Lecture 3: transformer

谢丹
清华大学数学系

2025/08

大型语言模型训练概览

核心问题

- Transformer based Language models

- 重点解析背后的数学原理

语言模型的基本任务

语言建模问题

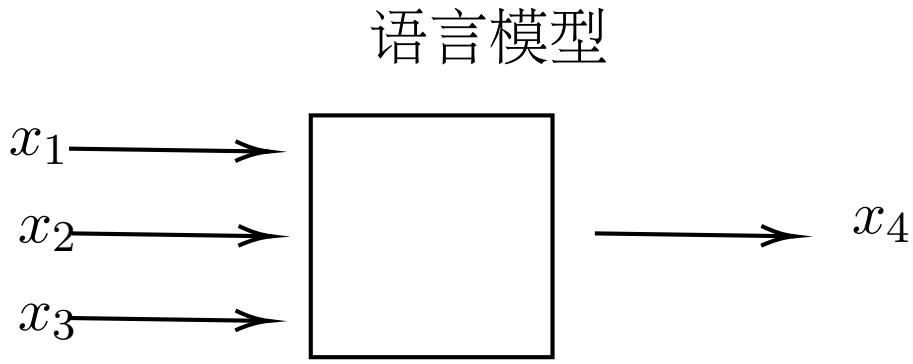
给定上下文，预测下一个最可能的词语：

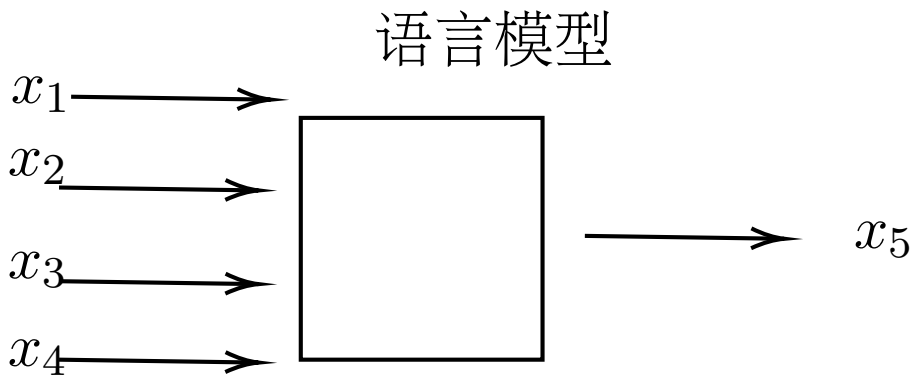
- *"I swam across the river to get to the other"* bank
- *"I walked across the road to get cash from the"* bank

技术挑战

- **长距离依赖**：需要捕捉远距离词语关系
- **多义性处理**：相同词语在不同上下文的含义（如"bank"）
- **复杂逻辑**：理解语法、常识和推理

一旦有了一个语言模型，就可以生成人类语言：





语言模型的数学表示

条件概率模型

语言模型的核心是计算条件概率 (auto regressive):

$$p(x_n|x_1,\ldots,x_{n-1})$$

其中给定前$n-1$个词，预测第n个词的概率分布。

示例

考虑词汇表：$\{\mathsf{I},\mathsf{can},\mathsf{do},\mathsf{it}\}$，给定上下文"I can do"：

$$\begin{aligned} p(\mathsf{it}|\mathsf{I},\mathsf{can},\mathsf{do}) &= 0.7 \\ p(\mathsf{I}|\mathsf{I},\mathsf{can},\mathsf{do}) &= 0.1 \\ p(\mathsf{can}|\mathsf{I},\mathsf{can},\mathsf{do}) &= 0.1 \\ p(\mathsf{do}|\mathsf{I},\mathsf{can},\mathsf{do}) &= 0.1 \end{aligned}$$

模型将预测下一个词为"it"（概率最高）。

分词器 (Tokenizer)

文本向量化的第一步

将自然语言转换为数学向量的关键步骤:

- **单词级分词**:
  - 优点: 标记数量少
  - 缺点: 丢失形态信息（如单复数、时态）
- **字符级分词**:
  - 优点: 保留更多信息
  - 缺点: 处理效率低
- 子词分词 **(Subword Tokenization)**:
  - 折中方案（如BPE、WordPiece算法）
  - 基本单位称为token
  - 平衡信息保留和计算效率

词嵌入 (Embedding)

