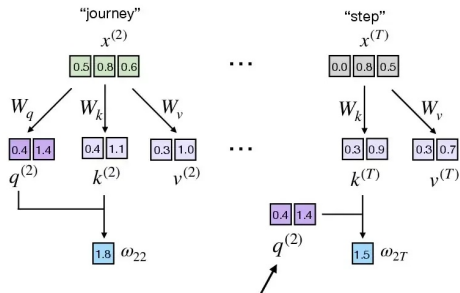
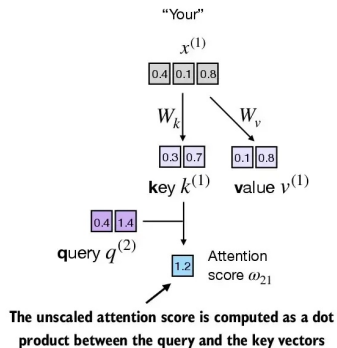
Self-attention Step-by-Step



© 2024 Sebastian Raschka

Source

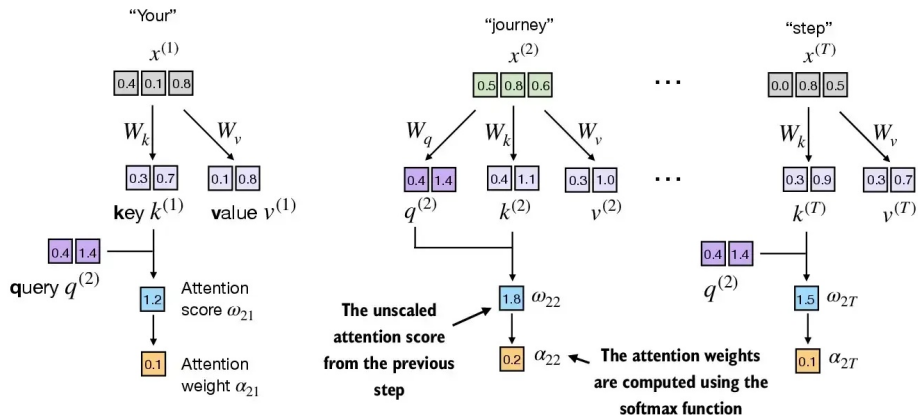
Self-attention Step-by-Step



© 2024 Sebastian Raschka

Source

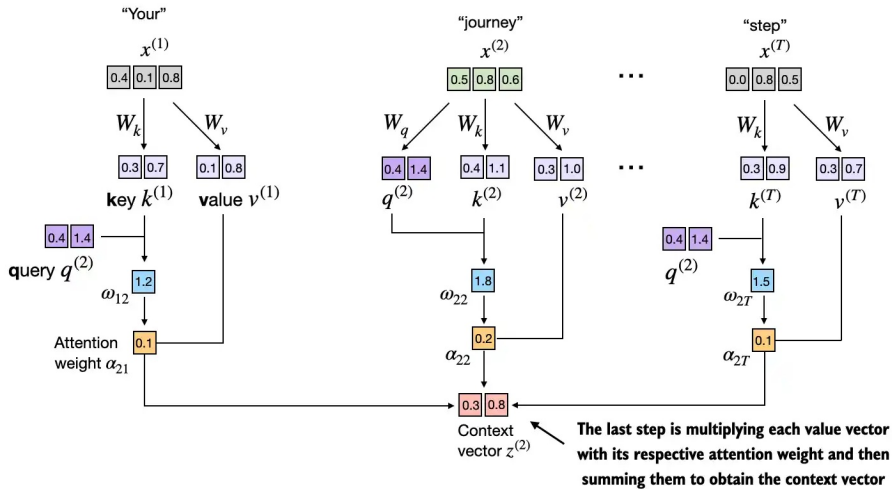
Self-attention Step-by-Step



© 2024 Sebastian Raschka

Source

Self-attention Step-by-Step



© 2024 Sebastian Raschka

Self-attention - Mathematical Formulation

Self-attention is a function $f : \mathbf{X} \mapsto \mathbf{Z}$ with notation:

