同一个问题，问了五代 GPT

ChatGPT 是怎么诞生的？

从一个荒谬到简单的目标，到一个会聊天的助手

为什么海水是咸的？

1 GPT-1
几乎答非所问

2 GPT-2
会说话了，但仍空洞

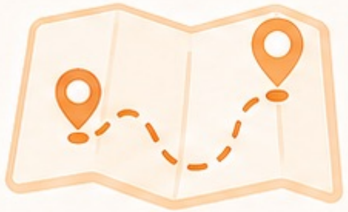
3 GPT-3.5
基本答对了

4 GPT-4
解释更清楚、更完整

5 GPT-4.5
像老师一样会讲

这一讲，我们既看它怎么工作，也看它怎么被训练出来。

本讲地图：一台机器的『诞生史』



前半讲：它怎么工作 | 后半讲：它怎么长大

1

开场

预测下一个词

先看 AI 到底在做什么

2

读懂你的话

Token · Embedding

文字怎样进入机器

3

机器的心脏

Attention · Transformer

模型最关键的内部结构

4

养成记

预训练 · 微调 · 对齐 · 涌现

它怎么从不会到会聊天

小提示：每遇到一个新术语，都回到这张图：它在哪一段、负责哪一步？

记住一句话：前半讲讲『它怎么工作』，后半讲讲『它怎么长大』。

五代 GPT，回答同一个问题

同一个问题：为什么海水是咸的？

为什么海水是咸的？

越来越聪明

1

GPT-1（2018）

重复绕圈，几乎答非所问

不断重复“生命的目的是什么”之类内容。

2

GPT-2（2019）

句子通顺了，但仍空洞跑偏

会说“海洋是咸水，也有很多淡水”，却没答清原因。

3

GPT-3.5T（2023）

基本答对

盐来自岩石和土壤，被水溶解后经河流带入海中。

4

GPT-4T（2023）

分层讲清

雨水侵蚀岩石
↓
河流搬运盐分
↓
海洋长期积累

5

GPT-4.5（2025）

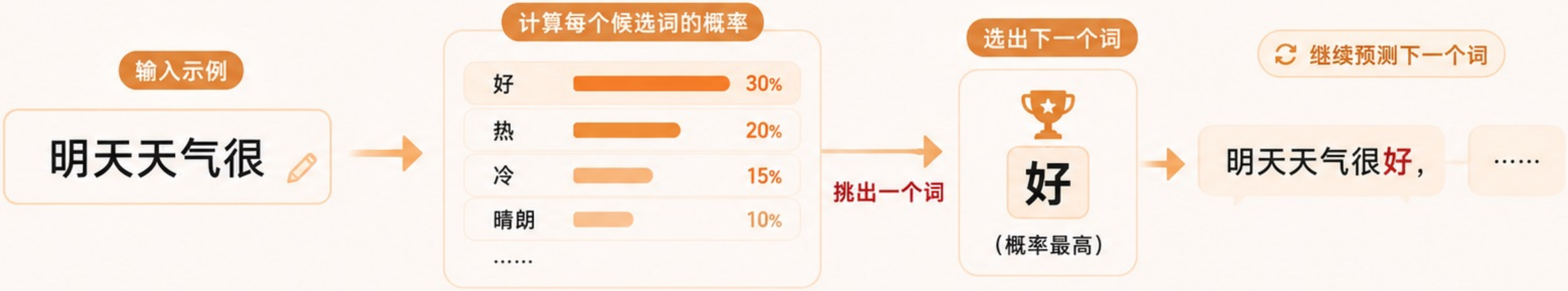
正确之外，更像老师在讲解

雨水落在岩石上，
河流把盐分带进海洋；
水蒸发，盐留下。

底层做的仍是同一件事；进步主要来自：更大、读得更多、调教得更好。

它做的事，简单到荒谬：预测下一个词

大语言模型的核心任务，就是**不断预测「下一个最合适的词」**。



1. 输入问题
2. 计算每个候选词的概率
3. 选出下一个词
4. 重复很多次，形成整段回答

它不是一下子想完整段，而是一个词一个词往下接。

看似复杂的回答，本质上都是一次又一次「预测下一个词」的结果。

一个反直觉的真相

为什么「只训练机器猜下一个词」，反而带来了惊人的能力？

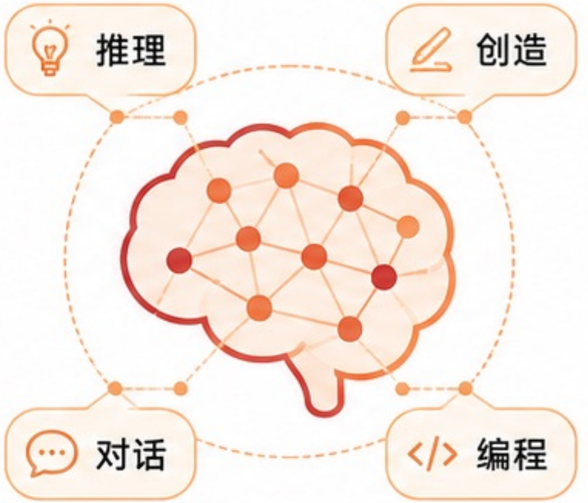
过去很多人以为……

- AI 要变聪明，得配**知识库**
- 得有专门的**推理模块**
- 得为每个任务精心设计**规则**

范式转变

实际发生的是……

- 只训练一个**巨型神经网络**
- 目标非常简单：**预测下一个词**
- 结果却逐渐表现出：
  推理、创造、对话、编程



当规模足够大、读的文字足够多，把简单的事做到极致，就会出现令人惊讶的能力。

04 · 开场

语言模型简史

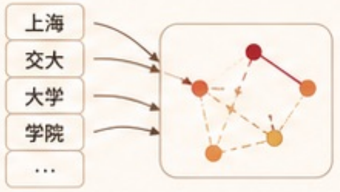
语言模型 70 年：一页纸看完

1 N-gram
(1950s–2000s)

查统计猜下一个词。
两堵墙：记不住远处、
不懂相似。

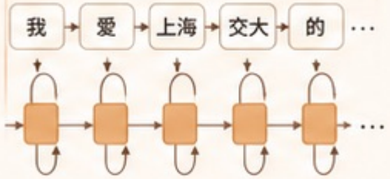
2 神经语言模型
(2003)

把词变成向量，
相似的词第一次有了
相似表示。



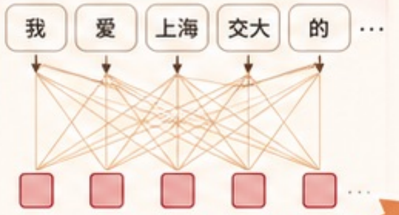
3 RNN / LSTM
(2012–2017)

用「状态」边读边记，
仍记不住太远、且只能
逐词串行。



4 Transformer
(2017 —)

一举解决两大难题，
统一了今天所有大模型。



现代大模型家族

- **GPT 系列** — 通用对话与创作
- **Claude 系列** — 长文本与推理
- **DeepSeek 系列** — 高性能与工程创新
- **Qwen 系列** — 多语言与开源生态

…

四次跃迁，不断突破边界

- **从统计到表示** — 从“频次”到“语义”
- **从词到向量** — 计算机开始“懂”相似
- **从顺序到记忆** — 有了状态，但容量有限
- **从记忆到注意力** — 并行、长程、可扩展

本质上，语言模型就是：**把人类的语言规律压缩进一个数学模型里，让机器学会“下一个词”。** Transformer 让这件事，第一次在所有维度上都足够好了。

第一部分

读懂你的话：文字怎么进入机器

先看模型读到你这句话的头两关——切碎（Token）、变成坐标（Embedding）。

这两关的意义

切碎让机器“看得见”，变成坐标让机器“看得懂”。

Token	让文字可计算、可处理
Embedding	让文字有含义、可比较

Token 长什么样：四个例子

ChatGPT 不直接认字

它处理的最小单位是 Token，
进门前先被分词器切碎。

四个例子

英文按子词、中文按字或词、
代码按符号、emoji 也算一块。

Token 不是字也不是词

它是「常见的文字片段」，
可能比字小，也可能比字大。

小结

不同做型、不同分词器，切法略有不同，
但原理一致：先切碎，后理解。

一句话如何被分词器切成 Token

输入句子

为什么海水是咸的？

分词器

被切成 Token（示例）

每一个方块 = 一个 Token

A 英文（按子词）

示例：unbelievable

un | believe | able

常见模法：按子词切分，
而不是按完整单词。

中 ② 中文（按字或词）

示例：为什么海水是咸的?