从离散标记到连续向量

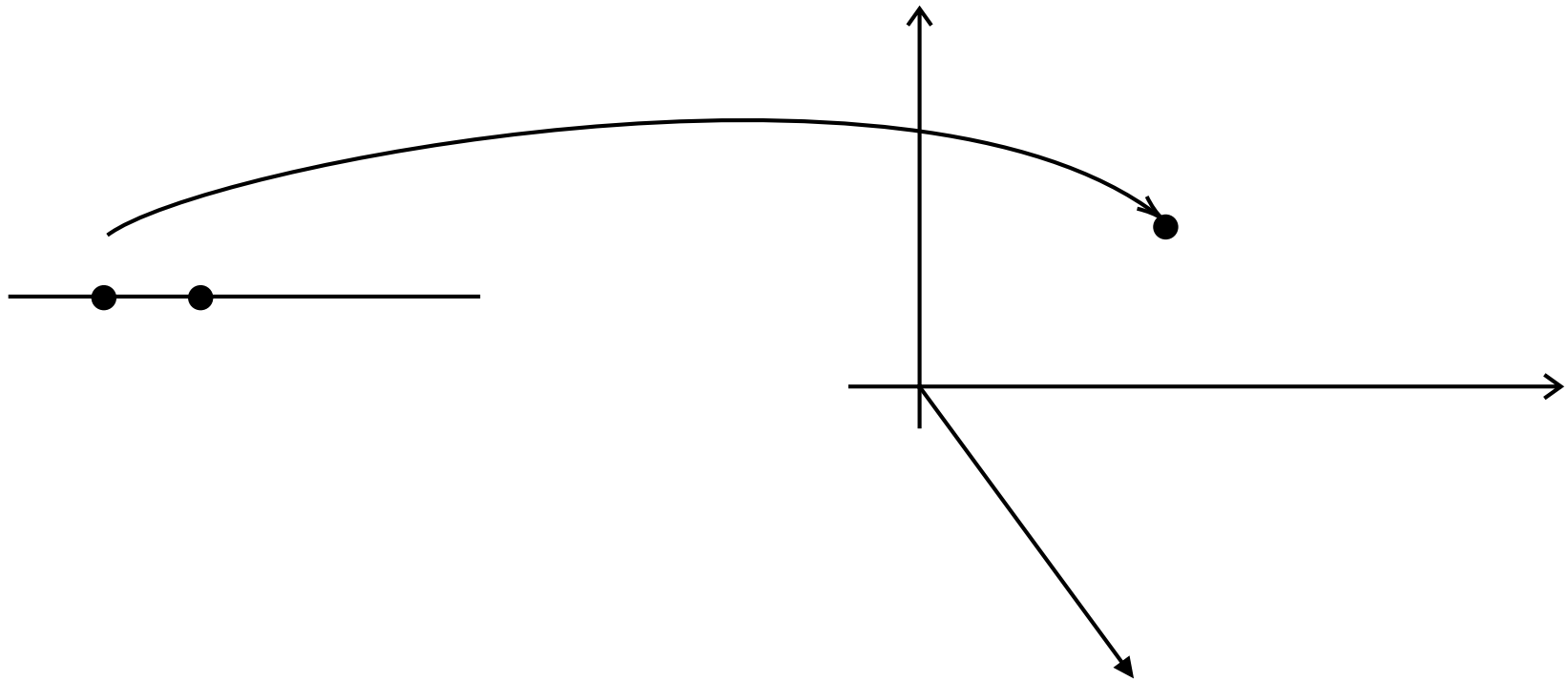
- **问题**：整数标记无法表达语义关系
- **解决方案**：将token映射到高维向量空间