- ▶ Input matrix $\mathbf{X} \in \mathbb{R}^{l \times d_{in}}$ and output/context matrix $\mathbf{Z} \in \mathbb{R}^{l \times d_{out}}$
- ▶ Sequence length l
- ▶ Input features (per element, e.g. token) d_{in} and output features d_{out} (often $d_{in} = d_{out}$)
- ▶ Trainable parameters: $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d_{in} \times d_{out}}$

Computation steps

- ▶ Query matrix $\mathbf{Q} = \mathbf{X} \mathbf{W}_q$
- ▶ Key matrix $\mathbf{K} = \mathbf{X} \mathbf{W}_k$
- ▶ Value matrix $\mathbf{V} = \mathbf{X} \mathbf{W}_v$
- ▶ Attention score matrix (normalized by dimension) from query and key $\mathbf{S} = \frac{\mathbf{Q} \mathbf{K}^T}{\sqrt{d_{out}}}$
- ▶ Attention matrix (normalization of S) $\mathbf{A}_{i,j} = \text{softmax}(\mathbf{S}_i) = \frac{e^{\mathbf{S}_{i,j}}}{\sum_{colc=1}^l e^{\mathbf{S}_{i,c}}}$ with $\mathbf{A}_{i,j}$ as element of \mathbf{A} in row i , column j .
- ▶ Output (context) matrix: $\mathbf{Z} = \mathbf{A} \mathbf{V}$

If not noted differently its standard matrix product.

Why is Self-Attention so Useful?

Self-attention is a parametric model for within-sequence interactions that works well in practice.

Why is Self-Attention so Useful?

Self-attention is a parametric model for within-sequence interactions that works well in practice.

- ▶ Flexible interactions over time: Without being a *true* time-series model like recurrent neural networks, all sequence elements can interact even when far apart.

Why is Self-Attention so Useful?

Self-attention is a parametric model for within-sequence interactions that works well in practice.

- ▶ Flexible interactions over time: Without being a *true* time-series model like recurrent neural networks, all sequence elements can interact even when far apart.
- ▶ **Parallel processing**: Parallel computation of the entire sequence is possible, unlike recurrent neural networks, which require sequential processing.

Why is Self-Attention so Useful?

Self-attention is a parametric model for within-sequence interactions that works well in practice.

- ▶ Flexible interactions over time: Without being a *true* time-series model like recurrent neural networks, all sequence elements can interact even when far apart.
- ▶ **Parallel processing**: Parallel computation of the entire sequence is possible, unlike recurrent neural networks, which require sequential processing.
- ▶ Parameter efficiency: More parameter-efficient compared to, for example, a fully connected feed-forward neural network with inputs of dimension $l \times d$. The parameter count does not depend on sequence length l .

Why is Self-Attention so Useful?

Self-attention is a parametric model for within-sequence interactions that works well in practice.

- ▶ Flexible interactions over time: Without being a *true* time-series model like recurrent neural networks, all sequence elements can interact even when far apart.
- ▶ **Parallel processing**: Parallel computation of the entire sequence is possible, unlike recurrent neural networks, which require sequential processing.
- ▶ Parameter efficiency: More parameter-efficient compared to, for example, a fully connected feed-forward neural network with inputs of dimension $l \times d$. The parameter count does not depend on sequence length l .

Now, let's **combine Self-Attention** with **feed-forward neural networks** to build our language modeling architecture: the **Transformer**.

Self-attention and Transformers

Transformers

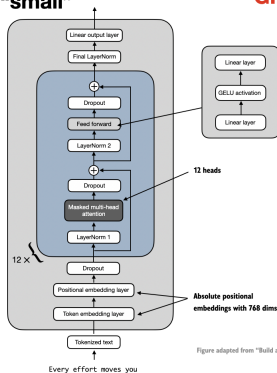
Which Transformer are we talking about?



this tool.

