Presentation on GPT

How does Chat GPT actually Work?

Tanishq Sadanala,
Dr.Zhongmei yao

Contents

01 What is ChatGPT

02 History of GPT

03 Working Motive

04 Other Models using GPT

05 Limitations

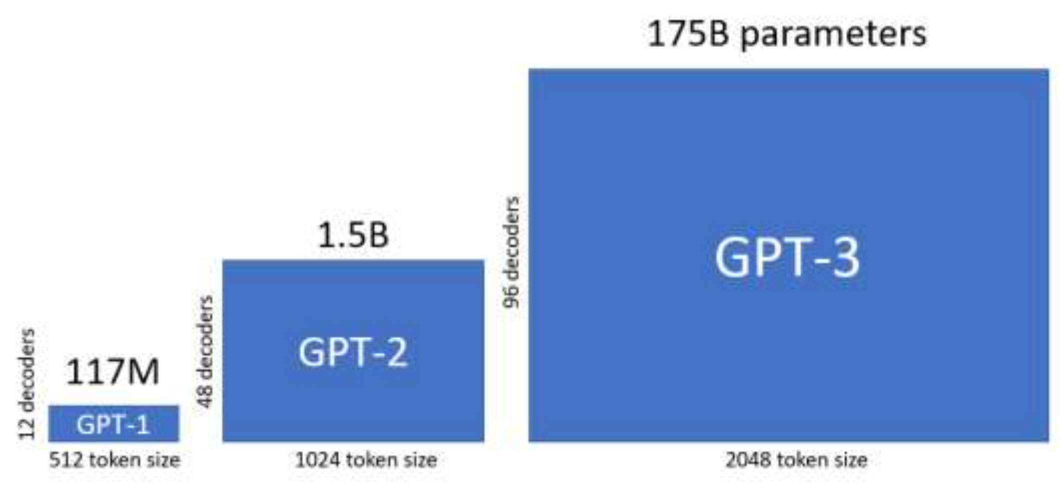
PORTFOLIO 2023

01

What is ChatGPT?

Chat GPT

- ChatGPT is a conversational AI developed by OpenAI.
- Based on the GPT (Generative Pre-trained Transformer) architecture.
- Designed to engage in natural language conversations with users, providing responses that are contextually relevant and coherent.
- ChatGPT has been trained on a vast amount of text data from the internet, allowing it to generate human-like responses to a wide range of prompts and questions



02

History that led to ChatGPT

HOW IT ALL STARTED?

GPT-1 (2018-2019):

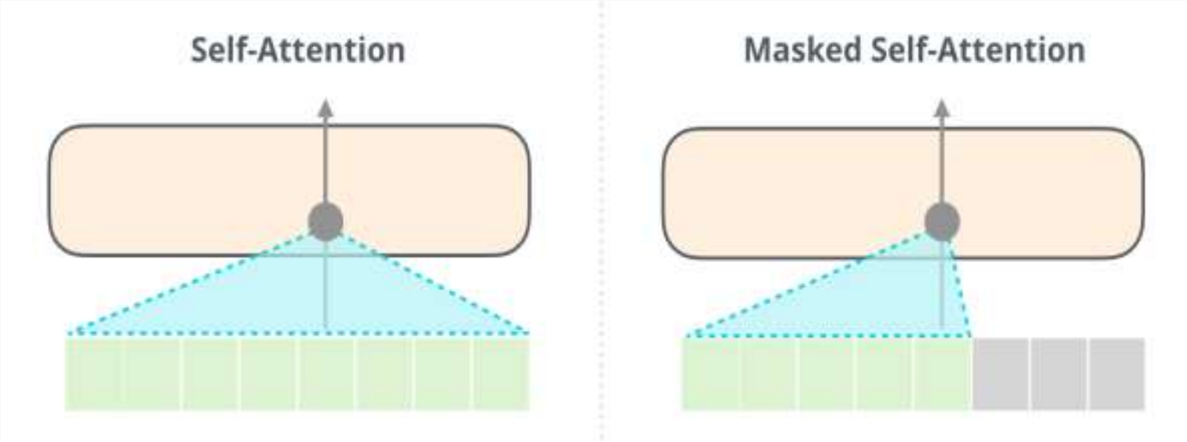
- OpenAI introduced GPT-1 [1] in June 2018, marking a significant advancement in natural language processing. It utilized a transformer architecture and was trained on a vast corpus of internet text.

Release of GPT-2 (2019):

- OpenAI released the full version of GPT-2 [2] in November 2019.
- A staged release plan was implemented to address concerns about potential misuse.
- Researchers and developers began exploring various applications of GPT-2, including text generation, chatbots, content creation, and language translation.

ChatGPT-3, 3.5, 4:

- Following the release of GPT-3 [3] , OpenAI and other research teams continued to advance and refine transformer-based models for natural language processing.
- These advancements also extended to incorporating capabilities for image and voice inputs.

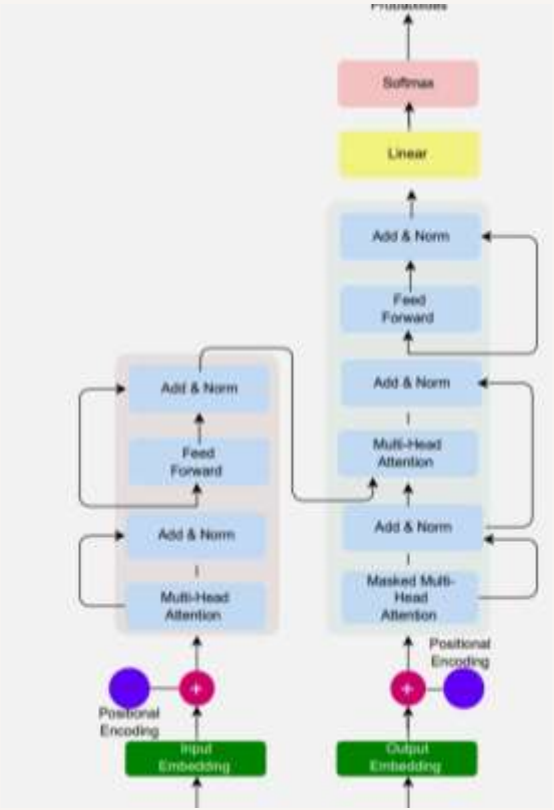


03

Working of GPT

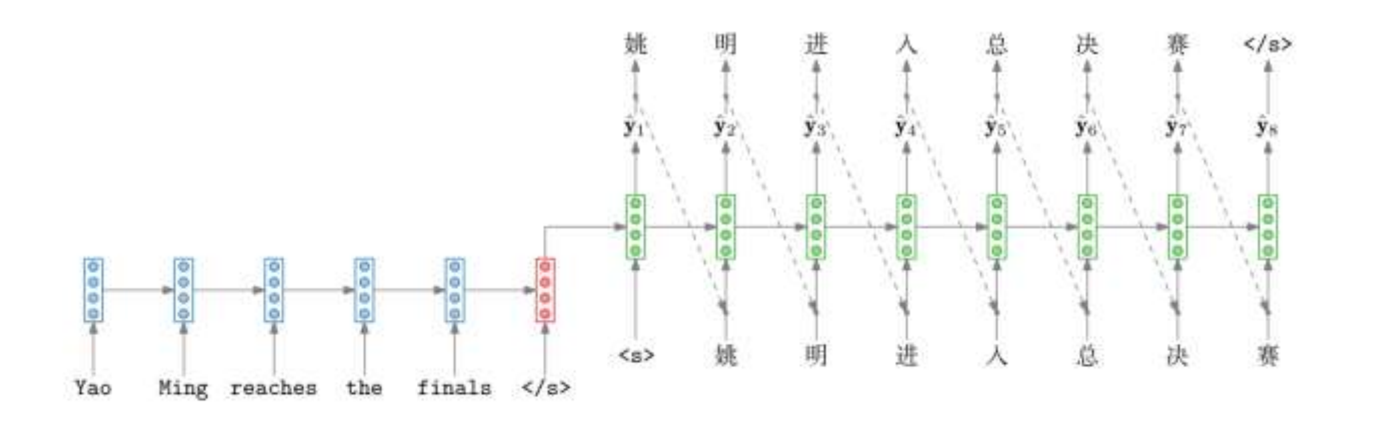
Transformer Architecture

- Even though we don’t know exact source code for the ChatGPT., we do know how it works.
- Most of the heavy things are done through the Transformer model.
- This is first proposed in the paper “Attention is all you need” [4].
- Let us dive deep into this monstrosity and peel block by block on how it works.



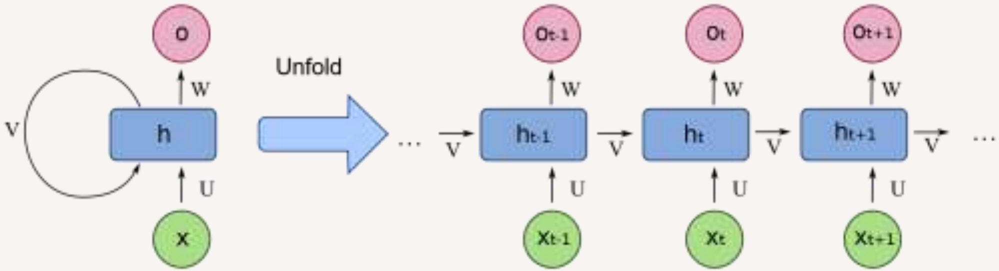
Encoder & Decoder Model

- To understand from the start lets refresh our minds on the previous ways to generate text, predictions and machine translation.
- We will not talk about the vanilla RNN encoder decoder model in detail. but let us view this in a higher picture.



E&D model

- Encoder contains RNN network that takes the input in a sequence and tokenizes the sentence
- It only contains one block and connects to itself.
- But the time steps can be unfolded to get a better visualization.



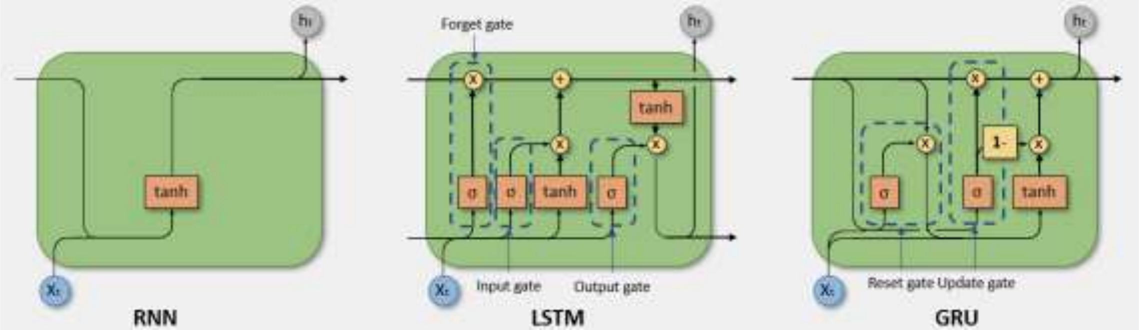
- All we are trying to do is to **minimize the negative log likelihood** to get better predictions

Issues

- The issue arises at the context vector (latent space), where all input sequence information is condensed into one large vector space and sent to the decoder.

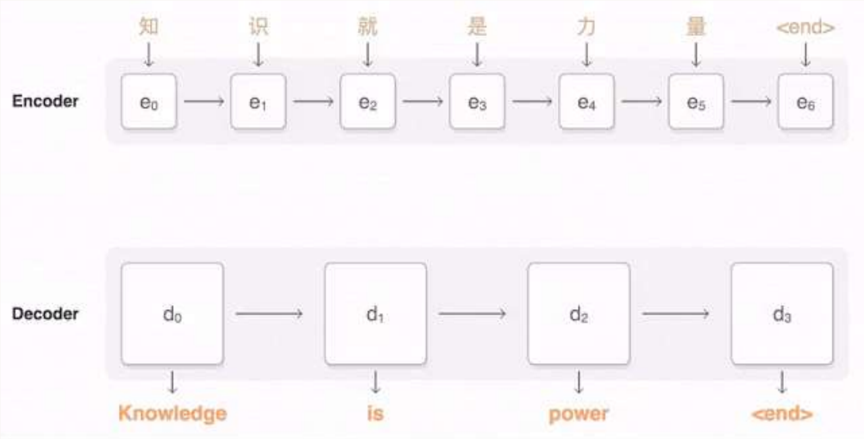
- The expectation is for the model to *retain* this information until the decoding stage, but it *often fails* to do so.

- Various versions of RNN, such as LSTM and GRU, have been developed to address this issue and enable the model to preserve information until the very end.



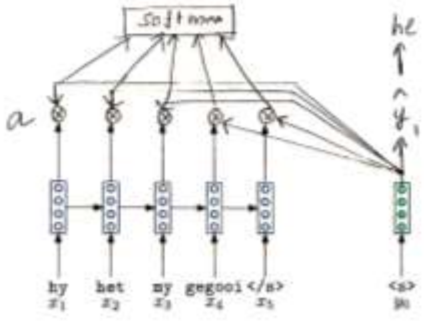
Attention (Encoding)

- Attention is proposed in 2015 (Bengio et. al.)
- Attention focuses on looking all the inputs at each time step.
- In this way we can see the ground truth and make a higher prediction probability.



Forward pass:

the output of $y_0 = (\hat{y}_1)$ can be high for first word, as it has more weightage.



here x_n, y_n are Tensors,

$a_n = (x_n \otimes y_n) \ \exists \ x = 1 \cdots n, \ y = 1 \cdots n$

[The dot product o/p can be $\boxed{-\alpha \text{ to } \alpha}$]

The output of softmax be $\alpha_1, \alpha_2 \cdots \alpha_n$

If α is high, the present decoder Time step is really looking at that time point.

- Encoder hidden states:

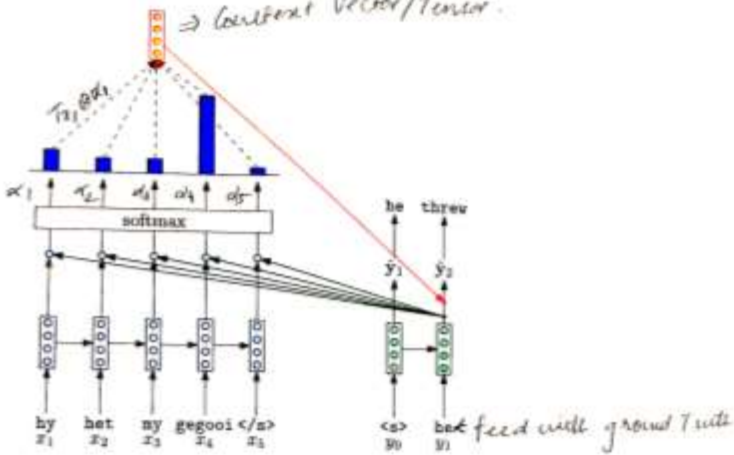
$$\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N \in \mathbb{R}^D$$

- Decoder hidden state at time step t:

$$\mathbf{s}_t \in \mathbb{R}^D$$

- Attention score for encoder hidden state $\mathbf{h}_n$:

$$a(\mathbf{s}_t, \mathbf{h}_n) = \mathbf{s}_t^\top \mathbf{h}_n \in \mathbb{R}$$



- Take the First hidden Representation, scale it to alpha $T_{x_1} \otimes \alpha_1$ ||ly others $T_{x_2} \otimes \alpha_2 \cdots T_n \otimes \alpha_n$
- Sum it $\boxed{\sum_{n=1}^{K} T_{x_n} \otimes \alpha_n}$

The O/p of this is a One Giant Tensor '$\underline{C}$'.
This '$\underline{C}$' is "context vector" (like bottle neck) vector average of all Time steps for curnt decoder Time step.

- Attention weight for encoder hidden state $\mathbf{h}_n$:

$$\alpha(\mathbf{s}_t, \mathbf{h}_n) = \mathrm{softmax}_n\left(a(\mathbf{s}_t, \mathbf{h}_n)\right) = \frac{\exp\{a(\mathbf{s}_t, \mathbf{h}_n)\}}{\sum_{j=1}^{N} \exp\{a(\mathbf{s}_t, \mathbf{h}_j)\}} \in [0, 1]$$

- Context vector at decoder time step t:

$$\mathbf{c}_t = \sum_{n=1}^{N} \alpha(\mathbf{s}_t, \mathbf{h}_n)\mathbf{h}_n \in \mathbb{R}^D$$

- Concatenate:

$$\left[\mathbf{c}_t; \mathbf{s}_t\right] \in \mathbb{R}^{2D}$$

and continue as in the non-attention decoding case, e.g.

$$\hat{\mathbf{y}}_t = \mathrm{softmax}\left(\mathbf{W}_{ho}\left[\mathbf{c}_t; \mathbf{s}_t\right] + \mathbf{b}_o\right)$$

Cont.

- Attention is calculated at each hidden representation and projected onto each input layer, resulting in soft probabilities.

- These probabilities provide high and low weights during training, aiding the machine in understanding what to predict.

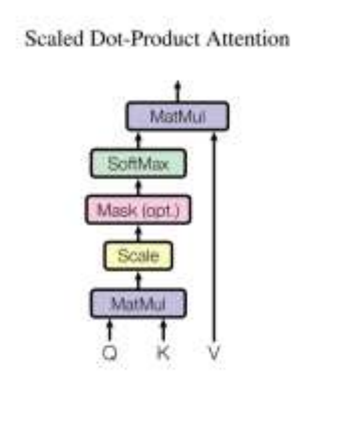
- All context vectors at each time stamp are concatenated into the hidden representation at that time frame to generate predictions or translations.

- This approach leads to significant improvements, as it enables GPU performance by **parallelizing data calculations** similar to feedforward networks, without relying on output data.

- Consequently, large datasets can be efficiently represented in vector forms, facilitating parallelized calculations.

Attention with keys, Queries, Values

- In the paper Attention is all you need., Vaswani et.al has proposed self-attention.



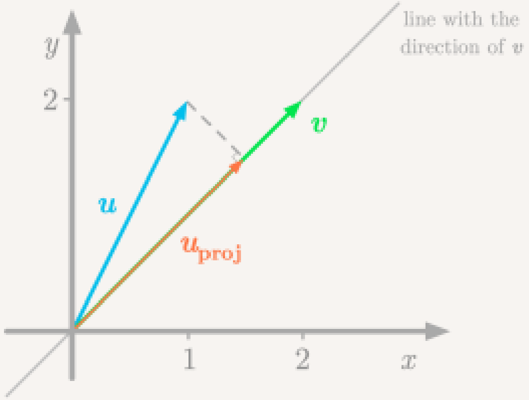
- Queries – Vector from which attention is looking
- Keys – vector at which query looks to compute weights
- Values – Their weighted sum is attention output

Contd.

- Mathematically, represented as :

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

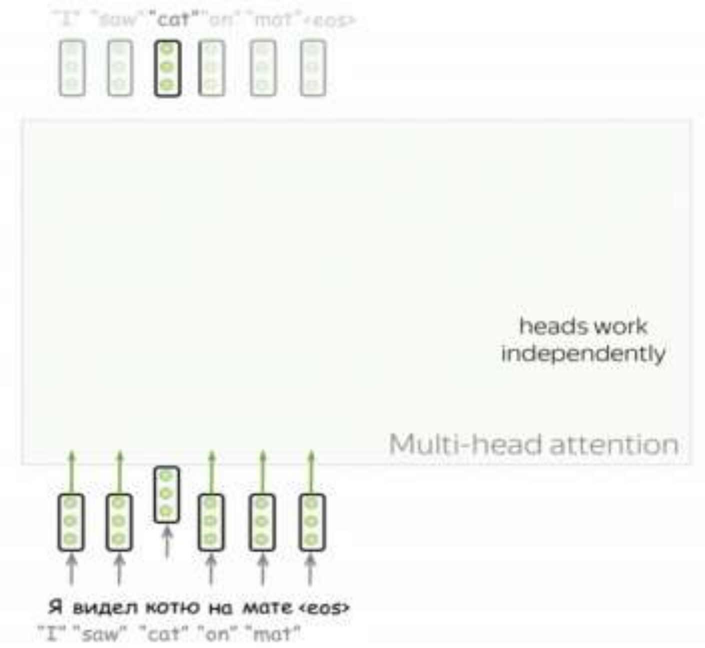
- We do scale dot product for keeping all the vector space/span in same dimension.
- This way we get similarity matrix to high accuracy.



$$u \cdot v = ||u_{\text{proj}}||_2 \cdot ||v||_2$$

Multi-head & masked attention

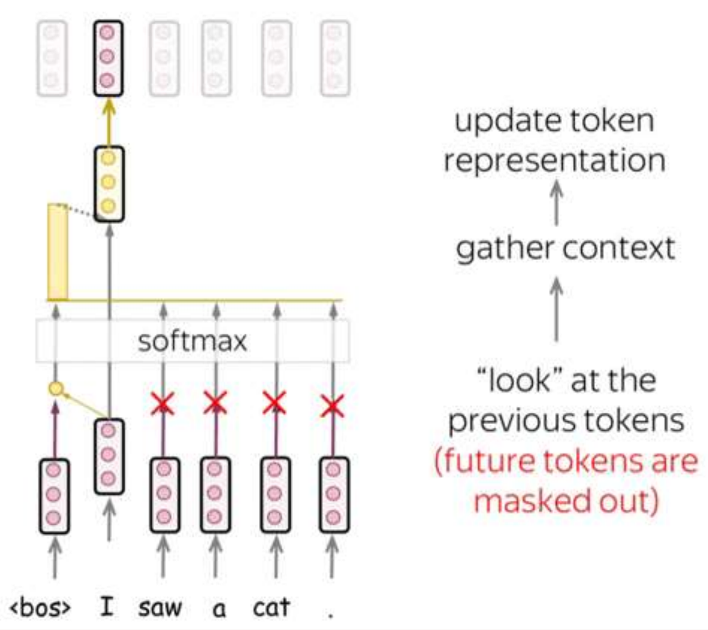
- We have been seeing single Attention layer till now. But, in a sentence, we *might be giving* weights regarding only on certain semantic info or Syntactic info.
- The selection process in these layers is like a black box.
- To understand what the machine picks up, additional layers are added, similar to how CNN adds filters.
- This helps in capturing various features in different filters, bridging the black box gap.
- This is called multi-head attention.



©Lena Voita

Masked Attn.

- Decoder's self-attention allows tokens to "look at previous tokens" during generation.
- Unlike the encoder, the decoder generates one token at a time.
- Masked self-attention prevents the decoder from **looking ahead** during generation.
- During training, reference translations are used to feed the whole target sentence to the decoder without masks.
- The Transformer's computational efficiency, without recurrence, enables processing all tokens at once, making it *faster for training* compared to recurrent models.



©Lena Voita

$$\mathbf{a}_i = \mathrm{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d}} + \mathrm{mask}_i\right)\mathbf{V}_i$$

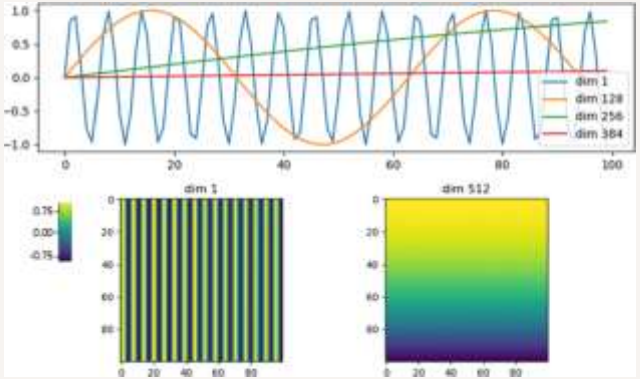
$$\mathrm{mask}_i = \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{otherwise} \end{cases}$$

Positional Encoding

- Positional encodings are essential in Transformers to convey the order of input tokens.
- Two sets of embeddings are used: token embeddings and positional embeddings.
- The input representation of a token is the sum of its token embedding and positional embedding.
- Fixed positional encodings are employed in Transformers, utilizing sinusoidal functions.
- The formula for fixed positional encodings is:

$$\text{positional\_encoding}(p, 2i) = \sin\left(\frac{p}{10000^{2i/d_{\text{model}}}}\right)$$

$$\text{positional\_encoding}(p, 2i+1) = \cos\left(\frac{p}{10000^{2i/d_{\text{model}}}}\right)$$



Residual Connections:

- Gradient Flow: Residual connections aid gradient flow by providing shortcut paths, easing training.
- Deep Stacking: They enable the effective stacking of layers, facilitating deeper architectures.
- Training Enhancement: Residual connections mitigate gradient degradation, enhancing training stability.

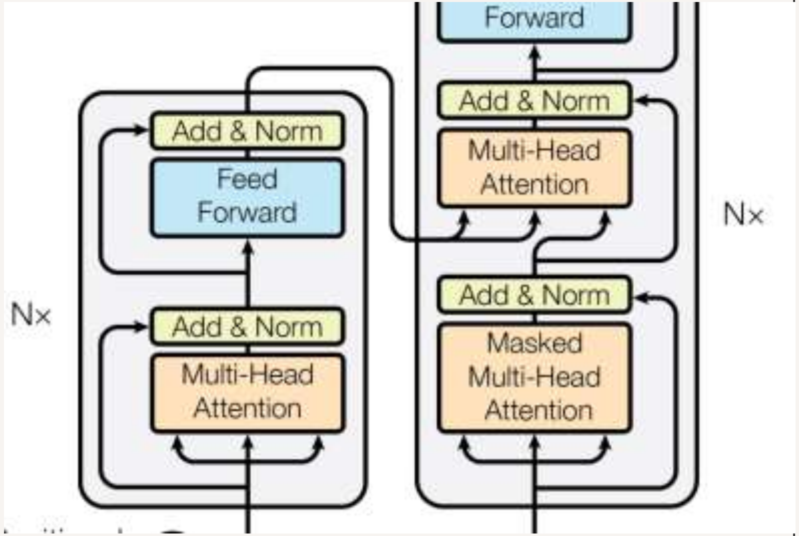
Layer Normalization:

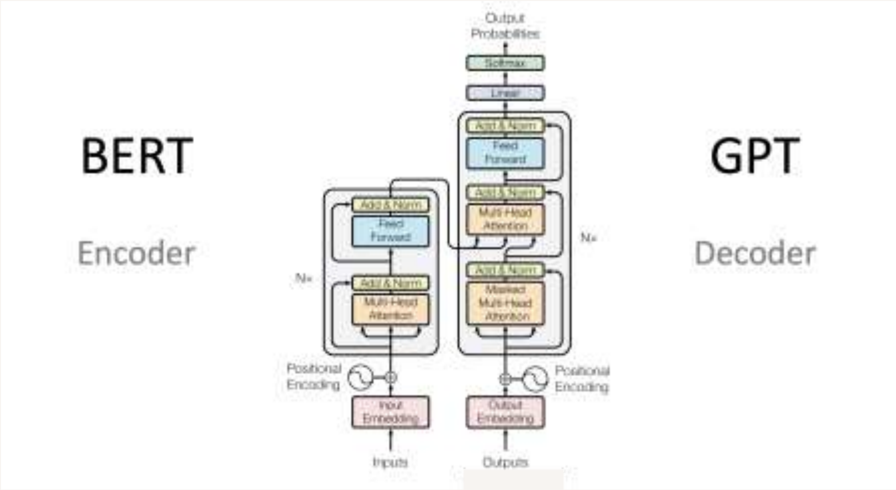
- Batch-wise Regulation: Layer normalization independently normalizes each example in a batch, regulating information flow.
- Stability and Quality: Improves convergence stability and prediction quality by reducing internal covariate shift.
- Trainable Parameters: Includes trainable parameters γ and β for rescaling, enhancing model adaptability.

Cross Attention(Decoding)

- **Keys Extraction**: Extract keys from the encoder outputs for each decoding step.
- **Dot Product**: Perform dot product between the decoder's query and each key.
- **Attention Scores**: Compute attention scores based on the dot products.
- **Weighted Sum**: Calculate a weighted sum of encoder outputs using attention scores.
- **Prediction Output**: Generate prediction output by passing the weighted sum through a linear transformation.

Cross attention

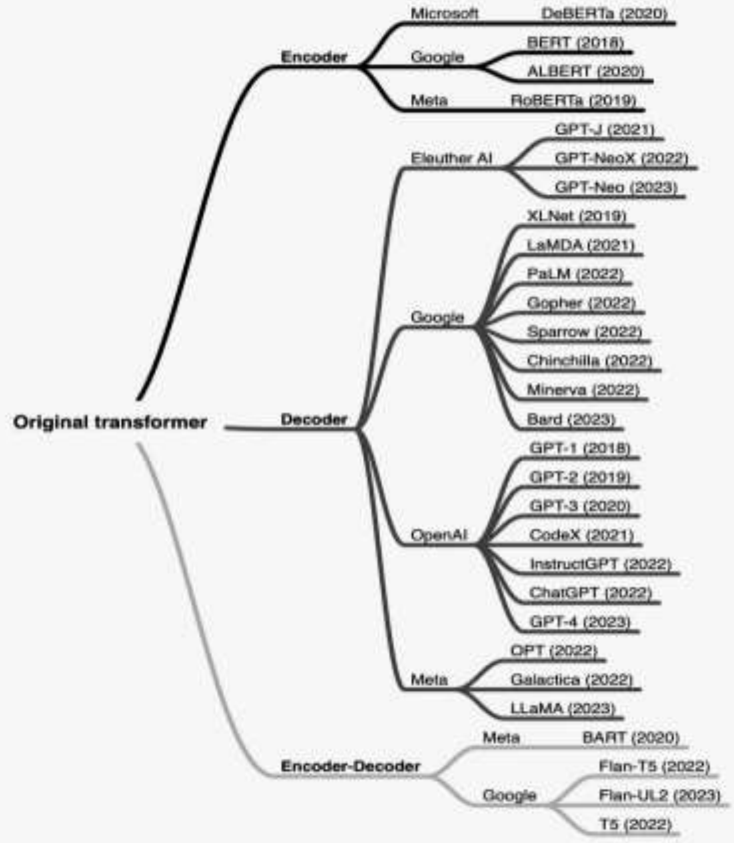




05

Who Uses GPT?

Encoder only / Decoder only?



ChatGPT

When to use What?

- ChatGPT uses BPE(byte pair encoding) and transformer decoding.

- Encoder-only models focus on processing input sequences, suitable for tasks like classification.
- Decoder-only models generate output sequences based on encoded representations, ideal for text generation tasks.

- ChatGPT and similar models use decoder-only architectures for generating contextually relevant responses in conversations.
- Encoder-decoder architecture is utilized in ChatGPT, where input prompts are first encoded, then decoded to generate responses.

05

Limitations of ChatGPT

Limitations

- **Common Sense & Background Knowledge**: Limited understanding of common sense and lacks extensive background knowledge.
- **Emotional Intelligence**: Unable to detect subtle emotional cues or respond appropriately to complex emotions.
- **Performs best with single tasks:** struggles with handling multiple tasks simultaneously.
- **Biases & Limited Knowledge** : Responses may reflect unintentional biases from training data and lack access to recent developments or specialized information, leading to potential accuracy issues.

Conclusion

- GPT operates on transformer architecture with self-attention for text generation.
- ChatGPT, a specialized variant, utilizes a decoder-only model for generating contextually relevant responses in conversations.
- Understanding these models enhances their effectiveness in natural language understanding and generation tasks, improving AI-driven conversational experiences.

REFERNCES

1. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

2. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

3. https://arxiv.org/pdf/2005.14165.pdf

4. https://arxiv.org/abs/1706.03762

Intuitions are heavily relied on :

I. https://lena-voita.github.io/posts.html

II. https://www.kamperh.com/nlp817/