关键特性

- **可学习性**：通过训练自动优化

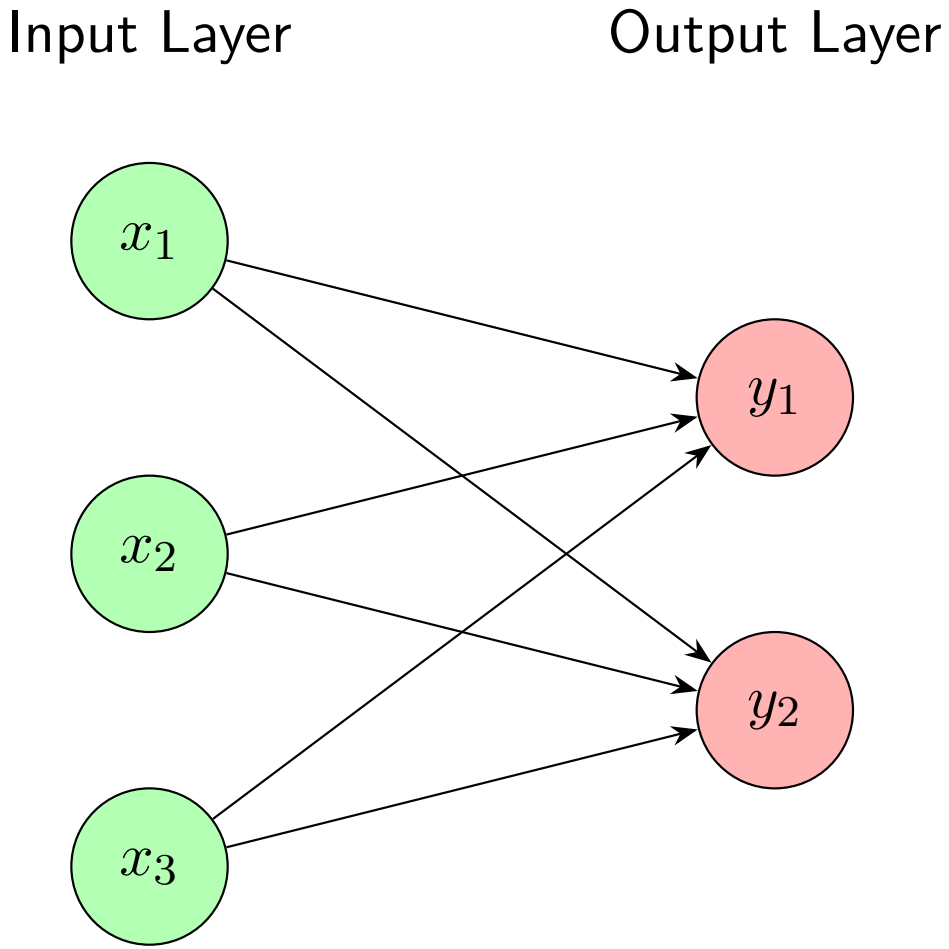
$$\text{embedding} : \mathbb{N} \to \mathbb{R}^d$$

- **语义保持**：相似词距离相近
- **维度可控**：典型维度$d \in [256, 1024]$



Embedding

The network for the embedding:



Attention Mechanism

Core Concept

注意力机制有效解决了语言模型中的长距离依赖问题，其核心思想是为每个token计算与其他token的关联权重：

- 权重越大表示关系越重要
- 权重通过训练自动学习

示例分析

句子："I swam across the river to get to the other bank"

	I	swam	across	the	river	to	get	to	the	other
bank		0.2			0.5		0.1			

- "river"和"swam"获得最高权重（0.5和0.2）
- 模型成功捕捉语义关联

Basic Attention Mechanism

Mathematical Formulation

注意力机制计算值的加权和，权重由查询-键的兼容性决定：

输入矩阵：

- 查询 $Q \in \mathbb{R}^{n \times d_k}$
- 键 $K \in \mathbb{R}^{m \times d_k}$
- 值 $V \in \mathbb{R}^{m \times d_v}$

维度说明：

- n: 查询数量
- m: 键值对数量
- d_k: 查询/键维度
- d_v: 值维度

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

The softmax is along the row direction. Check: the dimension of the matrix of the output.