Example transformer architecture - GPT2

GPT-2 “small”



GPT-2 “large”

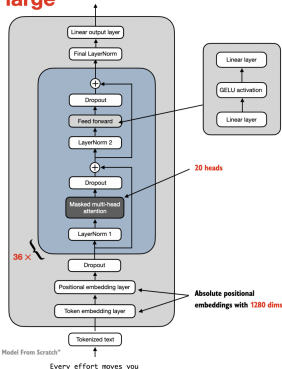
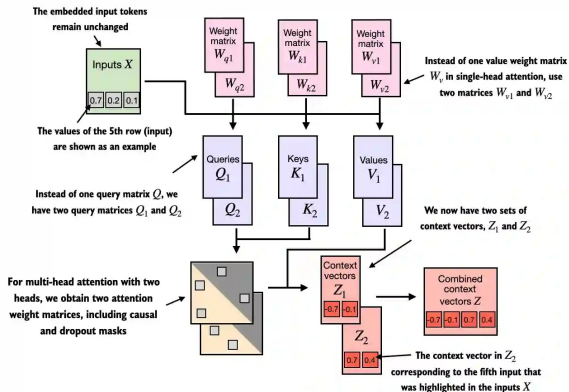


Figure adapted from “Build a Large Language Model From Scratch”

GPT2 model. [Source](#)

Multi-head Attention

Many attention mechanisms in parallel to allow to **pay attention to multiple concepts / meanings** simultaneously. Combined using a linear regression / layer.



Multi-head-attention Source

Gradient Flow

Gradient flow: the ability to consistently receive reliable gradient information at each part of the network to train parameters (even if very deep).

- ▶ Good flow: Most **activations are close to the sensitive region** of the non-linear loss function (e.g. *Relu* or *Sigmoid* around 0) and outputs are not large.
- ▶ Bad flow: Large variance and non-zero mean of activations lead to **vanishing** or **exploding gradients** (main issue of recurrent neural networks).

Transformer Tricks to Improve Gradient Flow

Transformers add components to improve gradient flow:

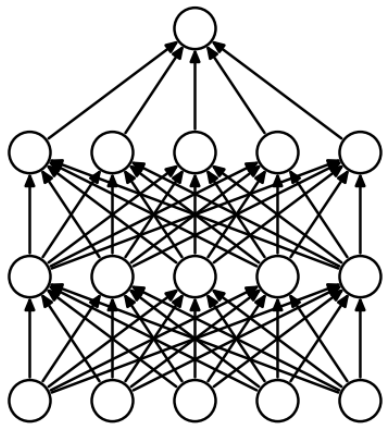
- ▶ **Residual/skip connections:** Add inputs of previous layers to output of current layer to allow.
- ▶ **Layer normalization**

$$\hat{\mathbf{x}}_t = \frac{\mathbf{x}_t - \mu_t}{\sqrt{\sigma_t^2 + \epsilon}} \odot \gamma + \beta$$

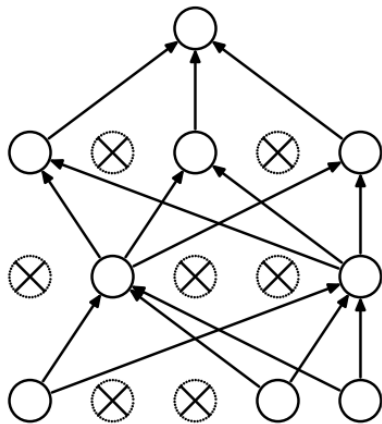
- ▶ $\gamma \in \mathbb{R}^d$ are learned weights,
- ▶ $\beta \in \mathbb{R}^d$ is the learned bias,
- ▶ $\hat{\mathbf{x}}_t \in \mathbb{R}^d$ is the normalized output for token (input) t ,
- ▶ $\mathbf{x}_t \in \mathbb{R}^d$ is the input vector for token (input) t ,
- ▶ $\mu_t \in \mathbb{R}$ is the mean of the input activations for token t over d features.
- ▶ $\sigma_t^2 \in \mathbb{R}$ is the variance of the input activations for token t ,
- ▶ ϵ is a small constant added for numerical stability,
- ▶ \odot denotes element-wise multiplication.

Transformer Tricks to Improve Gradient Flow (cont'd)

Dropout means randomly setting some outputs to 0 during training (regularization).



