Transformer

WyEhNf

2026.07.19

Start with Translation

- Transformer was first designed for **machine translation**.
- Translation is a fundamental reasoning task:
  - Understand the source language (English)
  - Generate the target language (Chinese)
  - Preserve meaning across languages

"Our director is great."

$\Downarrow$

"我们的主任很伟大。"

The Transformer Architecture

Proposed in **"Attention Is All You Need"** (Vaswani et al., NeurIPS 2017).

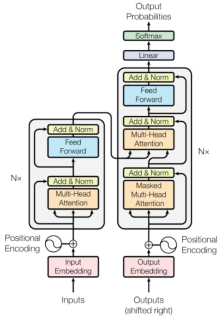


Figure 1: The Transformer - model architecture.

Input: `Our director is great.` Output: 我们的主任很伟大。

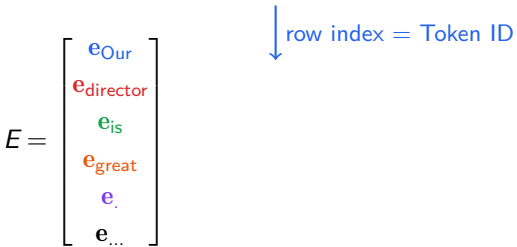
Tokenization

Step	Result	Meaning
Source text	`Our director is great.`	5 words
↓ split	`[Our, director, is, great, .]`	5 tokens
↓ map to IDs	`[12, 45, 7, 89, 3]`	integer IDs
Target text	我们的主任很伟大。	6 words
↓ split	[我们, 的, 主任, 很, 伟大, 。]	6 tokens
↓ map to IDs	`[34, 8, 56, 21, 90, 5]`	integer IDs

Each token maps to a unique integer ID in the vocabulary.

Embedding as a Lookup Table

Embedding matrix $E \in \mathbb{R}^{V \times d}$ (illustration: $V = 6,\ d = 4$)



Each row is a learnable vector for one word.

Key point: rows of E are ordered by Token ID, **not by sentence position**.
"Our" maps to $\mathbf{e}_{\text{Our}}$ regardless of where it appears.

From Tokens to Vectors

Lookup = One-Hot $\times$ E. **Row order of X = input word order.**

Input: "Our director is great ." $\rightarrow$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{One-Hot } [5\times 6]} \times \underbrace{\begin{bmatrix} \mathbf{e}_{\text{Our}} \\ \mathbf{e}_{\text{director}} \\ \mathbf{e}_{\text{is}} \\ \mathbf{e}_{\text{great}} \\ \mathbf{e}_{.} \\ \mathbf{e}_{\dots} \end{bmatrix}}_{E\ [6\times 4]} = \underbrace{\begin{bmatrix} \mathbf{x}_{\text{Our}} \\ \mathbf{x}_{\text{director}} \\ \mathbf{x}_{\text{is}} \\ \mathbf{x}_{\text{great}} \\ \mathbf{x}_{.} \end{bmatrix}}_{X_1\ [5\times 4]}$$

What if we change the word order?

"great director is Our ." $\rightarrow$ $X_2 = \begin{bmatrix} \mathbf{x}_{\text{great}} \\ \mathbf{x}_{\text{director}} \\ \mathbf{x}_{\text{is}} \\ \mathbf{x}_{\text{Our}} \\ \mathbf{x}_{.} \end{bmatrix}$

$X_1 \neq X_2$ —same vectors, different row order.

The matrix carries **no position information**: $\mathbf{x}_{\text{Our}}$ has no notion of which row it sits in.

(In practice: direct index lookup, no One-Hot multiplication —mathematically equivalent.)

Positional Encoding

- **Problem**: X merely stacks word vectors in input order. The vectors themselves contain no position info.
- "Dog bites man" vs. "Man bites dog" —identical word vectors, different order, opposite meaning.
- **Solution**: add a unique position vector to each token's embedding.

Sinusoidal Positional Encoding

$$PE_{(pos,\,2i)} = \sin\left(\frac{pos}{10000^{\,2i/d}}\right) \qquad PE_{(pos,\,2i+1)} = \cos\left(\frac{pos}{10000^{\,2i/d}}\right)$$

pos: position, *i*: dimension index. Even dims $\rightarrow \sin$, odd dims $\rightarrow \cos$.