$$\underset{Q}{\begin{Bmatrix} q_{11} & q_{12} & q_{13} & \cdot & \cdot & \cdot & q_{1p} \\ q_{21} & q_{22} & q_{23} & \cdot & \cdot & \cdot & q_{2p} \\ q_{31} & q_{32} & q_{33} & \cdot & \cdot & \cdot & q_{3p} \\ \cdot & & & \cdot & & & \\ \cdot & & & & \cdot & & \\ \cdot & & & & & \cdot & \\ q_{n1} & q_{n2} & q_{n3} & \cdot & \cdot & \cdot & q_{np} \end{Bmatrix}} \times \underset{K^T}{\begin{Bmatrix} k_{11} & k_{21} & k_{31} & \cdot & \cdot & \cdot & k_{n1} \\ k_{12} & k_{22} & k_{32} & \cdot & \cdot & \cdot & k_{n2} \\ k_{13} & k_{23} & k_{32} & \cdot & \cdot & \cdot & k_{n2} \\ \cdot & & & \cdot & & & \\ \cdot & & & & \cdot & & \\ \cdot & & & & & \cdot & \\ k_{1p} & k_{2p} & k_{3p} & \cdot & \cdot & \cdot & k_{np} \end{Bmatrix}} = \underset{C/\sqrt{d}}{\begin{Bmatrix} c_{11} & c_{12} & c_{13} & \cdot & \cdot & \cdot & c_{1n} \\ \cdot & \cdot & & & & & \\ \cdot & & \cdot & & & & \\ \cdot & & & \cdot & & & \\ \cdot & & & & \cdot & & \\ \cdot & & & & & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{Bmatrix}} \Longrightarrow \underset{Softmax(.)}{\begin{Bmatrix} 0.9 & 0 & 0 & \cdot & \cdot & \cdot & 0.1 \\ \cdot & \cdot & & & & & \\ \cdot & & \cdot & & & & \\ \cdot & & & \cdot & & & \\ \cdot & & & & \cdot & & \\ \cdot & & & & & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{Bmatrix}}$$

Figure: Attention score

Self-Attention

Special Case of Attention

自注意力中查询、键、值同源:

$$\text{SelfAttention}(X) = \text{Attention}(XW^Q, XW^K, XW^V) \quad (2)$$

其中X为输入序列: $n \times d_{model}$ matrix。

缩放因子

$\frac{1}{\sqrt{d_k}}$的作用:

- 防止点积结果过大
- 避免softmax进入梯度饱和区

The weight matrix W^Q, W^K, W^V are $d_{model} \times d_k$ matrices, and are learnable parameters

The attention mechanism processes an input sequence of tokens and produces a weighted dependence matrix as output, where each element quantifies the contextual relationship between token pairs.

Multi-Head Attention (Part 1/2)

Core Mechanism

Given input $X \in \mathbb{R}^{n \times d_{\text{model}}}$:

1. **Project into multiple subspaces**:

$$
\begin{aligned}
Q_i &= XW_i^Q \quad (W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}) \\
K_i &= XW_i^K \quad (W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}) \\
V_i &= XW_i^V \quad (W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v})
\end{aligned}
$$

   for each head $i \in \{1, ..., h\}$

2. **Compute attention per head**:

$$
\text{head}_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i
$$

Why Multiple Heads?

- Each head learns different attention patterns
- Enables simultaneous focus on different positions

Multi-Head Attention (Part 2/2)

Output Composition

$$\text{MultiHead}(X) = \underbrace{[\text{head}_1; \ldots; \text{head}_h]}_{\text{Concatenation}} \underbrace{W^O}_{\text{Projection}} \qquad (W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}})$$

Implementation Details

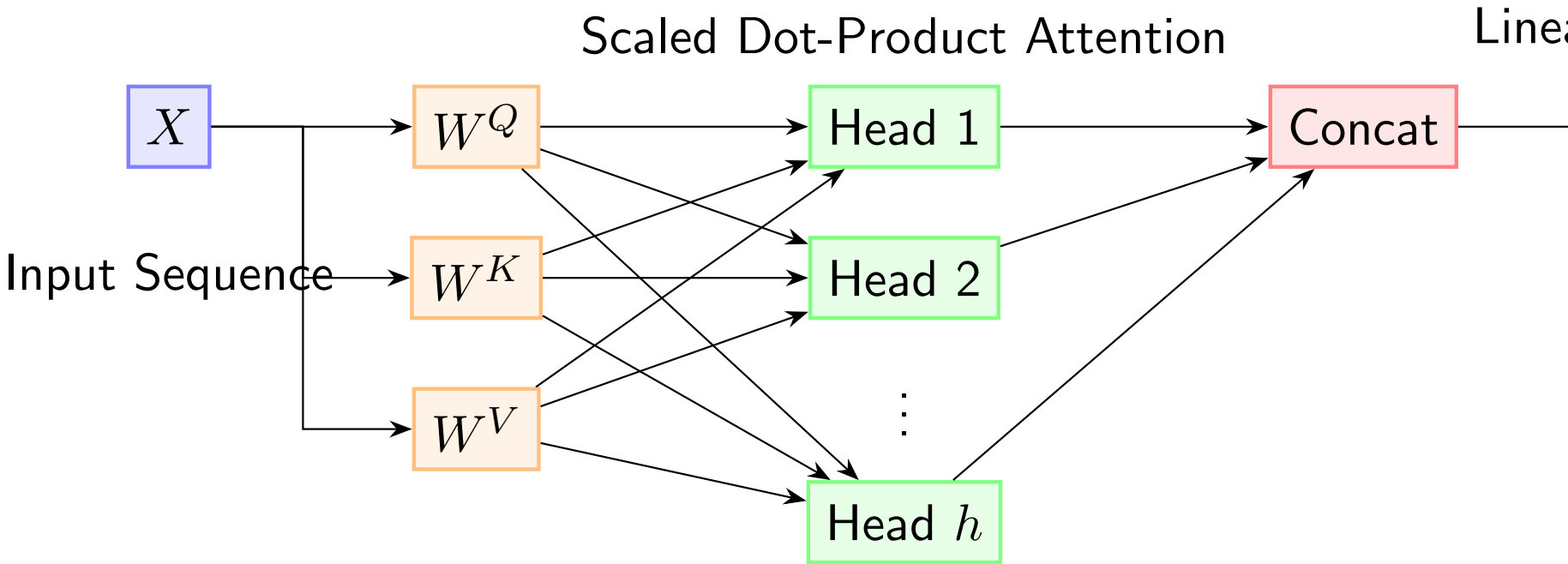
- Typical configuration: $h = 8$ heads
- Dimension allocation: $d_k = d_v = d_{\text{model}}/h$
- Computational complexity: $\mathcal{O}(n^2 \cdot d_{\text{model}})$

