



中国科学技术大学
University of Science and Technology of China

Introduction to LLM

Huiyang Peng

School of management, USTC

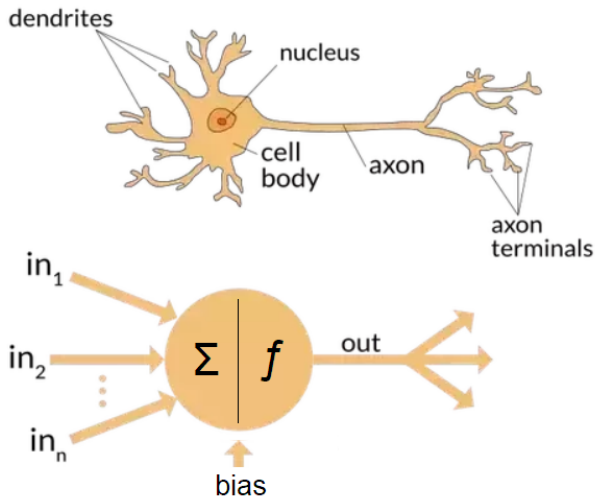
2025 年 3 月 7 日



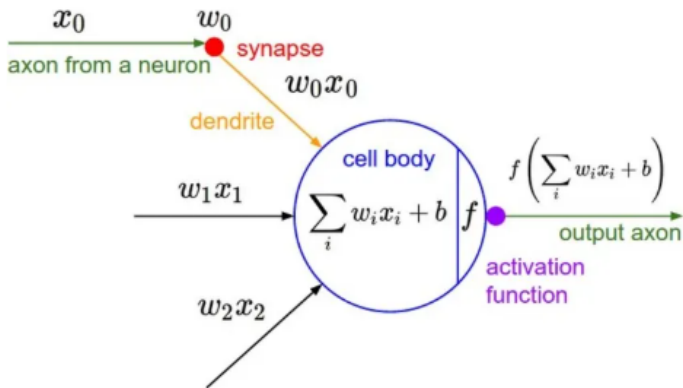
- 1 A review of Deep Learning
- 2 Attention and Transformer
- 3 BERT and GPT
- 4 Reference



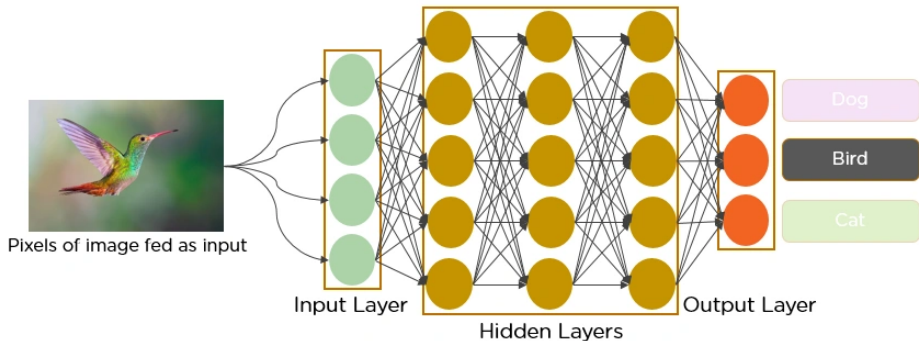
- 1 A review of Deep Learning
- 2 Attention and Transformer
- 3 BERT and GPT
- 4 Reference



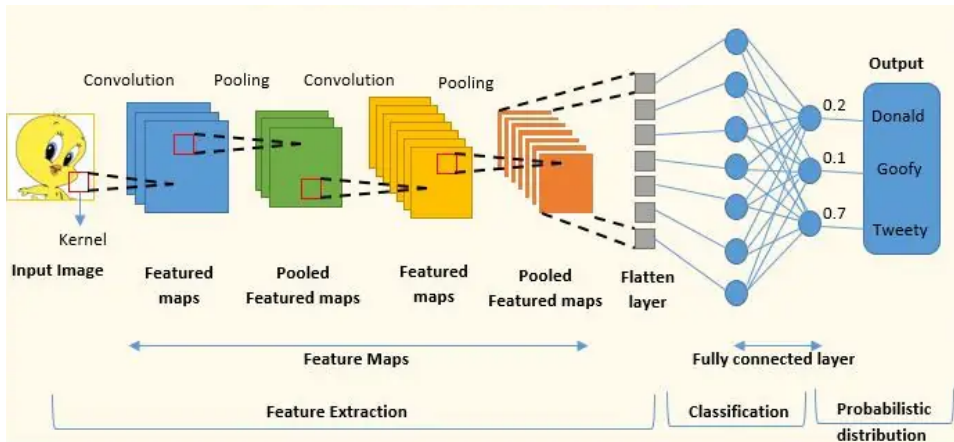
The structure of a single neuron.