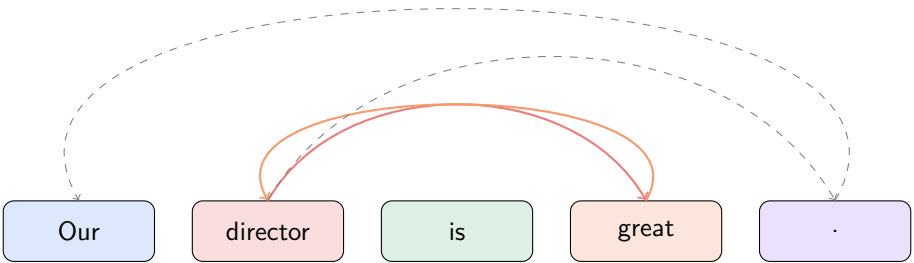
Add to Embedding:

$$\underbrace{\begin{bmatrix} \mathbf{x}_{\text{Our}} \\ \mathbf{x}_{\text{director}} \\ \mathbf{x}_{\text{is}} \\ \mathbf{x}_{\text{great}} \\ \mathbf{x}_{.} \end{bmatrix}}_{X\ [5\times 4]} + \underbrace{\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \mathbf{p}_4 \end{bmatrix}}_{\text{PE}\ [5\times 4]} = \underbrace{\begin{bmatrix} \mathbf{x}_{\text{Our}} + \mathbf{p}_0 \\ \mathbf{x}_{\text{director}} + \mathbf{p}_1 \\ \mathbf{x}_{\text{is}} + \mathbf{p}_2 \\ \mathbf{x}_{\text{great}} + \mathbf{p}_3 \\ \mathbf{x}_{.} + \mathbf{p}_4 \end{bmatrix}}_{\text{Encoder input}\ [5\times 4]}$$

PE: fixed values, no training. Same word at different positions $\rightarrow$ different representation.

Self-Attention: Intuition

Every word asks: "How relevant are you to me?"



director ↔ great: semantically related → high mutual attention
director ↔ .: unrelated → low attention

Why Does Relevance Matter?

During Training

- W^Q, W^K, W^V are learned via backpropagation
- The model learns:
  related words (director $\leftrightarrow$ great) $\rightarrow$ high score
  unrelated words (director $\leftrightarrow$.) $\rightarrow$ low score
- Word vectors align in semantic space

During Inference

- Self-Attention:
  each word **aggregates context** to resolve ambiguity
- Cross-Attention:
  generating "主任" $\rightarrow$ focuses on "director"
  generating "伟大" $\rightarrow$ focuses on "great"
- The model dynamically **queries** the source

Relevance = automatic filtering. Each word directly "sees" every other word and focuses on what matters —this is why Transformers handle long sentences.

Q, K, V: Three Learnable Projections

Same input X, project with three different weight matrices

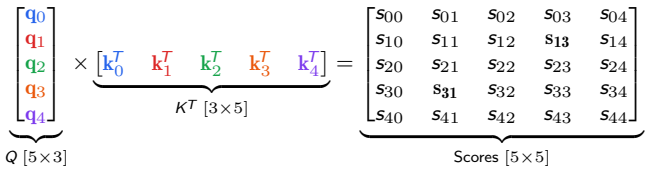
$$\underbrace{\begin{bmatrix} \mathbf{x}_{\text{Our}} \\ \mathbf{x}_{\text{director}} \\ \mathbf{x}_{\text{is}} \\ \mathbf{x}_{\text{great}} \\ \mathbf{x}_{.} \end{bmatrix}}_{X\ [5\times 4]} \times \underbrace{\begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \\ w_{30} & w_{31} & w_{32} \end{bmatrix}}_{W^Q\ [4\times 3]} = \underbrace{\begin{bmatrix} q_0^0 & q_0^1 & q_0^2 \\ q_1^0 & q_1^1 & q_1^2 \\ q_2^0 & q_2^1 & q_2^2 \\ q_3^0 & q_3^1 & q_3^2 \\ q_4^0 & q_4^1 & q_4^2 \end{bmatrix}}_{Q\ [5\times 3]}$$

