Transformers: Attention is all You Need

CuriosityDeck · Follow

Published in Python in Plain English · 5 min read · Nov 20, 2022

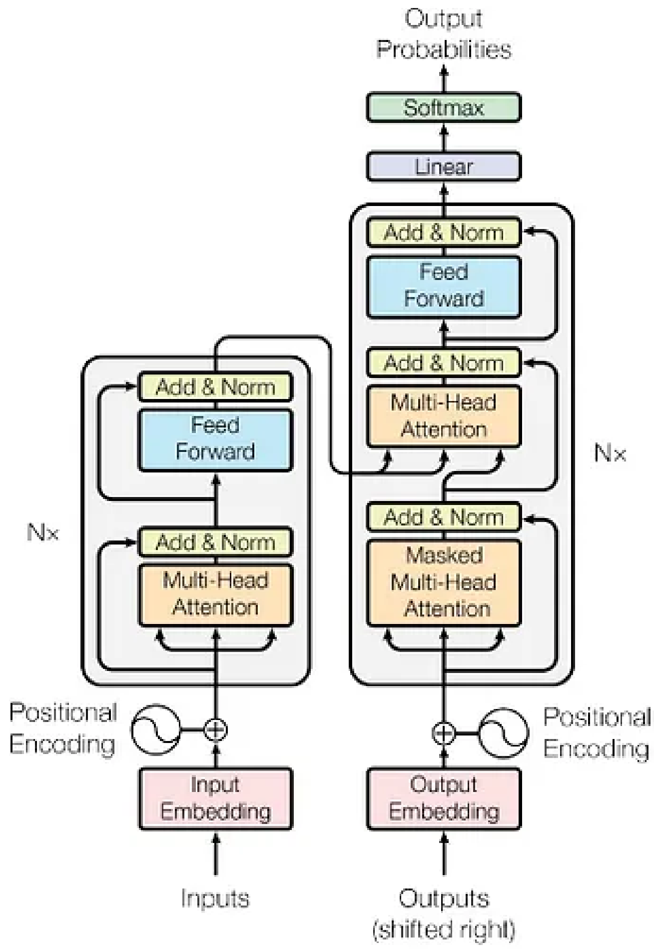
Introduction

In one of the previous blogs, we discussed LSTMs and their structures. However, they are slow and need the inputs to be passed sequentially. Because today's GPUs are designed for parallel computing, these sequence data must be parallelized. This is when the transformers shook the natural language processing world by storm. They are now used in many applications like machine language translation, conversational chatbots, and better search results. They are the foundation for well-known transformer models like BERT, GPT-2, and GPT-3. This all started with one paper- ***Attention is all you need.*** Let us understand the attention mechanism which lays the foundation for these models.

The generated sequence by a model depends on its ability to reference or attend to words. This is true in the case of RNNs and LSTMs, but these models have a limited window. Whereas given enough compute resources, an ideal attention mechanism will have an infinite window to reference from and will be capable of using the entire context of the story while generating the text.

Transformer-Architecture

Transformers are also an encoder-decoder model, the encoder maps an input sequence into an abstract continuous representation that holds all the learned information of that input. The decoder then takes that continuous representation and step by step generates a single output while also being fed the previous outputs.