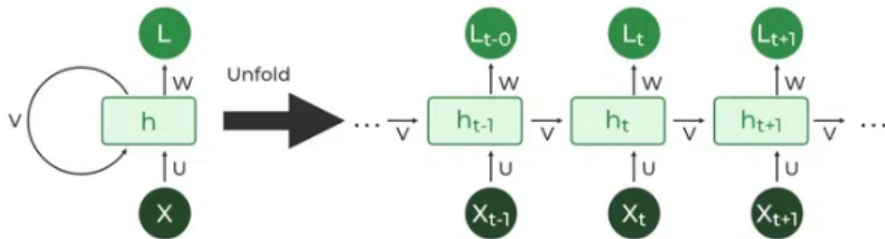
Every great AI is made up of small units with little intelligence.



Convolutional Neural Network(CNN)



Recurrent Neural Network(RNN)



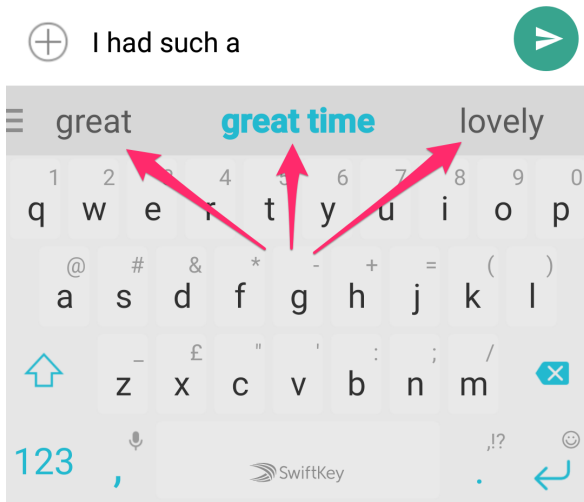
The RNN emerged from the need to process text, which laid the foundation for the emergence of the **Attention** mechanism.

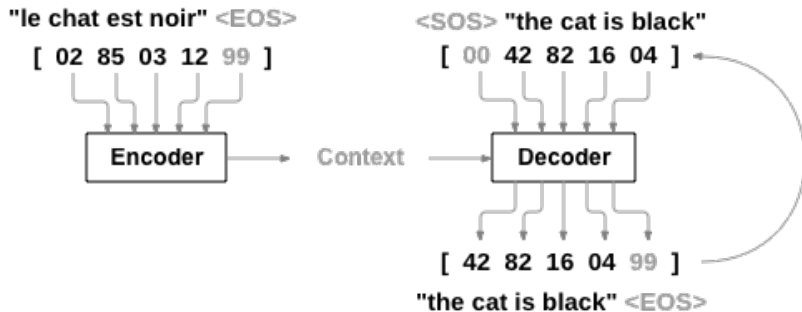


- ▶ 1958: Perceptron
- ▶ 1980s: Multi-layer perceptron
- ▶ 2009: GPU; 2012: CNN
 - ▶ 2015: Image recognition surpassing human-level performance
 - ▶ 2016: Alpha GO beats Lee Sedol
 - ▶ 2016: Speech recognition system as good as humans
 - ▶ 2017: Attention is all you need
- ▶ 2022: ChatGPT
 - ▶ 2023: GPT-4, Claude
 - ▶ 2024: GPT-o1
 - ▶ 2025: Deepseek-r1
 - ▶ ...



- 1 A review of Deep Learning
- 2 Attention and Transformer
- 3 BERT and GPT
- 4 Reference







How next-word prediction makes ChatGPT? **Human Guidance**

Here is a simplified guidance of ChatGPT:

You are a helpful assistant. Always be truthful. If you are unsure, say "I don't know".