Similarly, X multiplied by $W^K, W^V \in \mathbb{R}^{4\times 3}$ yields $K, V \in \mathbb{R}^{5\times 3}$

$Q[0]$ = Our's Query "What am I looking for?" $K[0]$ = Our's Key "Here is my label." $V[0]$ = Our's V

Step 1 —Score Computation: $Q \times K^T$

Each Query dots with every Key → pairwise relevance scores



Dot product $\mathbf{q} \cdot \mathbf{k} = \|\mathbf{q}\|\|\mathbf{k}\| \cos\theta$ measures similarity:

Similar direction ($\cos\theta \to 1$) → larger product → more "relevant".

$Q \times K^T$ is a **pairwise similarity comparison** between every Query and every Key.

$s_{13} = \mathbf{q}_{\text{director}} \cdot \mathbf{k}_{\text{great}}$: director → great —should be **high** (semantically related)

$s_{31} = \mathbf{q}_{\text{great}} \cdot \mathbf{k}_{\text{director}}$: great → director —also **high**

Step 2 & 3 —Scale & Softmax

Scale: divide by $\sqrt{d_k} = \sqrt{3}$ to keep dot products from growing too large.
Softmax: normalize each row to a probability distribution (sum $= 1$).

$$\text{Scores } [5 \times 5] \xrightarrow{\div\sqrt{3}} \xrightarrow{\text{softmax(row-wise)}} \text{Attention Weights } [5 \times 5]$$

$$A = \begin{bmatrix} 0.80 & 0.10 & 0.05 & 0.03 & 0.02 \\ 0.15 & 0.70 & 0.05 & \mathbf{0.08} & 0.02 \\ 0.10 & 0.10 & 0.65 & 0.10 & 0.05 \\ 0.05 & \mathbf{0.30} & 0.10 & 0.50 & 0.05 \\ 0.10 & 0.10 & 0.10 & 0.10 & 0.60 \end{bmatrix}$$

Row *i*: how token *i* distributes attention across all tokens.
director → great (0.08), great → director (0.30) —mutual high attention.

Why Scale by $\sqrt{d_k}$?

- **Assumption**: components of $\mathbf{q}, \mathbf{k}$ are independent, mean 0, variance 1.

$$\textbf{Dot product } \mathbf{q} \cdot \mathbf{k} = q_1 k_1 + q_2 k_2 + \cdots + q_{d_k} k_{d_k}$$

Each term $q_j k_j$ has variance ≈ 1; there are d_k such terms:

$$\Downarrow$$

$$\boxed{\text{Var}(\mathbf{q} \cdot \mathbf{k}) \approx d_k}$$

- With $d_k = 64$ (the original paper), std dev ≈ 8, values can reach $\pm 20 \sim 30$.
- Such large values enter softmax $\rightarrow$ output is extremely sharp (near one-hot)
- **Gradients near zero** $\rightarrow$ model cannot learn

Division by $\sqrt{d_k}$: $\text{Var}\left(\dfrac{\mathbf{q} \cdot \mathbf{k}}{\sqrt{d_k}}\right) \approx 1 \quad \rightarrow$ Variance restored to 1; softmax stays smooth; gradients flow.

Step 4 —Weighted Sum: $A \times V$

Blend Value vectors using attention weights —each output is context-aware

$$\underbrace{\begin{bmatrix} 0.80 & 0.10 & 0.05 & 0.03 & 0.02 \\ 0.15 & 0.70 & 0.05 & 0.08 & 0.02 \\ 0.10 & 0.10 & 0.65 & 0.10 & 0.05 \\ 0.05 & 0.30 & 0.10 & 0.50 & 0.05 \\ 0.10 & 0.10 & 0.10 & 0.10 & 0.60 \end{bmatrix}}_{A\ [5\times5]} \times \underbrace{\begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \end{bmatrix}}_{V\ [5\times3]} = \underbrace{\begin{bmatrix} \mathbf{o}_0 \\ \mathbf{o}_1 \\ \mathbf{o}_2 \\ \mathbf{o}_3 \\ \mathbf{o}_4 \end{bmatrix}}_{\text{Output}\ [5\times3]}$$