Counting parameters: the contribution of the multiple attention heads is

$$3 \times h \times d_{model} \times d_k$$

The concatenation contributes $h \times d_v \times d_model = d_k \times d_{model}$.

Multi-Head Attention Architecture



Positional Encoding in Transformers

The attention mechanism is symmetric in positions, we need to add position information.

Sinusoidal Positional Encoding

For position pos and $2i$ and $2i+1$ entries in model dimension d_{model}:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{3}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{4}$$

This will produce a d_{model} dimensional model.

- Fixed (non-learnable) encoding pattern
- Captures relative positions through sinusoidal frequencies
- Allows extrapolation to longer sequences

The input for the ith token is now a d_{model} dimensional vector derived from the embedding model, and the contribution of the position encoding.

Advanced Position Representations

Relative Position Representations

Modifies attention scores with relative positions:

$$e_{ij} = \frac{(x_i W^Q)(x_j W^K + \boldsymbol{a}_{ij}^K)^T}{\sqrt{d_k}} \tag{5}$$

where $\boldsymbol{a}_{ij}^K$ encodes relative position between tokens i and j.

- Captures pairwise distance relationships
- Learned embeddings for different offsets

Rotary Position Embedding (RoPE) Formulation

For a position m and embedding dimension i:

$$\theta_i = 10000^{-2i/d}$$

Rotation matrix R_m for position m:

$$R_m = \begin{pmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{pmatrix}$$