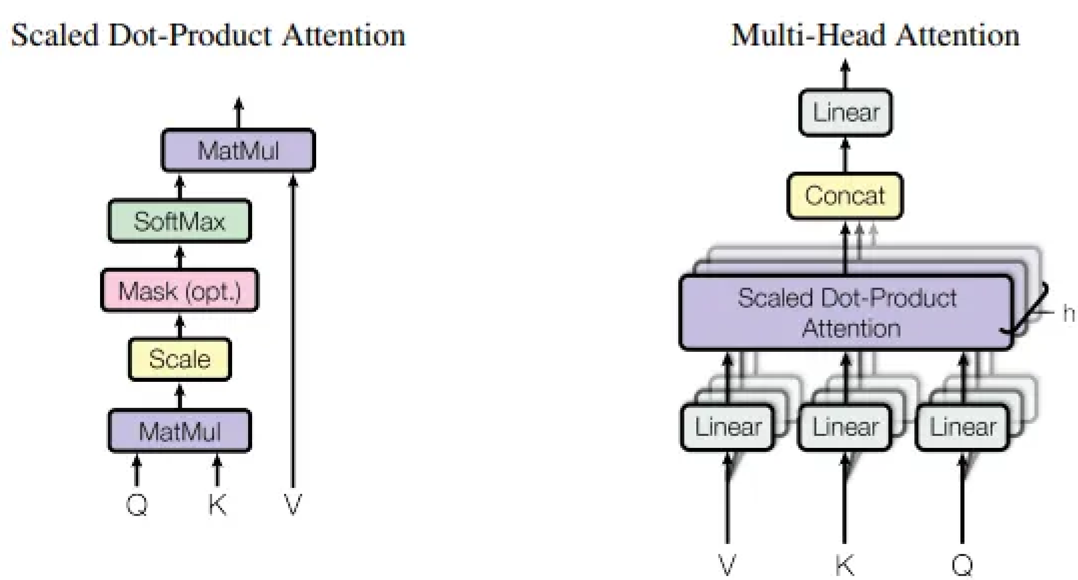
The Transformer-Model Architecture

Let's take a step-by-step look at how the transformer network works. The first step is to feed our data into a word embedding layer. A word is mapped to a point in space where words with similar meanings are physically closer to each other. We can either train the embedding space or use pre-trained embedding space in such a way that similar words are closer in the space. The positional information is added to input embedding using positional encoding with help of sine and cosine functions.

We pass this into the encoder layer. The encoder layer maps all the input sequences into a representation that contains the sequence's learned information. It consists of multi-headed attention and a fully connected network. There are also residual connections around each of the two sub-modules followed by a layer normalization.

Multi-headed Attention Layer

Multi-headed attention in the encoder involves a specific attention mechanism called self-attention. It says how relevant a selected word is to other words in the sequence. To achieve self-attention, we feed the input into three distinct fully connected layers to create the query, key, and value vectors. The dot product of queries and keys produces a score matrix that determines the focus of a word on other words. Then the scores get scaled down by dividing the value by the square root of the dimension of the queries and the keys. Softmax is applied to make probability values between 0 and 1. Then you take the attention weights and multiply them by your value vector to get an output vector. The score values will decide the words to be filtered. You feed the output vector into a linear layer to process.



Attention layer

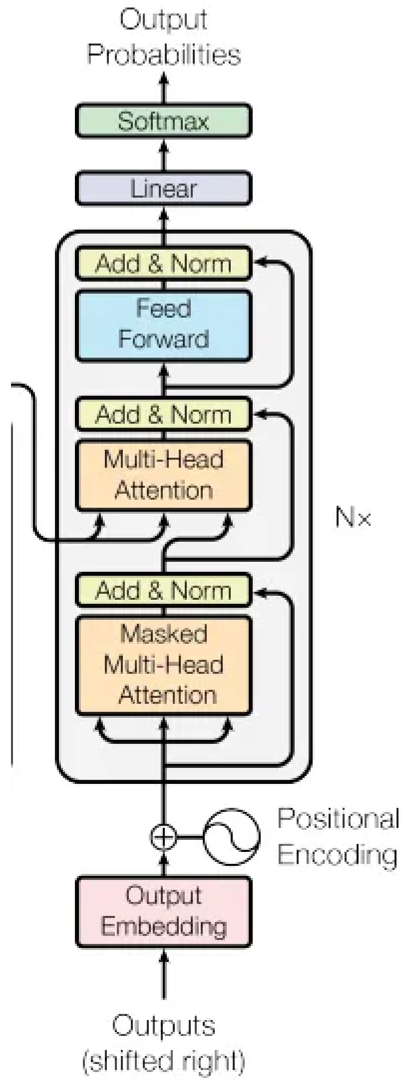
For multi-headed attention computation, we have multiple query, key, and value vectors. These vectors calculate the self-attention process individually and generate different vectors. The output vector produced by each head gets concatenated into a single vector before going through the final linear layer.