Example: the new "great" vector (row 4):

$$\mathbf{o}_{\text{great}} = 0.05 \cdot \mathbf{v}_{\text{Our}} + \mathbf{0.30} \cdot \mathbf{v}_{\text{director}} + 0.10 \cdot \mathbf{v}_{\text{is}} + 0.50 \cdot \mathbf{v}_{\text{great}} + 0.05 \cdot \mathbf{v}_{.}$$

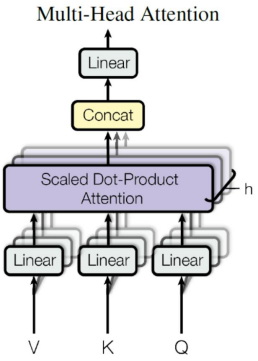
Each output is a **context-aware** blend —the word "sees" the entire sentence.

Self-Attention: Summary

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

Multi-Head Attention

One head → one relationship pattern. **Multiple heads → multiple perspectives.**



Multi-Head Attention: How It Works

Illustration with $h = 2$ heads, $d_k = 2$ (total $d = 4$):

$$\text{head}_1 = \text{Attention}(XW_1^Q, XW_1^K, XW_1^V) \in \mathbb{R}^{5\times 2}$$

$$\text{head}_2 = \text{Attention}(XW_2^Q, XW_2^K, XW_2^V) \in \mathbb{R}^{5\times 2}$$

$$\text{MultiHead} = \underbrace{[\ \text{head}_1 \mid \text{head}_2\ ]}_{[5\times 4]}\ W^O_{[4\times 4]}$$

Each head has its own $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{4\times 2}$ (parameters not shared).
Head 1 may capture syntactic patterns; Head 2 may capture semantic relationships.

Masked Self-Attention

- **Problem**: during training, the Decoder sees the entire target sentence. When generating token i, it must not peek at tokens $i+1, i+2, \ldots$
- **Solution**: set the upper-triangle scores to $-\infty$ before softmax.

Recall Softmax:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \qquad \Rightarrow \qquad e^{-\infty} = 0 \;\Rightarrow\; \textbf{masked positions} \to \textbf{0 after softmax}$$

Demonstration:

$$\text{Scores } [5 \times 5] \xrightarrow{\text{mask } -\infty} \begin{bmatrix} s_{00} & -\infty & -\infty & -\infty & -\infty \\ s_{10} & s_{11} & -\infty & -\infty & -\infty \\ s_{20} & s_{21} & s_{22} & -\infty & -\infty \\ s_{30} & s_{31} & s_{32} & s_{33} & -\infty \\ s_{40} & s_{41} & s_{42} & s_{43} & s_{44} \end{bmatrix} \xrightarrow{\text{softmax}} \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 \\ 0.4 & 0.6 & 0 & 0 & 0 \\ 0.2 & 0.1 & 0.7 & 0 & 0 \\ 0.1 & 0.15 & 0.2 & 0.55 & 0 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.6 \end{bmatrix}$$

Training: upper-triangle mask enables parallel computation without cheating.
Inference: naturally causal —tokens generated one at a time.

Cross-Attention: Decoder ← Encoder

The bridge between source and target languages.

$$\underbrace{Q_{\text{dec}}}_{[6\times3]} \quad \underbrace{K_{\text{enc}}}_{[5\times3]} \quad \underbrace{V_{\text{enc}}}_{[5\times3]}$$

	Self-Attention	**Cross-Attention**
Q from	current layer input	**Decoder** current layer
K, V from	current layer input	**Encoder** final output
Purpose	intra-language relations	cross-language alignment
Example	"great" → "director"	"主任" → "director"

When generating "主任": Cross-Attention assigns highest weight to "director".
When generating "伟大": Cross-Attention assigns highest weight to "great".
At every step, the Decoder **re-examines the entire source sentence**.

Feed-Forward & Residual Connections

Feed-Forward Network (FFN)

$$\text{FFN}(\mathbf{x}) = \text{ReLU}(\mathbf{x}W_1 + \mathbf{b}_1)W_2 + \mathbf{b}_2$$

- Applied per-position, parameters shared
- $W_1 \in \mathbb{R}^{4\times 8}$, $W_2 \in \mathbb{R}^{8\times 4}$
- Expand $\rightarrow$ non-linear transform $\rightarrow$ compress

Residual + LayerNorm

$$\text{output} = \text{LayerNorm}(\ \mathbf{x} + \text{sublayer}(\mathbf{x})\ )$$

- **Add** (residual): $\mathbf{x} + f(\mathbf{x})$ —gradient highway
- **LayerNorm**: normalize each row to mean 0, variance 1
- Stabilizes and accelerates training

Every sub-layer: Input $\rightarrow$ [Attention / FFN] $\rightarrow$ Add $\rightarrow$ LayerNorm $\rightarrow$ Output

Output: Linear & Softmax

Decoder output $\xrightarrow{\text{Linear}}$ **logits** $\xrightarrow{\text{Softmax}}$ **vocabulary probabilities**

$$\underbrace{\begin{bmatrix} \mathbf{o}_{\text{我们}} \\ \mathbf{o}_{\text{的}} \\ \mathbf{o}_{\text{主任}} \\ \mathbf{o}_{\text{很}} \\ \mathbf{o}_{\text{伟大}} \\ \mathbf{o}_{\text{。}} \end{bmatrix}}_{[6\times 4]} \times \underbrace{W^{\text{out}}}_{[4\times V_{\text{zh}}]} = \underbrace{\begin{bmatrix} z_0^1 & z_0^2 & \cdots & z_0^V \\ \vdots & \vdots & \ddots & \vdots \\ z_5^1 & z_5^2 & \cdots & z_5^V \end{bmatrix}}_{\text{logits } [6\times V_{\text{zh}}]} \xrightarrow{\text{softmax}} \underbrace{\begin{bmatrix} p(\text{我们}) & p(\text{的}) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}}_{\text{probabilities } [6\times V_{\text{zh}}]}$$

Each row $\xrightarrow{\arg\max}$ predicted next token:

我们 → 的 → 主任 → 很 → 伟大 → 。 → <eos>

Autoregressive Decoding

Generate one token at a time, feed it back as input.

Step	Decoder Input	Output (next token)
1	<s>	我们
2	<s> 我们	的
3	<s> 我们的	主任 ← attends to “director”
4	<s> ... 主任	很
5	<s> ... 很	伟大 ← attends to “great”
6	<s> ... 伟大	。
7	<s> ... 。	<eos>

At every step, Cross-Attention **re-examines the entire Encoder output**.
The model aligns each Chinese token with its relevant English source word(s).

Training vs. Inference

Training

- **Teacher Forcing**: feed the *ground-truth* previous tokens
- Entire target sequence input at once
- Mask prevents looking ahead
- Loss: $\mathcal{L} = -\sum_t \log P(y_t \mid y_{<t}, \mathbf{x})$
- Gradients flow through all positions in parallel

Inference

- **Autoregressive**: generate token by token
- Stop when <eos> is generated or max length reached
- Causal masking is naturally satisfied

Why Is Training Parallel?

Compare with RNN:

	RNN	**Transformer**
Computation	sequential: $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$	parallel: Attention in one step
Position dependency	t must wait for $t-1$	all positions computed simultaneously
Speed	slow, inherently sequential	fast, all positions at once

Why can Transformer do this?

1. **No recurrence** —no hidden state waiting for the previous step
2. **Self-Attention sees all at once**: $Q \times K^T$ computes all pairwise scores in a single matmul
3. **Teacher Forcing + Mask**: feed the entire target sequence at once; mask ensures causality
4. Gradients backpropagate through all positions simultaneously

In one sentence: RNN waits for step $t-1$ to finish before step t;
Transformer flattens all positions into a matrix and computes everything in one pass.

Summary

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

Encoder-Decoder
encode source
decode target

Self-Attention
Q queries · K matches · V delivers
every word sees all others

Parallel Training
no recurrence
mask + one forward pass

Throw away RNN's sequential bottleneck. Model an entire sequence's dependencies in a single matrix multiplication.

Q & A

Thanks for listening!