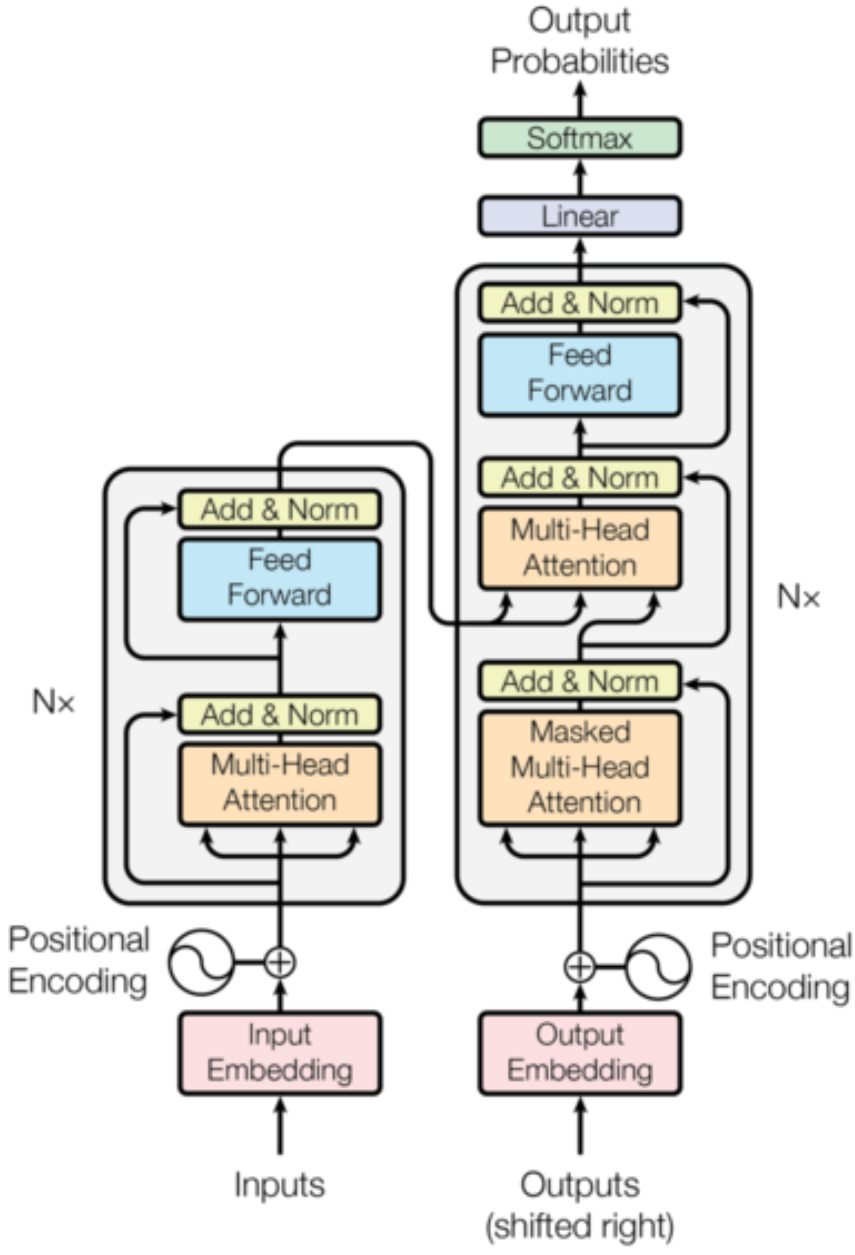
We get a $d \times d$ dimensional matrix.
Applied to queries/keys:

$$\text{Attention}(m, n) = (R_m q)^\top (R_n k) = q^\top R_{n-m} k$$

Key Property

Attention scores depend only on **relative position** $(n - m)$.

Transformer



Masked Self-Attention Formulation

For the purpose of text generation, we only need to know the attention score before a given token.

Core Equations

Given input $\boldsymbol{X} \in \mathbb{R}^{n \times d_{\text{model}}}$:

1. Project to queries, keys, values:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^{Q}, \quad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^{K}, \quad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^{V}$$

2. Compute masked attention:

$$\text{MaskedAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\underbrace{\frac{\boldsymbol{Q}\boldsymbol{K}^{T}}{\sqrt{d_k}}}_{\text{Scaled scores}} + \boldsymbol{M} \right) \boldsymbol{V}$$

Mask Matrix $\boldsymbol{M}$

$$\boldsymbol{M} = \begin{pmatrix} 0 & -\infty & -\infty & \cdots & -\infty \\ 0 & 0 & -\infty & \cdots & -\infty \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & -\infty \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

- Lower triangular structure
- Allows attention only to previous positions
- Critical for autoregressive generation

What is "Add & Norm"?

- **Core component** in Transformer layers (encoder/decoder).
- Combines two operations:
  1. **Add**: Residual connection (skip connection).
  2. **Norm**: Layer Normalization.
- Applied after **each sub-layer** (self-attention, FFN).

Purpose

Stabilize training and mitigate vanishing gradients.

Residual Connection

- **Operation**: Adds the sub-layer's input to its output.

$$\mathbf{x}_{\text{out}} = \mathbf{x} + \text{SelfAttention}(\mathbf{x})$$

- **Why?**
  - Preserves gradients (avoids vanishing gradients).
  - Allows deep networks to train efficiently.