A residual connection is used to add the output vector of the multi-headed attention output vector to the original input. This helps the network train by allowing gradients to flow through the networks directly. Then the layer

giving it a richer representation.

The Decoder

This encoded vector with attention information will help the decoder focus on the appropriate words. The decoder's job is to generate text sequences. The decoder is autoregressive and has a similar structure as the encoder. It takes in the list of previous outputs as inputs as well as the encoder outputs that contain the attention information from the input. The decoder stops decoding when it generates an end token as an output.



The decoder

In a decoder, the positional encoded input embedding passes through the attention layer giving us the scores. Since the decoder is autoregressive and generates the sequence word by word, the attention layer is slightly different. We mask the future word to prevent it from conditioning to future tokens. The output of the first attention is a mask output vector with information on how the model should attend to the decoder's inputs. For the second attention layer, the encoder's output is the queries and the keys and the first multi-headed attention layer outputs are the values. This process matches the encoder's input to the decoder's input, allowing the decoder to decide which encoder input is relevant to put focus on. The output of the second multi-headed attention goes through a pointwise feed-forward layer for further processing. The output of the final pointwise feed-forward layer goes through a classifier. The output of the classifier then gets fed into a softmax layer. We take the index of the highest probability score and that equals our predicted word. The decoder then takes the output and adds it to the list of decoder inputs and continues decoding again until the end token is predicted.

Transformers leverage the power of the attention mechanism to make better predictions. Recurrent known networks try to achieve similar things, but because they suffer from short-term memory, transformers are usually better, especially if we want to encode or generate longer sequences. Because of the transformer architecture, the natural language processing industry can now achieve unprecedented results. Next, let's discuss BERT by google which can be used as a pre-train model for common NLP tasks.

References:

Paper: https://arxiv.org/pdf/1706.03762.pdf

Youtube1: https://www.youtube.com/watch?v=4Bdc55j80l8

Youtube2: https://www.youtube.com/watch?v=TQQlZhbC5ps&list=PLTl9hO2Oobd_bzXUpzKMKA3liq2kj6LfE

blog: https://jalammar.github.io/illustrated-transformer/

Transformers NLP Bert Gpt 3 Artificial Intelligence

Written by CuriosityDeck

27 Followers · Writer for Python in Plain English

Follow

More from CuriosityDeck and Python in Plain English

CuriosityDeck

An Introduction to SORT a Tracking Algoirthm

Introduction

3 min read · May 11, 2023

27

Miner Of Ideas in Python in Plain English

Stop Doing It in Python

Once Taylor Swift said “You will never be able to find happiness if you stay attached to the...

· 7 min read · Jan 25, 2024

1K 26

Jesús Lagares in Python in Plain English

Why Are There So Many Programmers Who Cannot Find...

Why code sniffing is no longer worth obtaining $100.000/y salaries.

· 4 min read · Dec 6, 2023

2.3K 52

CuriosityDeck

Neural Network Quantization: Types

Different types of quantizations and their features

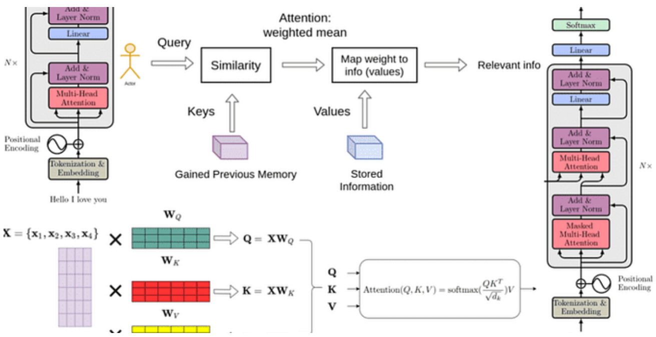
2 min read · Jan 21, 2024

8

See all from CuriosityDeck

See all from Python in Plain English

Recommended from Medium



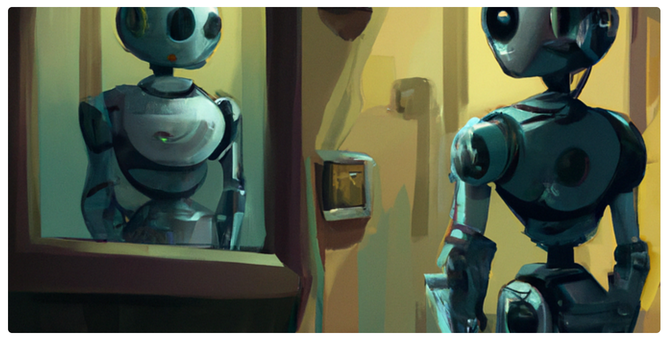
Amanatullah

Transformer Architecture explained

Transformers are a new development in machine learning that have been making a lo…

10 min read · Sep 1, 2023

304 2



Thomas van Dongen in Towards Data Science

Demystifying efficient self-attention

A practical overview

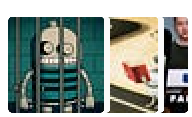
20 min read · Nov 7, 2022

623 2

Lists

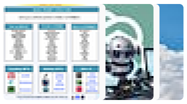
Natural Language Processing

1196 stories · 668 saves



AI Regulation

6 stories · 317 saves

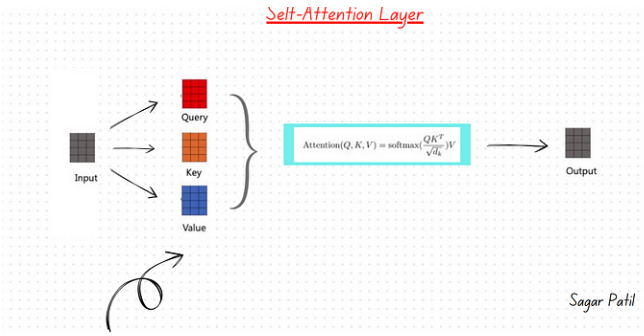


ChatGPT

21 stories · 464 saves

What is ChatGPT?

9 stories · 293 saves



Sagar Patil

Attention Mechanism in the Transformers

In the world of natural language processing and machine learning, few innovations have...

4 min read · Sep 25, 2023

52

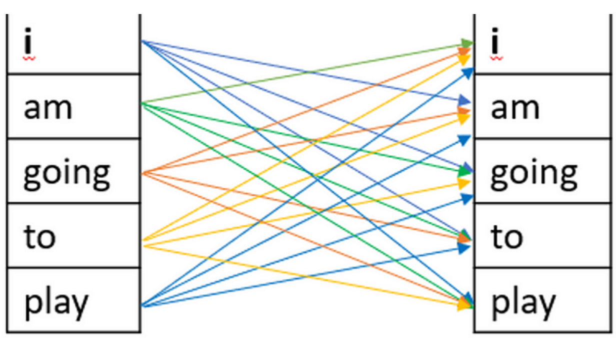
Aneesha B Soman

Transformers Part 4: Building a conversational chatbot using...

Before we build a conversational chatbot there are various features we need to...

4 min read · Sep 6, 2023

2

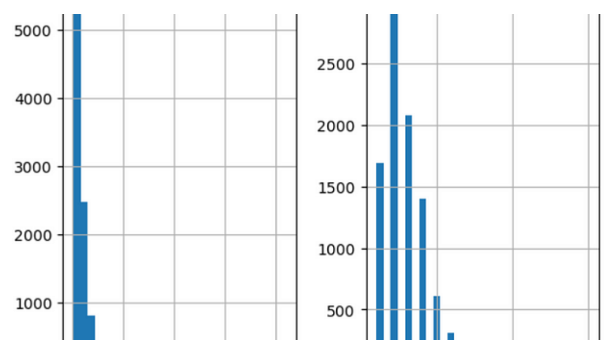


Lovelyn David

Self-Attention: A step-by-step guide to calculating the context...

Introduction

7 min read · Oct 16, 2023



Koyela Chakrabarti

Abstractive text summarization with Transformer model using...

Text summarization is one of the many applications of natural language processing....

21 min read · Oct 18, 2023

4

7

See more recommendations