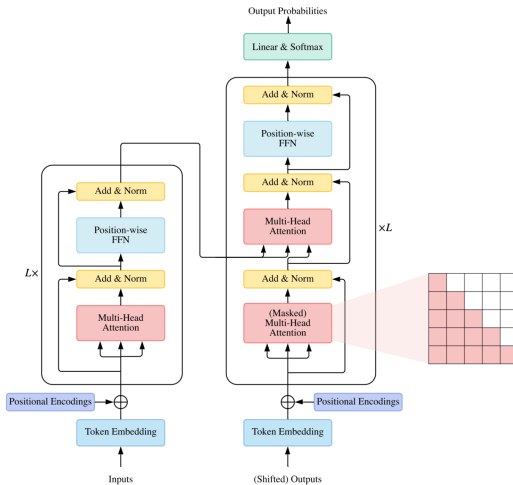
User: {User's input}

Respond:

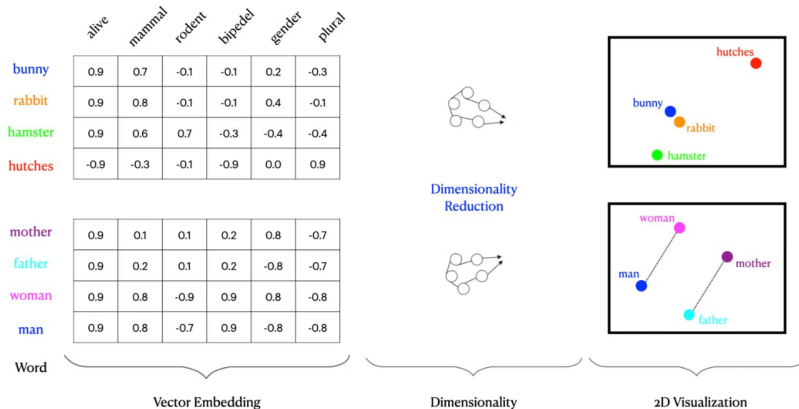
When the LLMs try to predict what's next after "Respond:", that's the answer that we expect.



Let's Start Building A Transformer!



Everything is token in LLM.





- ▶ Make use of the order of the sequence.

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

- ▶ Why chose this function?
- ▶ For any fixed k , PE_{pos+k} is a linear function of PE_{pos} .
- ▶ Let $A_k \in \mathbb{R}^{d_{model} \times d_{model}}$, $A_k(2i, 2i) = A_k(2i+1, 2i+1) = \cos(k/10000^{2i/d_{model}})$,
 $-A_k(2i+1, 2i) = A_k(2i, 2i+1) = \sin(k/10000^{2i/d_{model}})$.
- ▶ Then $PE_{pos+k} = A_k PE_{pos}$.

- ▶ Let's express the above in matrix format.
- ▶ Assume that there're n tokens in the input, after the embedding layer, it produces outputs of dimension $d_{model} = 512$, that is, a embedding matrix $I \in \mathbb{R}^{n \times d_{model}}$, where $I = [\vec{I}_1^T; \vec{I}_2^T; \cdots; \vec{I}_n^T]$.
- ▶ The positional encodings also produce outputs of dimension d_{model} , so the corresponding matrix is $PE \in \mathbb{R}^{n \times d_{model}}$, where $PE = [P\vec{E}_1^T; P\vec{E}_2^T; \cdots; P\vec{E}_n^T]$.
- ▶ Now we add them together to represent both meaning and position, thus

$$E = I + PE$$



Let's recall what is happening when we are reading.
Here are three sentences about "stand":

- ▶ He stood up and walked away.
- ▶ I can't stand the noise.
- ▶ He stands for justice.

In fact, we infer the actual meaning of a word in a sentence by noticing its context.

Just as artificial neural networks simulate the human nervous system, Attention is a simulation of the human attention mechanism.

Even GPT-o1 mini can't avoid being confused by the context!

牙膏我只用中华为的就是5g就能刷干净



你好!

看起来你提到了“华为”与“5G”在使用牙膏方面的关联。让我帮你理清一下:

华为 (Huawei) : 华为是一家主要专注于通信设备、智能手机和相关技术的公司，目前并不生产牙膏产品。如果你指的是某个特定品牌的牙膏，请提供更多信息，我将乐意为你查找相关资料。

5G:

如果你指的是“5克” (5g) , 通常建议每次刷牙使用的牙膏量约为豌豆大小，大约在1克左右。5克可能超过推荐量，尤其是对于儿童来说。

如果你指的是“5G”移动通信技术，它与牙膏的清洁效果没有直接关系。

清洁效果: 有效的刷牙不仅依赖于牙膏的品牌或用量，还包括正确的刷牙方法、刷牙时间（建议至少两分钟）以及使用含氟牙膏以预防蛀牙。

如果你有更多具体的问题或需要关于牙膏选择和使用的建议，请随时告诉我!

Then we will construct three matrices: Q , K and V . We provide the formula here, later we will try to give it a plain explanation.