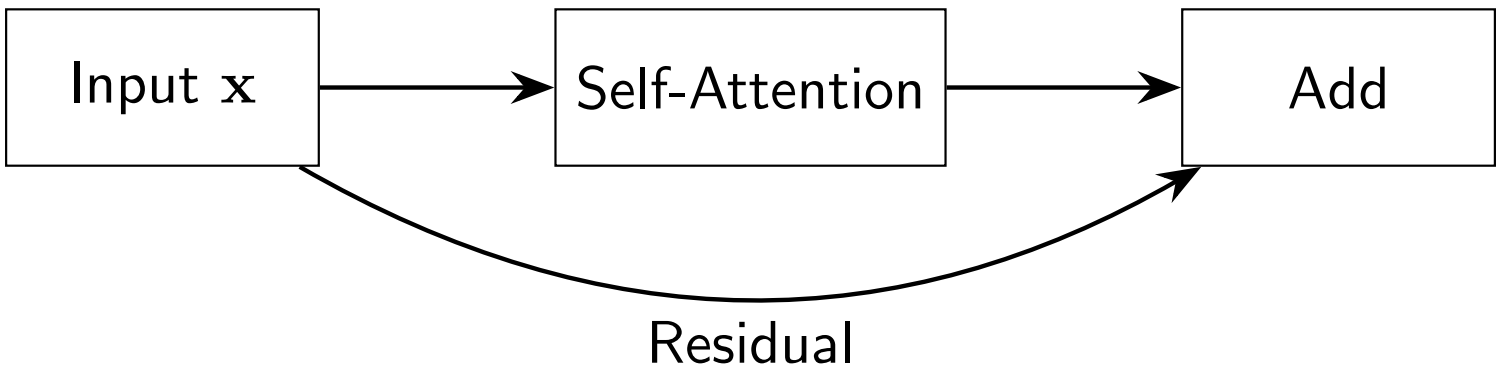


Figure: Residual connection in self-attention.

Layer Normalization

- **Operation**: Normalizes across the **feature dimension** (not batch).

$$\text{LayerNorm}(\mathbf{x}) = \gamma \left(\frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} \right) + \beta$$

  - μ, σ^2: Mean/variance of $\mathbf{x}$.
  - γ, β: Learnable parameters.
  - ϵ: Small constant (e.g., 10^{-5}).
- **Why?**
  - Stabilizes activations (reduces internal covariate shift).
  - Faster convergence.

Where is "Add & Norm" Used?

- **After self-attention**:

  $\mathbf{x}' = \text{LayerNorm}(\mathbf{x}+\text{SelfAttention}(\mathbf{x}))$

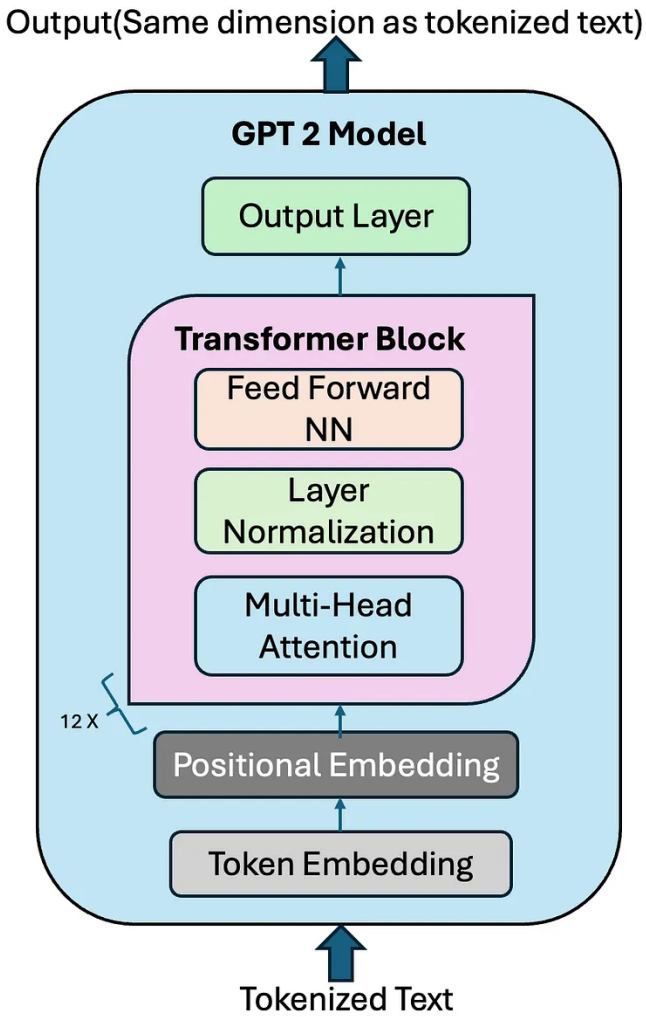
- **After FFN**:

  $\mathbf{x}'' = \text{LayerNorm}(\mathbf{x}' + \text{FFN}(\mathbf{x}'))$

The feedforward layer is the standard neural network with input d_{model} neuros and output d_{model} neuros (possibly hidden layers).