(a) Standard Neural Net



(b) After applying dropout.

Transformers

A transformer is a function $f : \mathbf{X}_{l \times d_{in}} \mapsto \mathbf{Z}_{l \times d_{out}}$ to transform embedding spaces.

Transformers

A transformer is a function $f : \mathbf{X}_{l \times d_{in}} \mapsto \mathbf{Z}_{l \times d_{out}}$ to transform embedding spaces.

- ▶ Original architecture introduced in [Vaswani et al. \(2023\)](#).
- ▶ Main component: k **transformer blocks** which are built from:
 - ▶ Multi-head self-attention with n heads.
 - ▶ Multilayer perceptron (MLP): 2 layers, with *fan-out* (e.g. $d \times 4$ neurons) and fan-in (back to d neurons).
 - ▶ Layer normalization
 - ▶ Residual connections
 - ▶ Dropout and attention dropout.
- ▶ **Same MLP is applied for each token**, on input dimension d (not $l \times d$!).
- ▶ Interactions only modelled by multi-head self-attention (*few* parameters).

Transformers (cont'd)

A transformer is a function $f : \mathbf{X}_{l \times d_{in}} \mapsto \mathbf{Z}_{l \times d_{out}}$ to transform embedding spaces.

- ▶ Input goes **sequentially** through each transformer block and is **passed to next block** (passed to next block). Always maps back to the **same dimensionality** same dimensionality (by choice of $d_{in} = d_{out}$, but could be fully flexible).
- ▶ Parameters depend on d , number of attention heads in each multi-head attention mechanism, MLPs dimension. E.g. *BERT* with 110/340 Million, *Llama* with 65 Billion, *GPT4* with 1.8 Trillion.
- ▶ **Parallel processing** for entire sequence at once possible.

Transformers (cont'd)

A transformer is a function $f : \mathbf{X}_{l \times d_{in}} \mapsto \mathbf{Z}_{l \times d_{out}}$ to transform embedding spaces.

- ▶ Input goes **sequentially** through each transformer block and is **passed to next block** (passed to next block). Always maps back to the **same dimensionality** same dimensionality (by choice of $d_{in} = d_{out}$, but could be fully flexible).
- ▶ Parameters depend on d , number of attention heads in each multi-head attention mechanism, MLPs dimension. E.g. *BERT* with 110/340 Million, *Llama* with 65 Billion, *GPT4* with 1.8 Trillion.
- ▶ **Parallel processing** for entire sequence at once possible.
- ▶ Architecture variants
 - ▶ **Encoder-only** models (e.g., *BERT*) aim to (only) find valuable latent text representations (but could also generate text).
 - ▶ **Decoder-only** models (e.g., *GPT*) aim to do autoregressive text generation (but also has valuable latent text representations).
 - ▶ **Encoder-decoder** models (e.g., *BART*) do both.

[Link to an amazing visualization for attention and the GPT-2 model architecture.](#)

Configuration of GPT-2

```
# init of GPT2
class GPT2Config(PretrainedConfig):
    def __init__(
        self,
        vocab_size=50257, # vocab size
        n_positions=1024, # max sequence length
        n_embd=768, # dim hidden representation
        n_layer=12, # number blocks
        n_head=12, # number of heads in multihead attention
        n_inner=None, # Default 4 times n_embd. Dimensionality of the inner feed-forward layers.
        activation_function="gelu_new",
        resid_pdrop=0.1, # dropout probability for all fully connected layers in the embeddings, encoder, and pooler
        embd_pdrop=0.1, # dropout ratio for the embeddings
        attn_pdrop=0.1, # dropout ratio for the attention
        layer_norm_epsilon=1e-5, # epsilon to use in the layer normalization layers
        initializer_range=0.02, # standard deviation for initializing all weight matrices
        scale_attn_weights=True, # Scale attention weights by dividing by sqrt(hidden_size).
    ):
        self
```

GPT-2 model configuration file that defines architecture from [Huggingface model](#). GPT-2 model configuration file that defines architecture from [Huggingface model](#).

Questions ?

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, July 2023.