- ▶ $Q = EW_Q$, $W_Q \in \mathbb{R}^{d_{model} \times d_k}$, $Q \in \mathbb{R}^{n \times d_k}$
- ▶ $K = EW_K$, $W_K \in \mathbb{R}^{d_{model} \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$
- ▶ $V = EW_V$, $W_V \in \mathbb{R}^{d_{model} \times d_v}$, $V \in \mathbb{R}^{n \times d_v}$

Here W_Q , W_K , W_V are parameters, usually d_k is much smaller than d_{model} . Then the attention output is computed as

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

Here *softmax* is applied for each row, $Attention(Q, K, V) \in \mathbb{R}^{n \times d_v}$.



- ▶ Q: query. When we are reading, we may propose questions to the word we meet. For example, when we see a noun, we ponder: Is there an adjective to describe it? Is there a verb preceding it? That's our **QUERY**.
- ▶ K: key. To answer our question, when we read some other words, our brain process it so that it become a **KEY**.
- ▶ V: value. When the **QUERY** and **KEY** match each other, **VALUE** raises in our mind. For example, in the poem "近乡情更怯，不敢问来人", we first read "近乡", then "怯", finally "不敢问". Only when they are combined, we understand that the poet missed the hometown. That is the **VALUE**.
- ▶ Question: How to evaluate whether the **QUERY** match the **KEY**?



Dot product

- ▶ We mentioned that for each word, its **QUERY** and **KEY** $Q_i, K_i \in \mathbb{R}^{d_k}$.
- ▶ So we apply dot product, the bigger $Q_i^T K_j$ is, the more they are matched.
- ▶ $\frac{1}{\sqrt{d_k}}$ is the scaling factor, and we apply a *softmax* function to obtain the weights.
- ▶ $Attention(Q, K, V) \in \mathbb{R}^{n \times d_v}$, to match the shape of E , we multiply a matrix $W_0 \in \mathbb{R}^{d_v \times d_{model}}$ in the end, thus $Attention(Q, K, V)W_0 \in \mathbb{R}^{n \times d_{model}}$.



The principle of Multi-head Attention is easy to understand:

- ▶ Create h self-attention modules W_Q^i, W_K^i, W_V^i , compute Q^i, K^i, V^i for each module.
- ▶ Apply *Attention* to each one, and paste the matrix together, that is

$$[Attention(Q^1, K^1, V^1), Attention(Q^2, K^2, V^2), \dots, Attention(Q^h, K^h, V^h)] \in \mathbb{R}^{n \times hd_v}$$

- ▶ Finally, we also apply a linear transformation $W_0 \in \mathbb{R}^{hd_v \times d_{model}}$ to get the shape of $\mathbb{R}^{n \times d_{model}}$.



- ▶ Add & Norm consists of Add and Norm, the formula can be expressed as follows:

$$\text{Norm}(X + \text{MultiHeadAttention}(X))$$

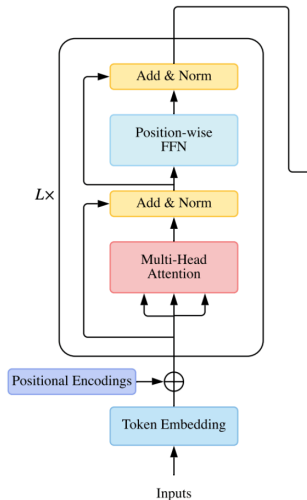
$$\text{Norm}(X + \text{FFN}(X))$$

- ▶ We have explained what's Multi-Head Attention.
- ▶ FFN is Feed Forward Network, here it's a simple fully connected network consists of two layers:

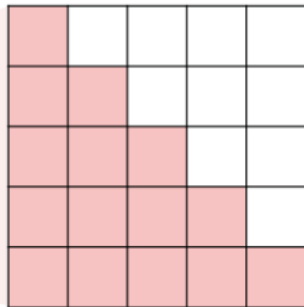
$$\max(0, XW_1 + b_1)W_2 + b_2$$

where $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{mid}}}$, $W_2 \in \mathbb{R}^{d_{\text{mid}} \times d_{\text{model}}}$.

- ▶ Now let's take a look at the whole Encoder structure.
- ▶ The Encoder is composed of a stack of $L = 6$ identical layers.
- ▶ If the input is n tokens, the embedding dimension d_{model} , then after L layers, the output **encoding information matrix** $C \in \mathbb{R}^{n \times d_{model}}$



- ▶ The mask is applied over the matrix before *softmax*, that is, $\frac{QK^T}{\sqrt{d_k}}$.
- ▶ In a Masked Multi-Head Attention, we apply a mask which assign the white blocks to $-\infty$, so that they turn out to be 0 after *softmax*.
- ▶ Question: why we need the mask?



Recall how we translate French into English: "le chat est noir" \rightarrow "the cat is black".

- ▶ $\langle SOS \rangle \rightarrow$ the
- ▶ $\langle SOS \rangle$ the \rightarrow cat
- ▶ $\langle SOS \rangle$ the cat \rightarrow is
- ▶ $\langle SOS \rangle$ the cat is \rightarrow black
- ▶ $\langle SOS \rangle$ the cat is black $\rightarrow \langle EOS \rangle$

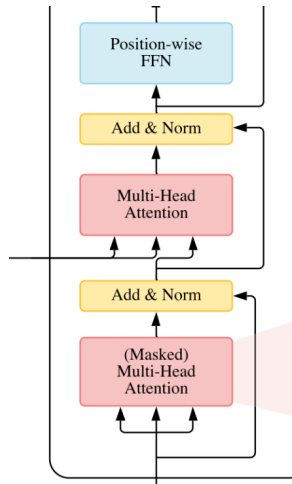
When we are training, we split the translation problem into five prediction problem, called **Teacher Forcing**. The mask does this for us, make it easier for the model to be trained in parallel.

The second Multi-Head Attention in decoder is a **Cross Attention**. The difference is that K, V comes from the output of the encoder C , and Q comes from the first decoder block X .

Suppose that the input has n tokens, and the previous output has m tokens. Then $C \in \mathbb{R}^{n \times d_{model}}$, $X \in \mathbb{R}^{m \times d_{model}}$.

Then $Q \in \mathbb{R}^{m \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$, $V \in \mathbb{R}^{n \times d_v}$, again we apply the Attention:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \in \mathbb{R}^{m \times d_v}$$

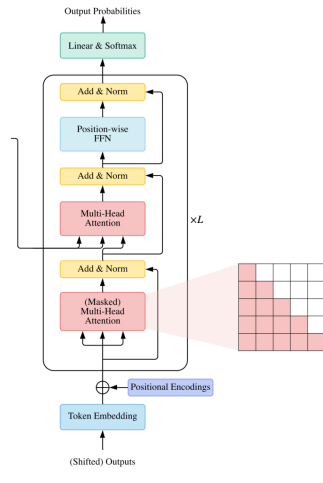


Then suppose that there are h Cross Attention blocks, then the Multi-Head Attention outputs a matrix $\mathbb{R}^{m \times hd_v}$. Finally apply a linear transformation $W_0 \in \mathbb{R}^{hd_v \times d_{model}}$, we finally obtain $\mathbb{R}^{m \times d_{model}}$.

So far, we have understood all the technical details within the transformer architecture, but there remains one question: **Why use cross-attention?**

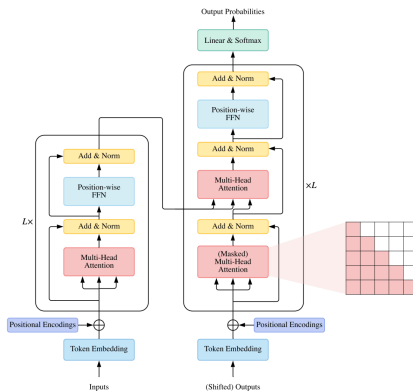
- ▶ Recall what is happening when you are writing a composition.
- ▶ First you consider the issue of coherence, that is, to ensure that the following text is fluent and coherent with the previous text. (Read the previous text, and ask: what should come next? That is Q .)
- ▶ Then you need to consider the theme, that is, what I write needs to be able to reflect the theme required by the title. (Find the answer and value in title. That is K and V)

- ▶ The Decoder is also composed of a stack of $L = 6$ identical layers, with three sub-layers in each layer.
- ▶ The encoding information matrix C is used in every decoder layers, representing the fully understanding of input.
- ▶ The mask, ensures that the predictions for position i can depend only on the known outputs at positions less than i .

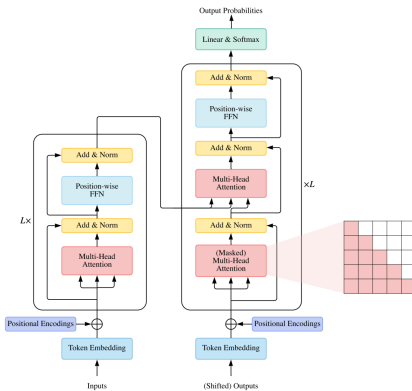


Observation 1: When ChatGPT generates outputs, for Chinese it generates several words at a time, for English it's several letters at a time. Why?

Observation 2: It usually takes some time for ChatGPT to generate the first output, but then the following outputs are generated comparatively quickly. Why?



- ▶ We now know that for a transformer, everything is token, no matter Chinese, English, voice or image. A common experience is that for English $1\text{ token} \approx 0.75\text{ words}$ and for Chinese $1\text{ token} \approx 1.5\text{ words}$.
- ▶ When generating the first output token, ChatGPT needs to **encode** (not exact here, more appropriate to say decode) the input first. After that the input don't need to be encoded again.





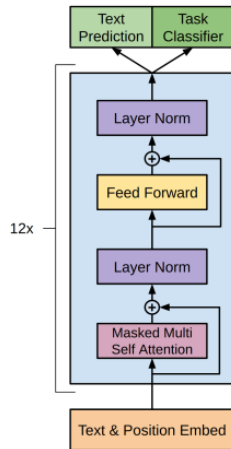
- 1 A review of Deep Learning
- 2 Attention and Transformer
- 3 BERT and GPT**
- 4 Reference

GPT-1 is a decoder-only model.

Since GPT-1 has no encoder, the Cross Attention layer and the following Add & Norm layer was deleted from the transformer.

Parameters:

- ▶ $L = 12$
- ▶ $h = 12$
- ▶ $d_{model} = 768$
- ▶ $d_k = d_v = 64$
- ▶ $d_{mid} = 3072$
- ▶ $n_{vocab} = 40000$
- ▶ $n_{max} = 512$



► Embedding layer:

- Text embedding: $n_{vocab} \times n_{model} = 30.72M$
- Position embedding: $n_{max} \times n_{model} = 0.39M$
- All: 31.11M

► Multi-Head Attention layer:

- $Q: d_{model} \times d_k = 49K$
- $K: d_{model} \times d_k = 49K$
- $V: d_{model} \times d_v = 49K$
- $W_0: d_v \times d_{model} = 49K$
- All: $196K \times 12 = 2.35M$

► Feed-Forward:

- First: $d_{model} \times d_{mid} = 2.36M$
- Second: $d_{mid} \times d_{model} = 2.36M$
- All: 4.72M

► Norm: $2 \times d_{model} = 1.5K$

► All:

$$31.11M + (2.35M + 4.72M + 1.5K \times 2) \times 12 = 115.95M$$

Parameters:

- $L = 12$
- $h = 12$
- $d_{model} = 768$
- $d_k = d_v = 64$
- $d_{mid} = 3072$
- $n_{vocab} = 40000$
- $n_{max} = 512$



Unsupervised Pre-training

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \dots, u_n\}$, maximize the following log likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

where $k = n_{max}$, here we notice that the a corpus with length n is split into $n - k$ prediction problems when training.

Supervised fine-tuning

After pre-training, they adapt the parameters to the supervised target task.

Assume a labeled dataset \mathcal{C} , where each instance consists of a sequence of input tokens x^1, \dots, x^m , along with a label y .

Maximize the following log likelihood:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \dots, x^m)$$

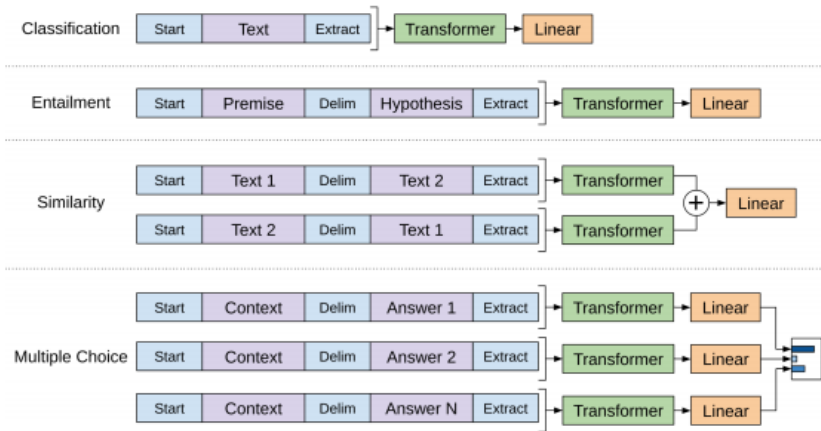
They also observe improved performance with an auxiliary objective. Specifically, they optimize the following objective

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda L_1(\mathcal{C})$$

At this point, what OpenAI was considering was not using the generative capabilities of GPT to "produce new text," but rather, like all NLP tasks before it, transforming the final output of the decoder through a linear transformation for various downstream task judgments.

Unlike the current logic of "GPT for everything," the output text generated by GPT-1 was not very logical, even chaotic. This situation did not improve until OpenAI scaled up the number of parameters to 175B (GPT-3, about 1500 times that of GPT-1).

Supervised fine-tuning

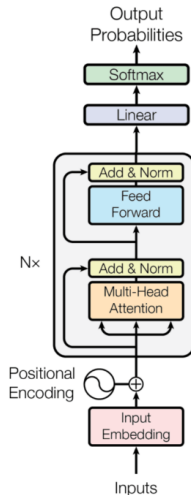


BERT is a encoder-only model.

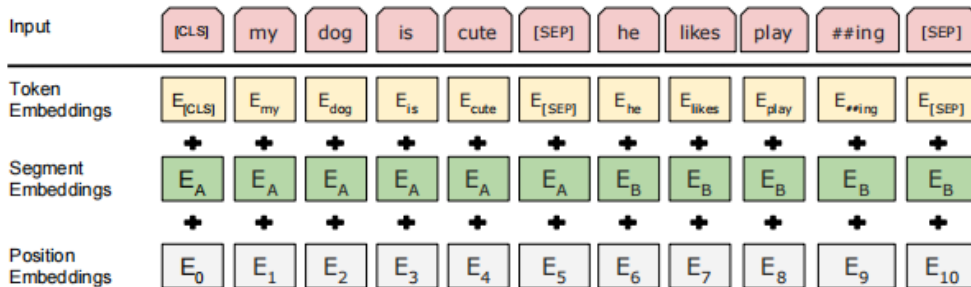
Since BERT has no decoder, the output of the Encoder was not delivered to the decoder, instead it's converted to probabilities with linear and softmax.

Parameters:

- ▶ $BERT_{BASE}(110M) : L = 12, h = 12, d_{model} = 768, d_k = d_v = 64, d_{mid} = 3072, n_{vocab} = 30000, n_{max} = 512$
- ▶ $BERT_{LARGE}(340M) : L = 24, h = 16, d_{model} = 1024, d_k = d_v = 64, d_{mid} = 4096, n_{vocab} = 30000, n_{max} = 512$



BERT input representation



Pre-training BERT

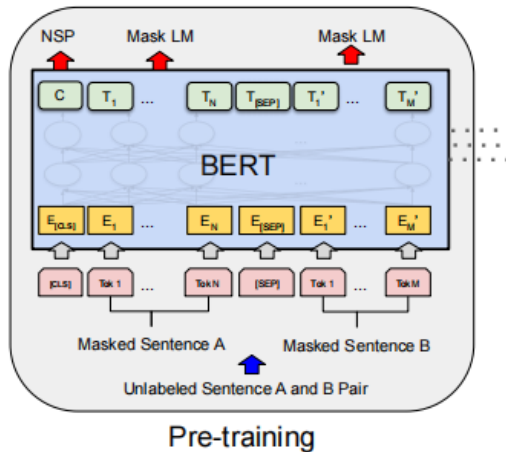
BERT is pre-trained with two unsupervised task: Masked LM and Next Sentence Prediction (NSP).

- ▶ Masked LM: They mask some percentage of the input tokens at random, and then predict those masked. Because they believe that a deep bidirectional model is strictly more powerful than either a left-to-right model. tokens.
- ▶ NSP: They combined some sentences together. Some of them are the relationship of "next sentence" while others are not.

Pre-training BERT

Some details:

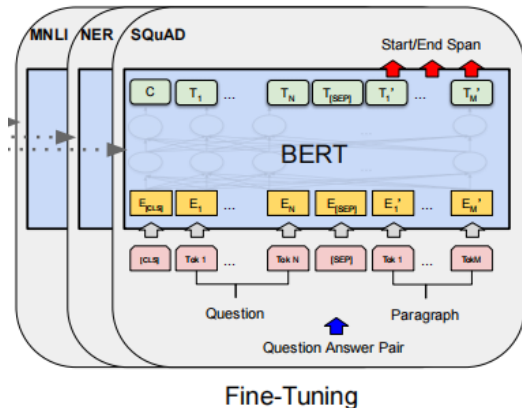
- ▶ In Masked LM, 15% of the input tokens are masked, of which
 - ▶ 80% [MASK] token
 - ▶ 10% a random token
 - ▶ 10% the unchanged token
- ▶ In Masked LM, use the corresponding output token of masked input to predict.
- ▶ In NSP, use the output token of [CLS], which is defined as "sequence representation for classification tasks", to predict.



Fine-tuning BERT

Some details:

- ▶ In Masked LM, 15% of the input tokens are masked, of which
 - ▶ 80% [MASK] token
 - ▶ 10% a random token
 - ▶ 10% the unchanged token
- ▶ In Masked LM, use the corresponding output token of masked input to predict.
- ▶ In NSP, use the output token of [CLS], which is defined as "sequence representation for classification tasks", to predict.





Language Models are Unsupervised Multitask Learners

- ▶ The contribution of the paper is more than a larger model.
- ▶ Aside from a larger model, OpenAI finds some new contributes of language models as the scale grows bigger: "Unsupervised Multitask Learners"
- ▶ "Multitask" includes Reading Comprehension, Summarization, Translation, Question Answering and so on. All of these can be done with **unsupervised learning**! It's amazing.
- ▶ This article is of landmark significance, as it demonstrates that when a model reaches a certain scale, it can learn knowledge from unannotated data.

Parameters:

$GPT-2_{XL}(1.55B) : L = 48, h = 25, d_{model} = 1600, d_k = d_v = 64, d_{mid} = 6400, n_{vocab} = 50257, n_{max} = 1024$

Datasets:

- ▶ WebText is a large-scale dataset created by scraping high-quality, diverse web pages recommended on Reddit(over 50M links, about 40G after cleaning).
- ▶ Unsupervised learning allows the model to leverage a vast amount of unlabelled data from the internet without the need for human annotation.

Language Models are Few-Shot Learners

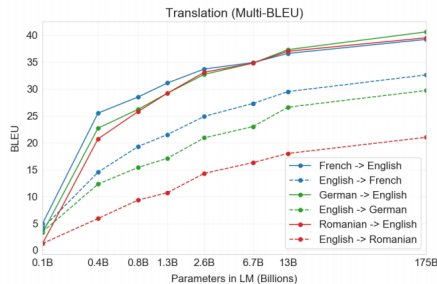
- ▶ The trained GPT-3 has the largest scale at the time, and the content it generates has truly achieved practical value for the first time.
- ▶ The **scaling law** in language models was systematically proposed, and this principle is still valid today.
- ▶ It was explained that language models are "few-shot learners," clarifying the ability of sufficiently large language models to learn knowledge from a small amount of data.

Parameters:

$GPT-3(175B) : L = 96, h = 96, d_{model} = 12288, d_k = d_v = 128, d_{mid} = 49152, n_{vocab} = 50257, n_{max} = 2048$

Scaling Law

Text completion, Question answer, Translation, Reading Comprehension, Reasoning ...



What makes OpenAI turn into CloseAI?

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p -value)	"I don't know" assignments
Control (deliberately bad model)	86%	83%–90%	-	3.6 %
GPT-3 Small	76%	72%–80%	3.9 ($2e-4$)	4.9%
GPT-3 Medium	61%	58%–65%	10.3 ($7e-21$)	6.0%
GPT-3 Large	68%	64%–72%	7.3 ($3e-11$)	8.7%
GPT-3 XL	62%	59%–65%	10.7 ($1e-19$)	7.5%
GPT-3 2.7B	62%	58%–65%	10.4 ($5e-19$)	7.1%
GPT-3 6.7B	60%	56%–63%	11.2 ($3e-21$)	6.2%
GPT-3 13B	55%	52%–58%	15.3 ($1e-32$)	7.1%
GPT-3 175B	52%	49%–54%	16.9 ($1e-34$)	7.8%

Table 3.11: Human accuracy in identifying whether short (~200 word) news articles are model generated. We find that human accuracy (measured by the ratio of correct assignments to non-neutral assignments) ranges from 86% on the control model to 52% on GPT-3 175B. This table compares mean accuracy between five different models, and shows the results of a two-sample T-Test for the difference in mean accuracy between each model and the control model (an unconditional GPT-3 Small model with increased output randomness).



In addition, this article:

- ▶ Systematically tested a large number of prompts, emphasizing the importance of prompts for the response quality of GPT-3.
- ▶ Discussed potential risks, including misuse, fairness, bias, and representation issues.

We also added that:

- ▶ The initial version of ChatGPT was based on this version of GPT-3 (175B) and was further fine-tuned using reinforcement learning from human feedback (RLHF) to better align with the role of a "chat" robot.



- 1 A review of Deep Learning
- 2 Attention and Transformer
- 3 BERT and GPT
- 4 Reference



1. Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
2. Radford, Alec, et al. "Improving language understanding by generative pre-training." (2018).
3. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019.



4. Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9.
5. Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
6. 3Blue1Brown. "Transformers (how LLMs work) explained visually | DL5." Youtube: <https://www.youtube.com/watch?v=wjZofJX0v4M>
7. 3Blue1Brown. "Attention in transformers, step-by-step | DL6." Youtube: <https://www.youtube.com/watch?v=eMlx5fFNoYc>



Thanks!