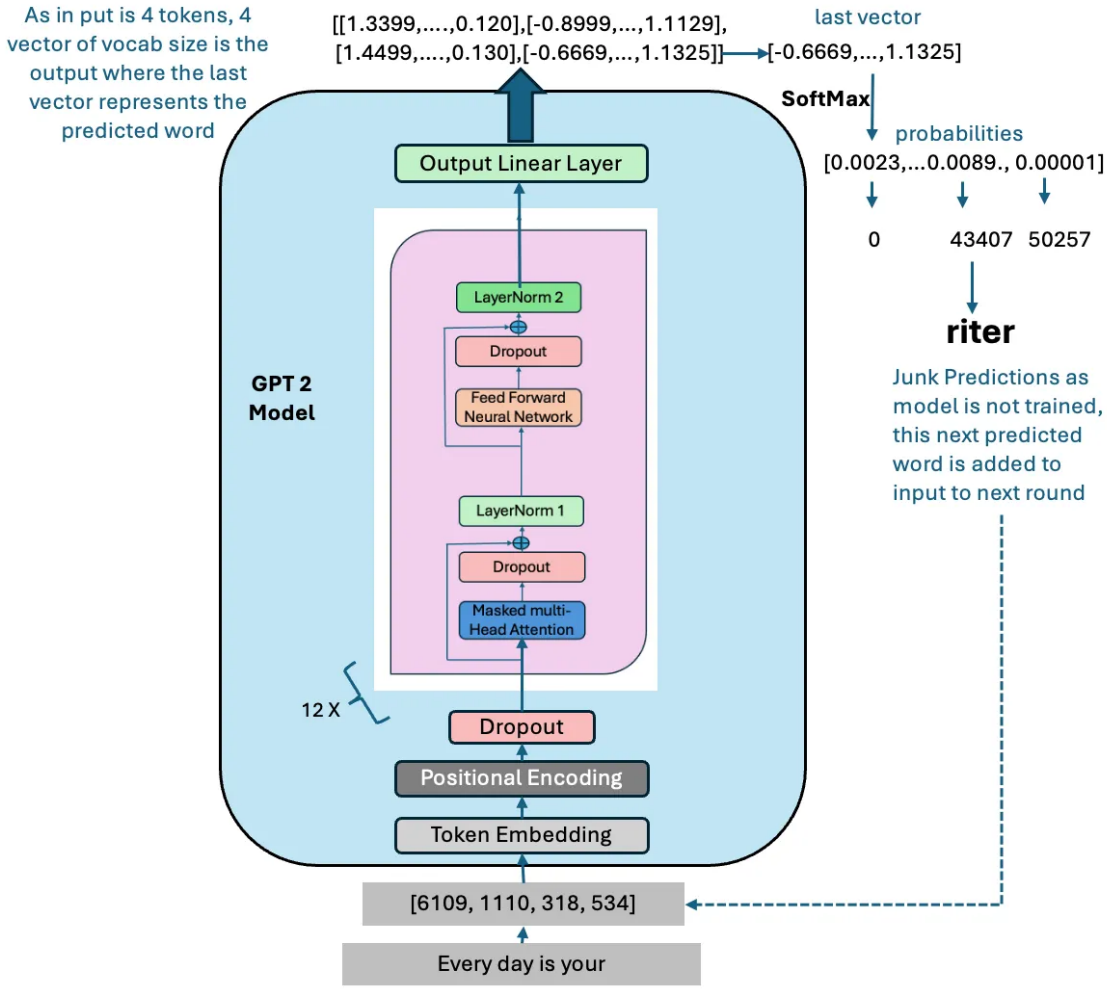
We can stack many transformer blocks and at the end add a linear layer with K output, finally, we produce K probabilities using the softmax function.

GPT2 architecture



Here is the summary: given the input of a sequence of words n, we get K probabilities, with K the size of vocabulary. We can then use it to make prediction for the next token.

$< start >$ token at the beginning, and followed by n tokens. The output are probabilities for $n + 1$ token.



Top_K: random select a work from top K probabilities.
Temperature T: