



GPT-2

Generative Pre-trained Transformer 2


Transformers for Language Modeling


- The original transformer model is made up of an encoder and decoder – each is a stack of what we can call transformer blocks. That architecture was appropriate because the model tackled machine translation – a problem where encoder-decoder architectures have been successful in the past.



One difference From Bert

- The GPT-2 is built using transformer decoder blocks. BERT, on the other hand, uses transformer encoder blocks. We will examine the difference in a following section. But one key difference between the two is that GPT2, like traditional language models, outputs one token at a time. Let's for example prompt a well-trained GPT-2 to recite the first law of robotics:

- 
- The way these models actually work is that after each token is produced, that token is added to the sequence of inputs. And that new sequence becomes the input to the model in its next step. This is an idea called “auto-regression”. This is one of the ideas that made RNNs unreasonably effective.

- 
- The GPT2, and some later models like TransformerXL and XLNet are auto-regressive in nature. BERT is not. That is a trade off. In losing auto-regression, BERT gained the ability to incorporate the context on both sides of a word to gain better results. XLNet brings back autoregression while finding an alternative way to incorporate the context on both sides.

The Evolution of the Transformer Block

- Two types of transformer blocks
- The Encode Block
- The Decoder Block




The Encoder Block

An encoder block from the original transformer paper can take inputs up until a certain max sequence length (e.g. 512 tokens). It's okay if an input sequence is shorter than this limit, we can just pad the rest of the sequence.



The Decoder Block

- there's the decoder block which has a small architectural variation from the encoder block – a layer to allow it to pay attention to specific segments from the encoder:

- 
- **Self-attention** is an attention mechanism that relates different positions of a single sequence in order to compute a representation of the sequence. Allowing the model to weigh the importance of different positions of the input sequence to compute a representation of each position
 - **Masked self-attention** is a variant of self-attention that allows a model to attend to all positions in the input sequence except for those that come after the current position. This is useful when training a model to generate text one word at a time, as it prevents the model from “cheating” by looking ahead at words it hasn’t generated yet