为 | 什么 | 海水 | 是 | 咸 | 的 | ?

中文通常按字或常用词切分，
一个词可能包含多个字。

</> ③ 代码（按符号）

示例：for(i=0;i<n;i++){sum+=a[i];}

for | (| i | = | 0 | ; | i | < | n | ; |
i | ++ |) | { | sum | += | a | [| i |] | ; | }

代码按符号和关键字切分，
利于理解语法结构。

Emoji（也算一块）

示例：😊

一个 emoji 通常被视为
一个整体 Token。

第一站 · Token

换算口诀、数量感与上下文窗口

1 换算口诀

约值：国际模型 ≈ 1–1.5 token/汉字；
中文优化模型常更省，具体以 tokenizer 为准。

2 上下文窗口 = 工作台面

模型一次能看到的最大 token 数；
长材料最好分段处理再汇总。

3 它解释了「AI 像忘了前面」

聊太久或资料太长，早期内容会滑出窗口，
真的看不见。

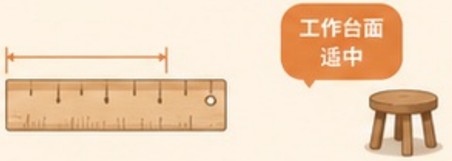
上下文窗口（工作台面）

模型一次能“看见”的范围

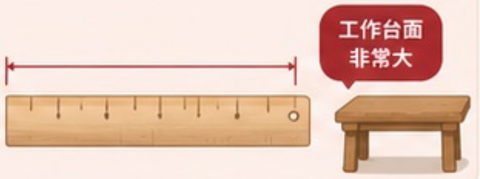
不同模型的上下文窗口大小（示例）

数字越大，工作台面越大

128K
典型大模型
约可容纳 ~8–10 万汉字



200万
超长上下文模型
约可容纳 ~150–200 万汉字



实用建议 | 长材料先分段 | 每段小结，再汇总 | 把最重要的信息放在窗口前面 | 必要时让模型“回看要点”，及时复述关键信息

第一站 · Token

BPE：分词表是「学」出来的

用数据自我学习，把常见的组合变成新的 token 积木。

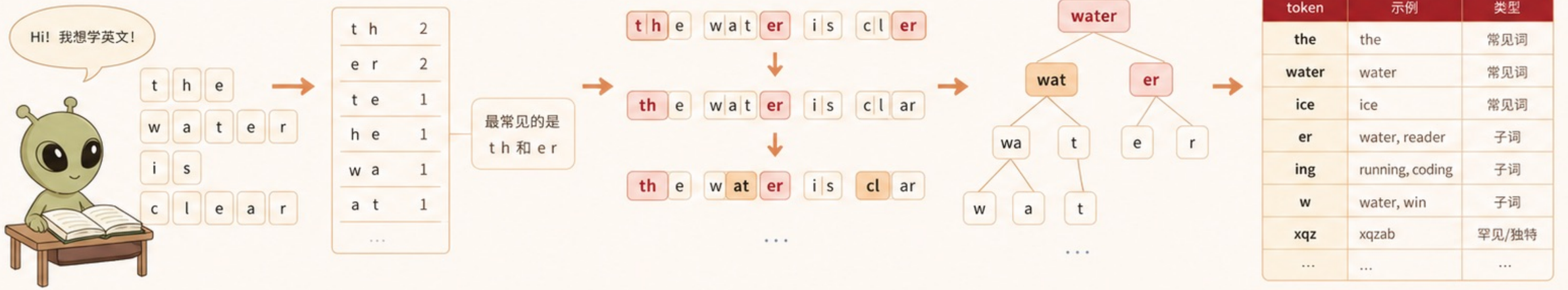
1 从字符开始
把所有文本拆成单个字符

2 统计相邻出现频次
找出最常一起出现的组合

3 合并最常见的组合
把它们变成新的积木

4 重复合并
不断迭代，学习更多组合

5 得到一张词表
这就是 BPE 学出来的词表（节选）



token	示例	类型
the	the	常见词
water	water	常见词
ice	ice	常见词
er	water, reader	子词
ing	running, coding	子词
w	water, win	子词
xqz	xqzab	罕见/独特
...	...	...

先拆成字母，
再不断把最常一起出现的
组合并成新积木。

合并几万次 → 一张词表

常见词不切碎、
罕见词拆子词、
任何字符串都能编码。

规律来自数据，
不是人手写

这正是深度学习的
核心精神。

例子：编码 "the water is clear"

最终可能变成：

[the] [water] [is] clear]

→ 高效、稳定、可泛化

第一站 · Token

Token 牵动的三件事 + 几个「原来如此」

为什么上下文窗口、输出方式和 API 计费都与 Token 有关

1 三件大事都按 token 算

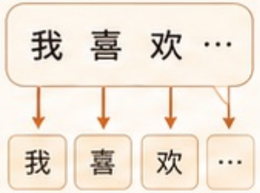
1 上下文窗口

模型一次最多能看到多少 token

- 包括：系统提示、对话历史、用户输入、检索内容等
- 超出窗口，模型“看不到”后面的内容

2 逐字输出

回答其实是一个 token 一个 token 生成



- 模型一步步“长”出答案
- 停在某个 token 就是你的答案
- 温度、停止词等影响结果

3 API 计费

输入 token + 输出 token 计费，通常输出更贵

- 输入：你发给模型的token
- 输出：模型生成的 token
- 输出单价通常高于输入
- 长回答 = 成本更高

2 几个「原来如此」

A 为什么数不准字数？

请写 100 字 → 模型眼里是 token，不是字

B 为什么以前总答不准 strawberry里有几个 r？

strawberry → [str] [aw] [berry]

它并不天然看见字母内部

C 为什么不同语言成本不同？

意思相近的表达	大致 token 数（示例） 少 → 多
中文：我喜欢人工智能。	≈ 6
English: I like artificial intelligence.	≈ 9
日本語：私は人工知能が好きです。	≈ 12

同样意思，不同语言切分数不同

3 老师使用时的提示

少下硬性字数命令

“写 500 字”并不准确，模型按 token 计数，难以精确控制。

多给结构要求

用要点、层级、格式、表格等结构化约束，更稳定、更可控。

重要信息前置

把最关键的背景、约束、资料放在前面，避免被截断或稀释。

Token 不是一个幕后小细节，
它直接影响 能看多长、怎么输出、以及花多少钱。

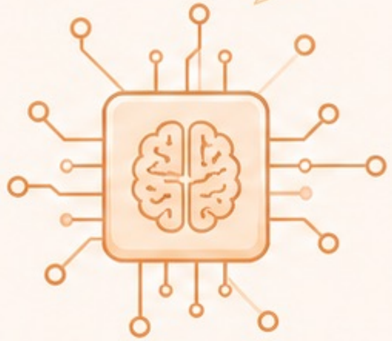
- 能看多长（上下文窗口）
- 怎么输出（逐 token 生成）
- 花多少钱（API 计费）

第二站 · Embedding

Token 怎么变数字——先看一个失败的笨办法

神经网络只认数字，但简单编号并不能表达词义

“人工”“智能”这些词，怎么变成数字？



笨办法：给每个词编号

老师	学生	苹果	中国	物理	上海	…
= 1	= 2	= 3	= 8172	= 19	= 204	= …

这些编号彼此孤立，毫无关系

为什么这个办法不行？

1. 编号只是名牌，不含意义

2. 1 和 2 挨着，不代表老师和学生更像

1 —— 2

3. 1 和 8172 差很远，也不代表毫不相关

1 ····· 8172

真正的问题不是“能不能数字化”，而是“数字里有没有意义”

Embedding 的突破，就在于让数字本身开始携带语义。

第二站 · Embedding

用向量表示词：每个词是空间里的一个点

把词变成一串有意义的数字——这就是 Embedding

给每个词一串数字（向量）

把每个词映射为多维空间中的一个向量。
（这里只展示 4 维向量示例）


老师 → [0.23, 0.87, 0.12, -0.45]


学生 → [0.21, 0.85, -0.30, 0.12]


课本 → [0.25, 0.76, -0.15, -0.22]

…


苹果 → [0.51, -0.12, 0.91, 0.34]

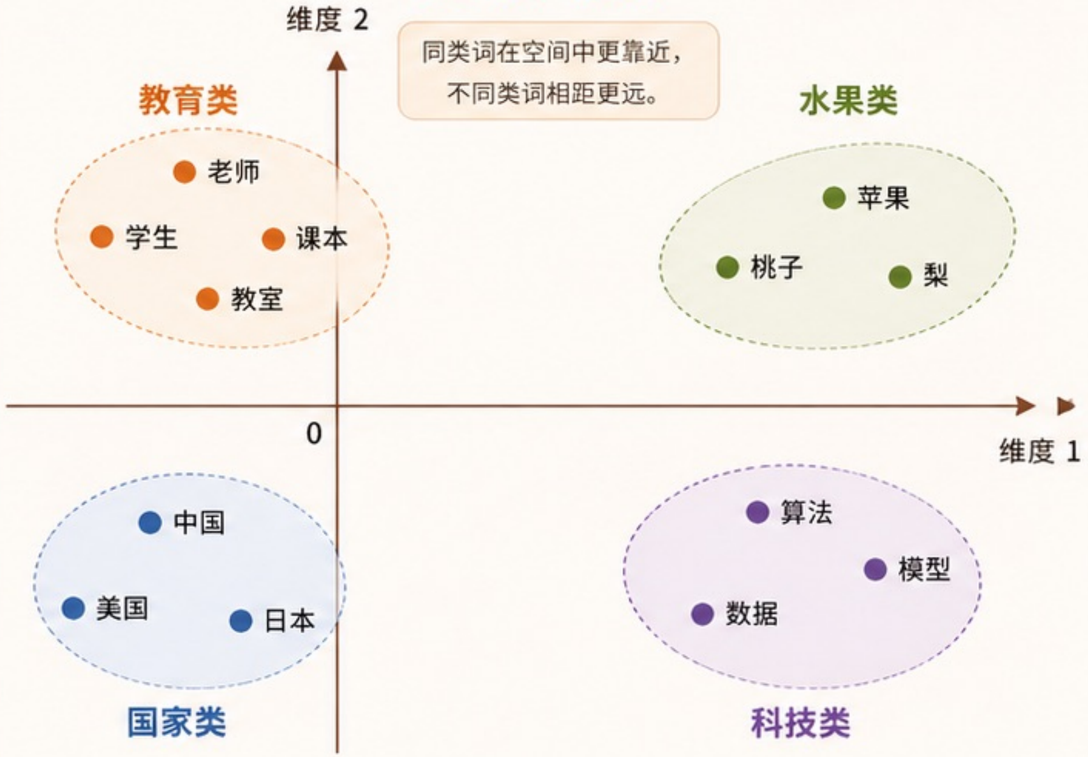

桃子 → [0.47, -0.18, 0.86, 0.28]


梨 → [0.44, -0.10, 0.89, 0.22]

…

向量的维度可以很高，常见如 128、256、768、1536…

语义地图（示意）



它为什么厉害？

1 相近的词，数字也相近

语义相似的词，向量距离更小，更容易被模型识别与迁移。

2 同类的词，会在空间里聚成一团

模型自动发现词与词之间的语义关系，形成自然的簇。

3 距离和方向开始有意义

距离代表相似度，方向代表语义差异或变化趋势。

Embedding 像给词语装上了“语义 GPS”：坐标不只是数字，而是意义的载体。

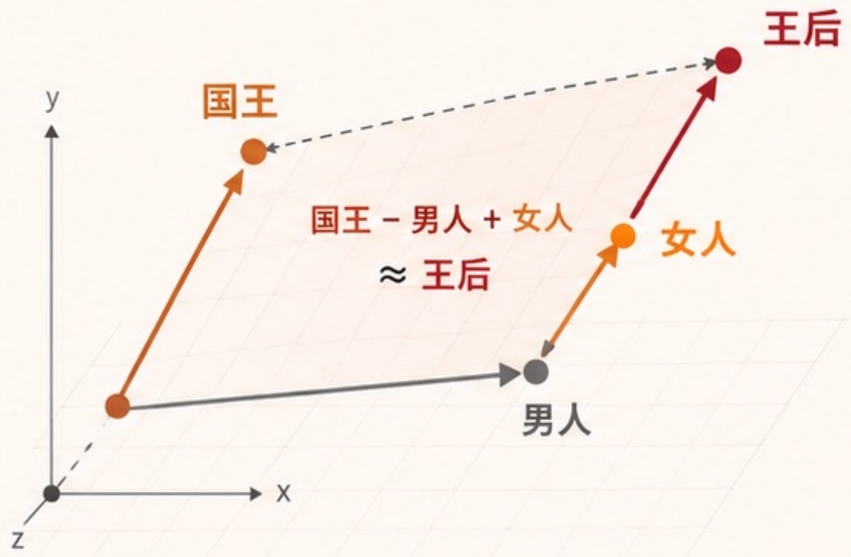
词 → 向量 → 空间位置 → 语义位置

惊人的发现：词向量能做加减法

语义关系不仅能表示出来，还能变成空间里的方向

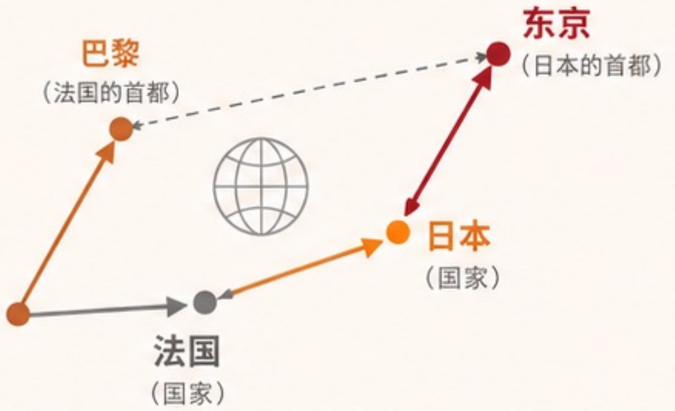
例子 1：性别关系（王 − 男 + 女 ≈ 王后）

国王 − 男人 + 女人 ≈ 王后



例子 2：首都关系（国家 − 法国 + 日本 ≈ 东京）

巴黎 − 法国 + 日本 ≈ 东京



同一关系 = 同一方向
只是起点不同，终点随之变化

这说明了什么？

1. 性别、首都、时态等关系可被编码成“方向”
2. 词向量空间**不是乱排的**
3. 意义被几何化了，加减法变成了**向量运算**

NLP 最美的发现之一

机器不仅学会了词，
还学会了词与词之间的**结构关系**。

从这一刻开始，语言中的关系第一次被看成了空间中的几何关系。

Embedding 是学出 来的——它解释了好多现象

语义地图不是人画的，而是在训练中自己长出来的

从随机到有序

A 训练前

词向量是随机的，没关系可言

反复预测
下一个词
训练

B 训练后

语义相近的词自然靠在一起

它解释了哪些现象？

1 为什么听得懂换个说法？

近义词在语义空间里很近

班主任 ≈ 主班老师
素质 ≈ 能力

2 为什么能跨语言？

不同语言里，意义相近的词会靠在一起

老师 ≈ teacher
苹果 ≈ apple

3 为什么能做向量检索？

在语义空间里找最近的点，就是找最相关的结果

搜索 = 去地图上找最近的点

一句话记住

没人手工画过这张语义地图；它是在海量文本中，**靠训练自己挪出来的。**

这正是深度学习的核心精神：
规律来自数据，不是人手写。

为什么「猜下一个词」能学会推理？

一个看似简单的目标，为什么会逼出推理能力？

一个典型例子

小张今年 28 岁，他的姐姐比他大 3 岁。三年前，他姐姐 ____ 岁。

模型需要这样一步步“想明白”

1 小张 28 岁

今年 28 岁

已知信息：
小张今年 28 岁。

2 姐姐现在 31 岁

比他大 3 岁
28 + 3 = 31

推理关系：
姐姐比他大 3 岁 → 现在 31 岁。

3 三年前要减 3

时间回退 3 年
31 − 3 = 28

时间推理：
三年前，姐姐的年龄减少 3 岁。

4 答案 = 28 岁

所以答案是 **28 岁！**

得出结论：
三年前，他姐姐 28 岁。

关键洞察

要猜对空格里的词，模型必须先把隐藏在句子里的**逻辑想明白**。

理解条件　建立关系　完成推理

本节 takeaway

所谓「预测下一个词」，很多时候等价于：**先理解、再推理、再生成。**

先理解 → 再推理 → 再生成

第二部分 · 预测下一个词

深刻任务，无处不在

当模型在全互联网文字上练习「预测下一个词」，它会被迫学会很多能力。

1 数学

理解数字与运算关系，
学会计算与推理。

例子　3 + 5 = __　→　要会算

2 科学

理解因果与自然规律，
建立对世界的认知。

例子　海水变咸，是因为河流把岩石里的盐带进海里，且盐不会__　→　蒸发

3 代码

理解代码结构与逻辑，
学会按规则解决问题。

例子

```
for i in range(10):
    print(i)
```

→　预测下一行要懂逻辑

很多复杂能力，
都是「猜对下一个词」
的副产品。

4 翻译

理解不同语言的对应关系，
学会跨语言表达。

例子　I love you.　→　我__你　→　爱

5 总结

理解长文本的核心内容，
提炼关键要点。

例子　这段文章主要讲……　→　要抓主题

6 对话/写作

理解语境与意图，
生成合适的语言表达。

例子　给老师写一封请假邮件　→　要懂语气与结构

读得越多、练得越久，模型学到的就不只是语言表面，而是隐藏在语言里的知识与规律。

第二部分 · 预测下一个词

压缩即理解

模型不是在死记硬背，而是在把海量文字压缩成参数里的规律。

互联网海量文本 / 几万亿词

来源广泛： 网页、书籍、对话、论文、代码、百科…

规模巨大： 几万亿词（Tokens）

训练 / 压缩 / 提炼规律

模型参数 / 几千亿参数

体积巨大缩减： 从几万亿词 → 几千亿参数

本质： 保留能用于生成的规律，丢弃冗余细节

1 为什么必须压缩？

- **模型容量远小于原始数据量**
  如果不压缩，存不下，也用不了。

2 压缩的结果是什么？

- **把能生成文字所需的规律存下来**
  这些规律就是统计关系、语义关联、结构模式等。

3 为什么这近似于理解？

- **能稳定预测，说明内部已抓住结构**
  虽然不是意识，但在功能上等价于“懂了”。

> “一个能精准预测下一个词的模型，内部必然**压缩了生成这些文字所需要的知识**。”

教学提示 这里说的「理解」是**功能意义上**的理解，不等于人类的意识与体验。

第三部分 · Attention

Transformer 的心脏——Attention

理解了 Attention，就抓住了今天大模型最核心的技术直觉。

解决问题 1：记不住远处

长句子前面的信息容易被遗忘。
Attention 可以直接关联任意位置，
远近信息都能被“记住”。

关键词：动态关注

不是平均看，而是按需分配关注。
不同的词会关注句子中不同的部分，
上下文随情况而变。

动态计算相关性，
把“注意力”分配给最重要的信息。

解决问题 2：不知道谁和谁相关

句子中哪些词真正相关？
Attention 学会计算相关性，
让模型知道“该看谁”。

作用：为 Transformer 提供理解句子的能力

让模型真正“理解”关系、指代、语义，
从而在问答、翻译、推理等任务中
表现出色。

旧方法：N-gram / RNN
看得短、串行、易遗忘

新方法：Attention
可回看、可并行、会分配关注

如果这一讲只记住一个技术名词，请记住：Attention。

人是怎么读句子的：注意力会「跳回去」

Attention 的直觉，先从我们自己读句子的方式开始。

我们读这句话时，大脑在多次「回头看」

昨天我在学校食堂吃午饭时遇到了老同学 **李华**，**他** 告诉我 **他从上海回到了深圳工作**，**这** 让我很意外，**因为** 他一直说 **喜欢上海**。

① ② ③

① 读到“他” → 注意力跳回“李华”

这里的“他”是谁？
大脑立刻回到最近的、合理的先行词“李华”。

他 ⇢ 跳回去 ⇢ 李华

作用： 理解“他告诉我”是谁在说话。

② 读到“这” → 跳回“回深圳工作这件事”

“这”指代什么？
大脑回到前面最近且语义上最相关的事件。

这 ⇢ 跳回去 ⇢ 他从上海回到了深圳工作

作用： 理解“这让我很意外”的原因。

③ 读到“因为” → 连接“很意外”与“喜欢上海”

“因为”告诉我们：后面的原因要解释前面的结果。
大脑把两边连起来，建立因果关系。

很意外 ⇢ 建立连接（因果） ⇢ 他一直说喜欢上海

作用： 理解原因与结果的逻辑关系。

刚才你的大脑做了什么？

不是平均看每个词
不会机械地从左到右平铺注意力。

会自动回头找最相关的信息
根据当前词的含义，跳回去找最该看的地方。

会动态分配注意力
把注意力集中在最有用的位置，忽略不重要的部分。

Attention 要模拟的，就是这种「读到当前词时，回头关注最相关前文」的动作。

第三部分 · Attention

Attention 三件套：Query、Key、Value

不急着记英文，先记住三个角色：需求、标签、内容。

用图书馆来理解 Q / K / V

Query（查询）

= “我现在想找什么？” = 需求

代表当前词的需求或关注点。

Key（键）

= “我是什么、我关于什么？” = 标签

代表每个词的特征或类别，用来回答 Query。

Value（值）

= “如果你关注我，我能提供什么信息？” = 内容

代表被关注时，能提供的具体信息或语义内容。

当前词（例如：他） → 拿着 Query 去找（我现在想找什么？） → 比对各词 Key（我是什么、我关于什么？） → 取回对应 Value（给我你真正的内容）

Q 问「我要什么」，K 答「我是什么」， V 给「真正的内容」。

先别死记英文缩写。 真正重要的是：Q=需求，K=标签，V=内容。

第三部分 · Attention

怎么算「谁和我最相关」，再变成注意力

先比相似，再变成一组加起来等于 1 的注意力分配。

句子上下文（从左到右）

昨天　我　在　食堂　遇到了　李华　，　他

当前词（要计算注意力）

Query(他)

候选的前文词（潜在关注对象）

李华	我	食堂	昨天
Key(L)	Key(我)	Key(食堂)	Key(昨天)

要解决的问题：

“他”和哪个前文词最相关？

1 第一步：算相关度

用 Query(他) 去分别和每个 Key 比相似，得到原始相关度分数。

李华	:	很高
我	:	中等
食堂	:	较低
昨天	:	很低

技术上可理解成比相似度（点积）。

Softmax / 归一化

把任意实数分数变成非负，且加和为 1 的分布。

2 第二步：变成注意力权重（Softmax）

将原始相关度通过 Softmax 归一化，得到注意力权重。

李华	0.72
我	0.15
食堂	0.07
昨天	0.03
其他（更早的词等）	0.03

最相关，获得最多关注

所有权重加起来 = 1

3 注意：Attention 不是人规定该看谁，而是模型自己根据内容算出来的。

不同位置，结果会变

内容变了，结果会变

越相关，分到越多关注

所谓注意力，就是：当前词把有限的关注，按比例分给最相关的上下文。

把信息加权汇总：得到「他」的真正含义

有了注意力比例，下一步是把各词的内容按比例混合回来。

「他」的新含义 = 0.72 × 李华的内容 + 0.15 × 我的内容 + 0.07 × 食堂的内容 + ……

像调一杯鸡尾酒 / 混合果汁

为什么是加权汇总？

因为不是只看一个词，
而是综合所有相关上下文。

为什么「他」最后主要指李华？

因为李华的权重最大。

更新后的词义是什么？

它不再只是一个孤立的“他”，
而是带着上下文理解过的“他”。

Query 找相关
当前词向其他词提问，找出相关的词。

→ **算权重**
计算每个词的重要程度（注意力比例）。

→ **Value 按比例汇总**
把各词的内容按权重加权汇总（混合）。

→ **当前词获得新含义**
得到融合了上下文信息的新表示（新的含义）。

Attention 的本质：每个词都用上下文，重新理解自己。

第三部分 · Attention

Self-Attention：一句话里，词与词互相投票

Q、K、V 全都来自同一句话内部，所以叫「自注意力」。

每个词都在给别的词投票

来自句中其他词的“投票”（注意力权重）

他（被关注的词）

李华	我	食堂	昨天	……
72%	15%	7%	3%	

投票结果

他

主票由 **李华** 来定义

为什么叫 Self-Attention？

因为信息不是去外面查，而是在句子内部互相参考。

它解决了什么？

指代、搭配、因果、语义关系都能在一句话里同时捕捉。

为什么它能并行？

所有词的“投票”可以同时进行，不必像 RNN 一样排队。

RNN：一个接一个读

昨天 → 我 → 在 → 学校 → 食堂 → 遇到 → 老同学 → 李华 → ， → 他 → 告诉 → 我 → ……

信息只能从前面一步步传过来，**距离越远越难记住**。

对比

Self-Attention：所有词同时互看

任意两个词可以**直接互看**，长距离关系也能轻松捕捉。

一句话里，每个词都在观察别人，也被别人观察。

第三部分 · Attention

两个补丁：位置编码 + 多头注意力

Attention 很强，但还需要两个关键配件，才更完整。

补丁一：位置编码

“我打你” ≈ “你打我”

词几乎一样，但顺序变了，意思就完全不同。

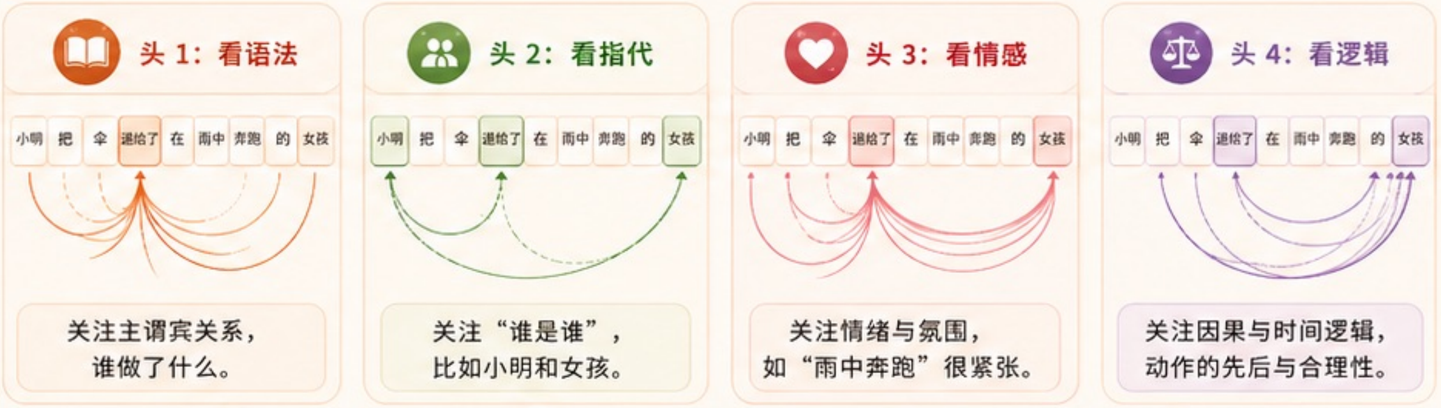
解决办法：给每个词安排“座位号”（位置）

位置编码 = 给每个词额外加上「它排在第几位」的信息。

模型既知道词是什么，也知道它在第几位。

补丁二：多头注意力

小明把伞递给了在雨中奔跑的女孩。



多头汇总：更丰富的理解

既知道句子结构，也理解人物关系、情绪氛围和逻辑链条。

像好几位老师同时批阅：一位看语法，一位看指代，一位看情感，一位看逻辑。

多头注意力 = 多角度同时理解一句话。

位置编码让模型知道顺序；多头注意力让模型同时看见多种关系。

第三部分 · Attention

一句话钉死 Attention

把前面讲过的内容，用一句最值得记住的话压实。

Attention = 让句子里每个词，动态地关注其他所有词，再按相关度把信息汇总过来。

伟大之处 1：看得远

每个词都能直接看到任意远处的词，长距离信息不容易衰减。

任意距离，直接相连，信息不衰减

伟大之处 2：自己学会看谁

看谁、看多少，由模型自己学习，而不是人工设定。

模型自动学会：关注“李华”最多，其它词少一些。

旧方法（N-gram / RNN）

- ✕ 看得短：只能看相邻或有限窗口的词
- ✕ 容易忘：距离越远，信息越难保留
- ✕ 串行处理：一个词一个词地顺序计算

新方法（Attention）

- ✓ 可回看任意位置：任意远的词都能直接连接
- ✓ 自动分配关注：模型学习谁重要、重要多少
- ✓ 并行计算：所有位置同时计算，更高效

如果这一部分只记住一句话，请记住：

Attention 让词与词建立动态关系，并把相关信息重新汇总成新的理解。

第四部分 · 把心脏装进整台机器

Transformer：把 Attention 装进整台机器

Attention 是心脏，但一台完整的模型还要靠一整套积木块协同工作。

多头自注意力：
词与词互相交流
每个词同时“关注”到
其他所有词，收集信息。

残差连接：
给信息和训练信号一条直通道
把输入直接加到输出上，
信息与梯度更容易传递。

GPT-3：96 层；现代模型常常上百层

前馈网络：
每个词自己深加工
对每个词独立处理，
提炼更抽象的特征。

归一化：
让训练更稳定
统一分布、抑制波动，
让模型更快更稳地收敛。

接下来三页看什么？

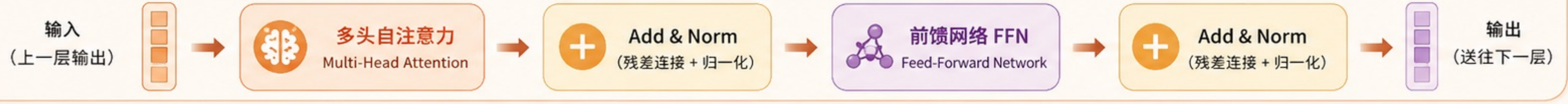
1. 前馈网络（深入理解）
2. 残差连接（为何如此关键）
3. 归一化（让训练更稳定）
4. 叠很多层的意义（深度带来的能力）

第四部分 · Transformer Block

三个零件：前馈网络、残差连接、归一化

Attention 不是全部；完整的 Transformer Block 还需要深加工和稳定器。

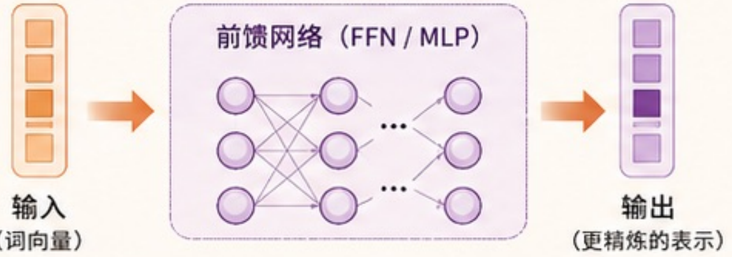
一个完整的 Transformer Block（四者缺一不可）



1 前馈网络 FFN

作用：让每个词自己再深加工一遍

对比：Attention 负责交流，FFN 负责消化

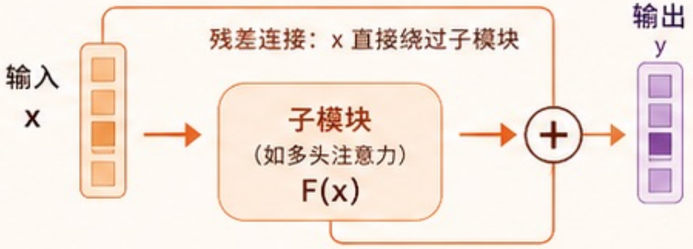


在很多现代模型里，
FFN / MLP 往往占参数大头，
大量事实知识常藏在这里。

2 残差连接

作用：让输入绕过模块，直接加到输出上

结构示意（以注意力子层为例）



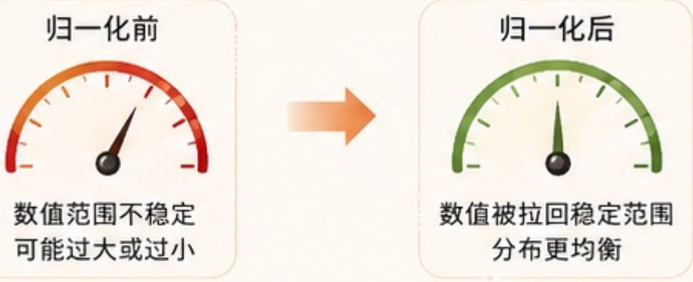
两大好处

- 原始信息不容易丢失
- 训练信号有高速公路，深层网络更容易训练

3 归一化

作用：把层层传递的数值拉回稳定范围

数值分布示意（以 LayerNorm 为例）



避免数值越来越大或越来越小
让训练更稳，更不跑偏

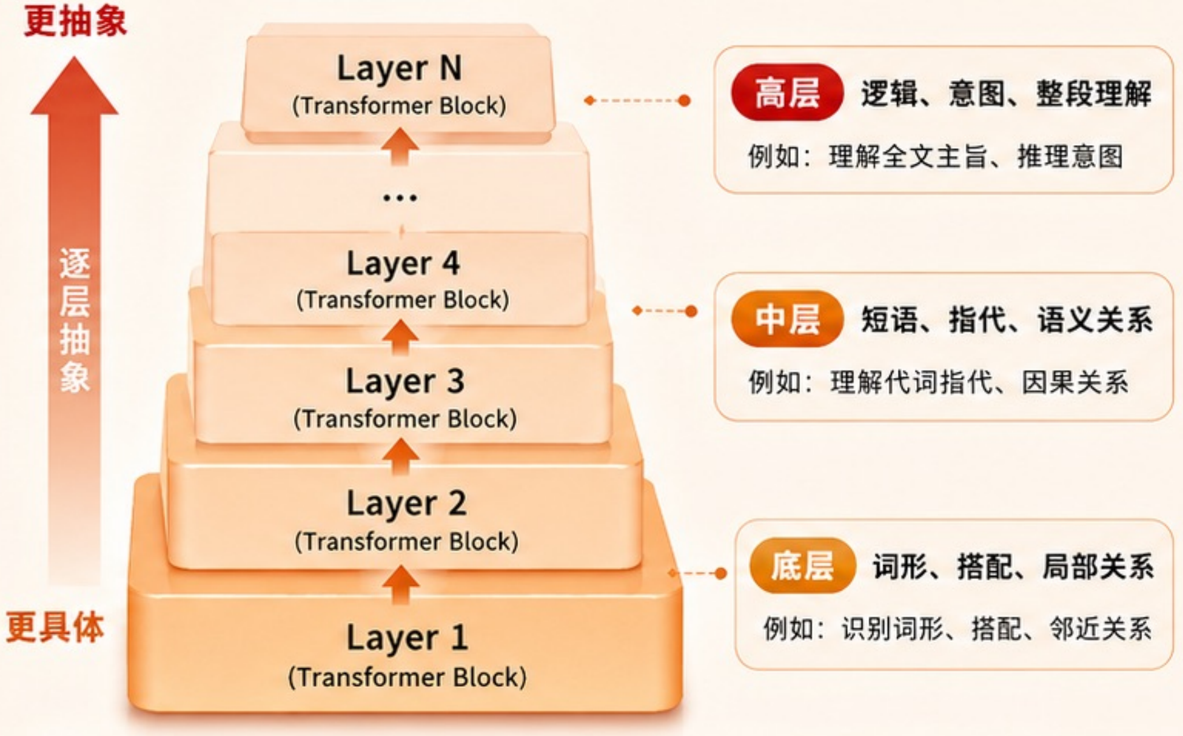
Attention 是开会讨论，FFN 是会后各自消化，残差与归一化是让整场会议越开越清醒。

第四部分 · 深层结构

叠很多层 + 为什么现代大模型都长一个样

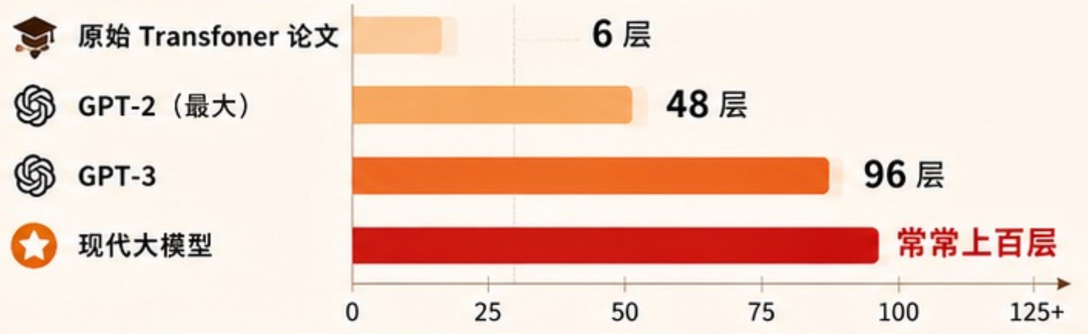
真正的威力，不只来自一个 Block，而来自几十上百层的堆叠。

深层堆叠：层数越深，表示越抽象



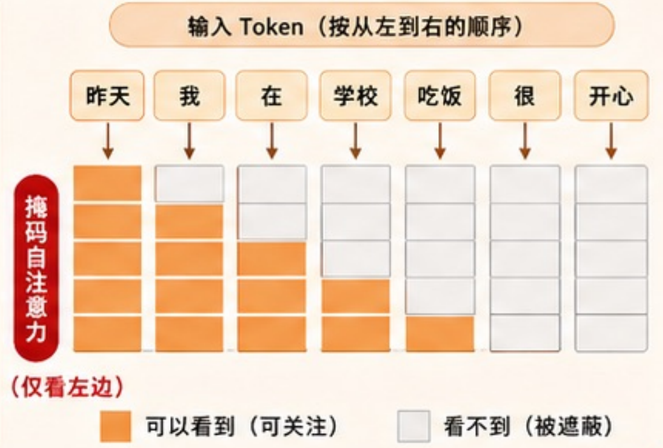
层数越多，模型能表达的层次越丰富，理解越深刻。

主流模型的层数对比（示例）



为什么现代大模型都长一个样？

- 只用解码器（Decoder-Only）
- 每个词只能看到前面的词，看不到后面的词
- 这正好匹配“预测下一个词”
- 因此 GPT、Claude、DeepSeek、Qwen、文心等主流模型，骨架几乎是**同一套 DNA**



现代大模型的差别，更多在规模与训练，而不在底层骨架。

第四部分 · 生成流程

从一个词到一段回答：完整走一遍

把「为什么海水是咸的？」放进去，看看模型怎样一步步生成回答。

1 “切碎”

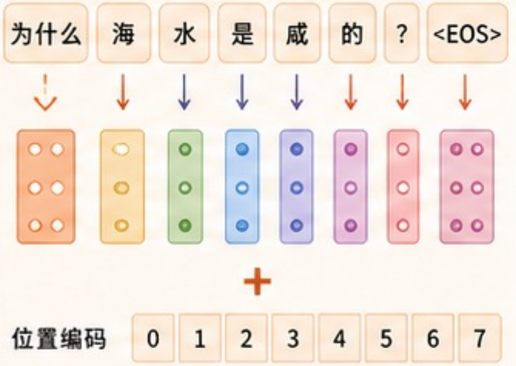
Tokenizer 把句子切成一个个 token。

为什么海水是咸的？

为什么	海	水	是
咸	的	？	<EOS>

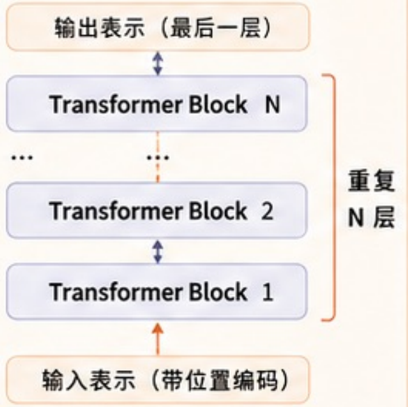
2 “变坐标”

每个 token 变成向量，并加入位置编码。



3 “过 N 层”

序列依次通过多层 Transformer Block。



4 “出概率”

最后一层输出下一个 token 的概率分布。

（数值是示意）

5 “选一个词”

按策略（如取概率最高）选出一个 token。

本轮选择：

海

（因为 40% 最高）

6 “接回去再来”

把选中的词接到末尾，重新进入步骤 1，再生成下一个词。

接回去后的序列：

为什么	海	水	是
咸	的	？	海

（新接的）

再次从步骤 1 开始：预测下一个词 → “水”

下一个选择：水

自回归生成：
吐一个词 → 接回输入末尾 → 再预测下一个词

直到生成 EOS（结束符），模型才停止。

工程上常用 KV 缓存复用前文计算，所以前文越长，生成通常越慢、越贵。

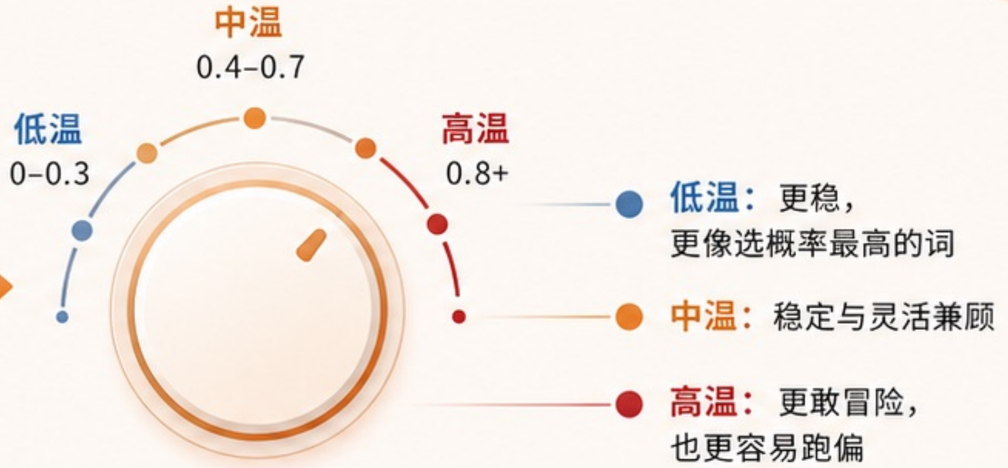
屏幕上一字一句蹦出来的回答，本质上就是这条流水线反复跑了很多圈。

选词的旋钮：温度与采样

同样一组候选词，选法不同，回答的气质就不同。

输入示例

请写一句关于春天的话



贪心（总选最高）

稳定、确定，但容易呆板重复

示例结果：春天很温暖。

采样（按概率抽签）

更有惊喜与创意，偶尔也会抽到怪词

示例结果：春天像一场明媚的苏醒。

常用折中

通常只在高概率的少数候选里抽签

示例结果：既自然，又保留一点灵活

低温更适合： 事实问答、翻译、写代码；**高温更适合：** 故事、头脑风暴、创意写作。

不用碰参数，也可以用自然语言调这个旋钮：想更活泼，就说「多一点创造性」；想更严谨，就说「更严谨些」。

上半场小结：你的一句话，走完了整台机器

从输入一句话，到一个词一个词生成答案。

1 输入问题

为什么
海水是咸的？

2 切成 Token

文字先被切成
一块一块

3 查 Embedding + 位置编码

每块 token 变成
带语义与顺序的数字

4 穿过 Transformer

Attention：
回头看谁最相关

前馈网络：
再深加工

残差与归一化保证稳定

5 输出下一个词概率

6 按温度选一个词

7 自回归 继续生成

……直到形成
完整回答

Token
把文字切成最小单位

Embedding
数字向量，带语义与位置

Attention
回头看谁最相关

自回归
用前面的词，预测下一个词

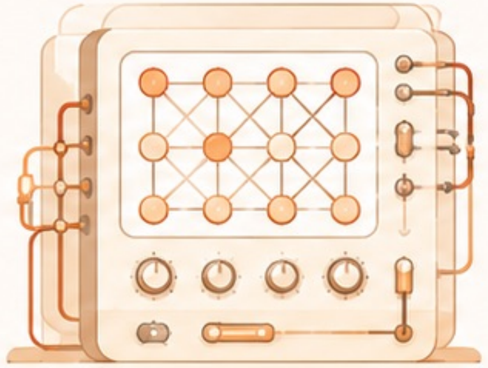
没有魔法，没有百科查询，只有一条流水线一遍遍地『**预测下一个词**』。

你现在已经懂了『它怎么工作』；下半场，我们转向另一个问题：它是怎么长大的？

会聊天的它，当初什么都不会

一开始，模型内部只是大量随机数字。

刚搭好的机器



里面全是
随机参数

问它问题，
只会吐出乱码

怎么
长大的？

今天的聊天助手

你好！有什么
我可以帮你的吗？

帮我解释一下“熵”的概念，
并举一个生活中的例子。

当然可以！熵是……
生活中可以这样理解……

能解释

能对话

能辅助思考

1 预训练

读万卷书

先在人类海量文字中学语言与知识

2 监督微调 SFT

拜师学规矩

学习如何把问题回答得更像助手

3 对齐

在反馈中懂事

朝着有用、无害、诚实去调整

从『随机数字』到『会聊天的助手』，关键就在这三阶段训练。

三阶段全景：像养一个孩子

从读万卷书，到学规矩，再到学会懂事。

1 2 3

预训练

读万卷书

- 输入：海量网页、书籍、代码、论文
- 任务：不断预测下一个词
- 学到：世界知识 + 语言能力

先广泛读书

监督微调 SFT

拜师学规矩

- 输入：大量高质量问答示范
- 任务：学会按提问来回答
- 学到：回答姿态、格式与助手风格

再学会回答

对齐

在反馈中懂事

- 输入：人类偏好与反馈
- 任务：让回答更有用、更安全
- 学到：有用、无害、诚实

最后学会分寸

为什么不能只做一步？

只预训练：
有学问，但不太会好好对话

只到 SFT：
会回答，但还可能说错或说得不妥

做完对齐：
才更像真正可用的聊天助手

三阶段缺一不可：知识、回答方式、行为边界，分别在不同阶段形成。

预训练：在人类文明的全部文字里泡几个月

规模巨大，而且靠的是『自监督』。

它读什么？

- **网页**：新闻、博客、文章、网页内容…
- **百科**：词条、知识库、百科全书…
- **书籍**：小说、教材、历史、传记…
- **代码**：开源代码、脚本、文档…
- **论文**：学术论文、研究报告…
- **论坛/问答**：贴吧、论坛、问答社区…

基础模型
(Base Model)

互联网级别的文字海洋

它怎么学？

海水之所以咸，是因为河流把 ____ 带进了海洋。

1 盖住下一个词

海水之所以咸，是因为
河流把 [ ] 带进了海洋。

2 让模型猜

盐分	0.82 ✓
泥沙	0.07
矿物质	0.05
水分	0.02
……	……

3 和原文对照，错了就调整参数

海水之所以咸，是因为
河流把 **盐分** 带进了海洋。

更新参数，变得更准确

标准答案就藏在原文里

不需要人工逐条标注

这一阶段得到什么？

得到：基础模型（Base Model）

- 读过很多，知道很多
- 语言已经很流利
- 但本质上还是一台『超级续写机』

例子

输入：为什么海水是咸的？

输出（模型可能的表现）：

可能继续续写相关文本，却不一定正面回答你的问题。 …

预训练最厉害的地方，不是人工喂答案，而是让模型在海量文本中自己找规律。

预训练负责『有学问』，但还不负责『会当一个好助手』。

预训练的代价：一个字，烧

养成记 · 预训练（快讲）

1 烧算力、烧电、烧时间

上万张 GPU、相当一个小镇的电、连跑数周到数月。

2 成本常达数百万到上千万美元

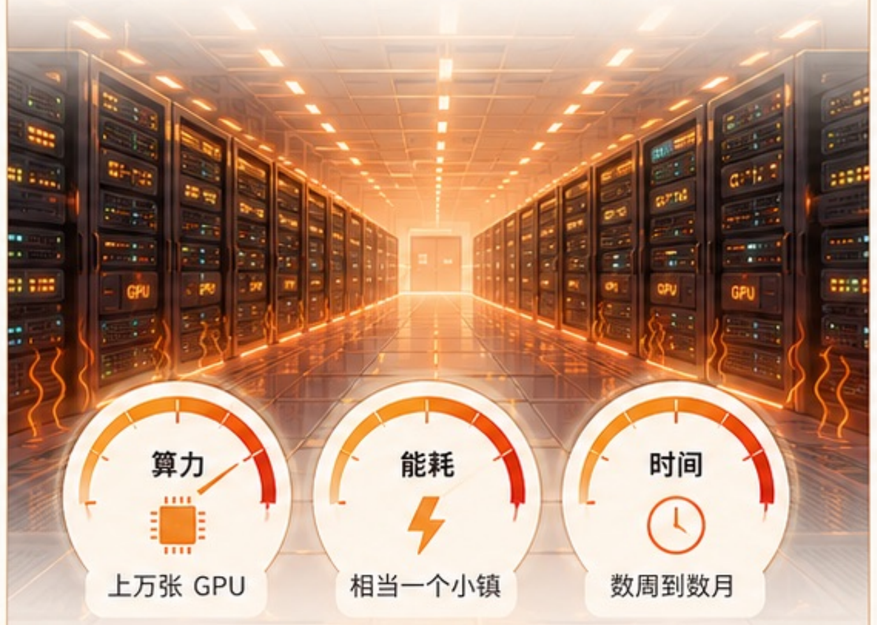
长期是少数巨头才玩得起的游戏。

3 反差故事：DeepSeek

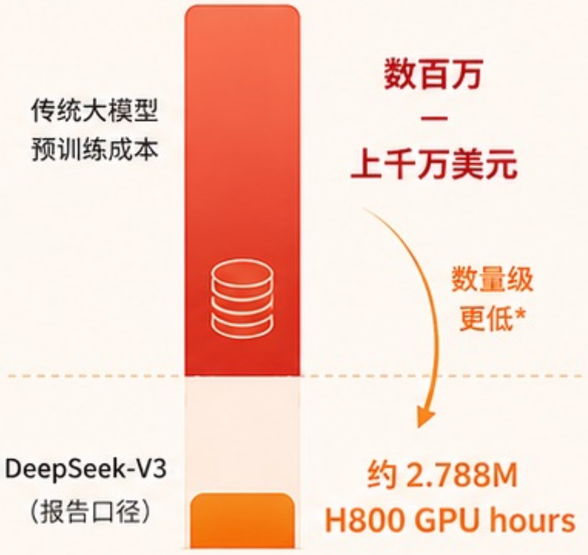
报告口径：DeepSeek-V3 约 2.788M H800 GPU hours；不等于全部研发成本。

预训练 = 超大规模分布式并行训练

数据中心昼夜运转，GPU 全力计算，热量与能耗巨大。



成本对比 （示例，数量级）



*注：为训练 GPU 计算时长口径，不等于全部研发成本。

训练大模型，本质上是在用巨量算力、时间与电力，把语言规律压进参数里。

监督微调（SFT）：从「续写机」到「对话机」

养成记 · 微调

从「续写机」（预训练模型）

只会接着写下去，缺乏指令意识。

为什么海水是咸的？

海水是咸的，这是因为海水中含有大量的盐分，主要来源于河流的冲刷、岩石的风化以及海底火山活动等……

✕ 内容松散、啰嗦，答不到重点

用示范进行训练

监督微调（SFT）

用高质量的「示范问答」教它如何回答。

示范问答（教师数据）

问：为什么海水是咸的？

答：海水是咸的，主要原因有三点：

1. 河流将陆地的盐类带入海洋；
2. 岩石风化释放矿物质离子；
3. 海底火山和热液活动补充矿物质。

因此，海水中积累了多种盐类，其中以氯化钠为主。

……

学会遵循指令作答

到「对话机」（SFT 后）

能按人类期待的方式，清晰地回答问题。

为什么海水是咸的？

海水是咸的，主要有三点原因：

1. 河流将陆地的盐类带入海洋；
2. 岩石风化释放矿物质离子；
3. 海底火山等活动补充矿物质。

因此，海水中积累了多种盐类，其中以氯化钠为主。

✓ 结构清晰、有条理、答到点子上

1 给它看优质「示范问答」

教它面对问题要直接、有条理、答到点子上。

2 质量远胜数量

几万条精挑的示范，胜过几百万条粗糙数据。

3 第一次像样的助手

它会好好回答「为什么海水是咸的」了，但还不够稳。

SFT 的核心，不是让模型多说，而是让它学会按人类期待的方式好好回答。

对齐：让它有用、无害、诚实

养成记 · 对齐

1 目标 HHH

有用、无害、诚实：
帮得上、不作恶、不知道就说不知道。

2 RLHF：用人类偏好来拉

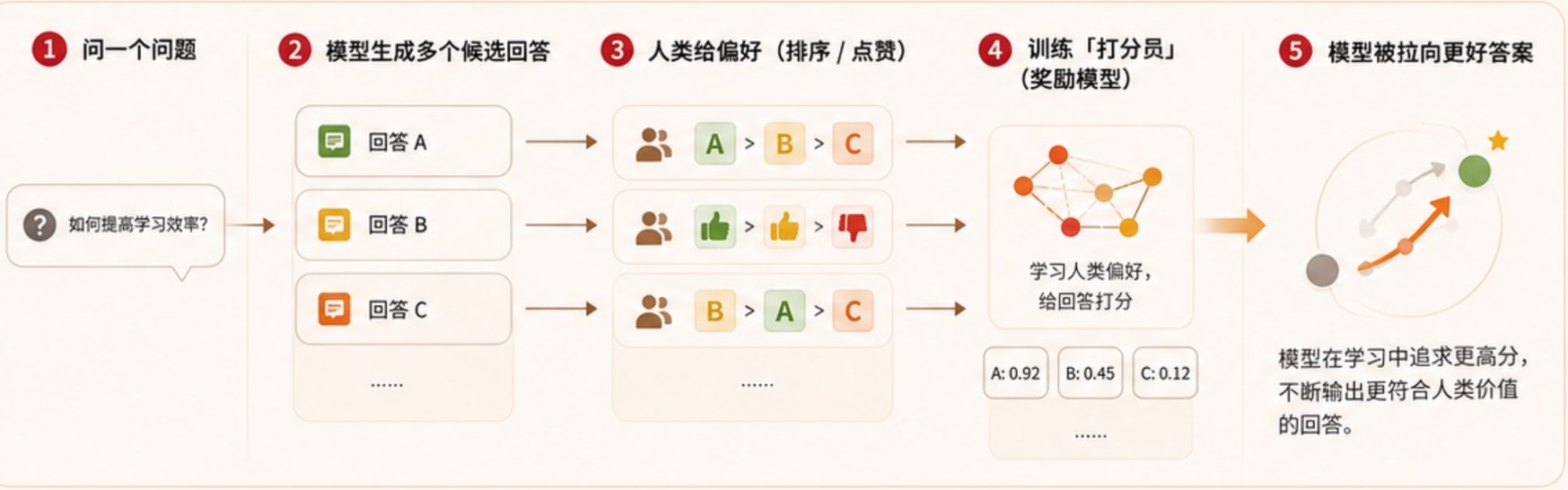
对同一问题的多个回答请人排序，训出「打分员」，模型去争高分。

3 效果惊艳，也有麻烦

从「能用」变「好用」；复杂烧钱，于是有更简的 DPO。

对齐的核心：让模型的目标与人类价值一致（HHH）

- **有用** 帮得上
- **无害** 不作恶
- **诚实** 不知道就说不知道



对齐做的事，是把“会说话”进一步变成“会负责任地说话”。

三阶段收束：一个模型的出生历程

养成记 · 三阶段

1 预训练
→ 微调 → 对齐

读万卷书

大量阅读，学会世界的知识与规律。

2 微调
→ 对齐

拜师学规矩

在少量高质量示例上学习任务与表达方式。

3 对齐

在反馈中长成可靠的人。

借助人类反馈，学会守规矩、更安全、可帮助。

4 出生后仍在成长

持续补数据、修缺陷、出新版本。

5 回到五代 GPT

结构同源，差别在规模与「养法」越来越扎实。

从随机参数到成熟助手，**本质上是三段式成长：学知识、学规矩、学负责。**

涌现：规模大到一定程度，能力突然「冒」出来

养成记 · 涌现（选讲）

1 小模型不会，大到某点突然会

像开关被打开，而非缓慢变好。

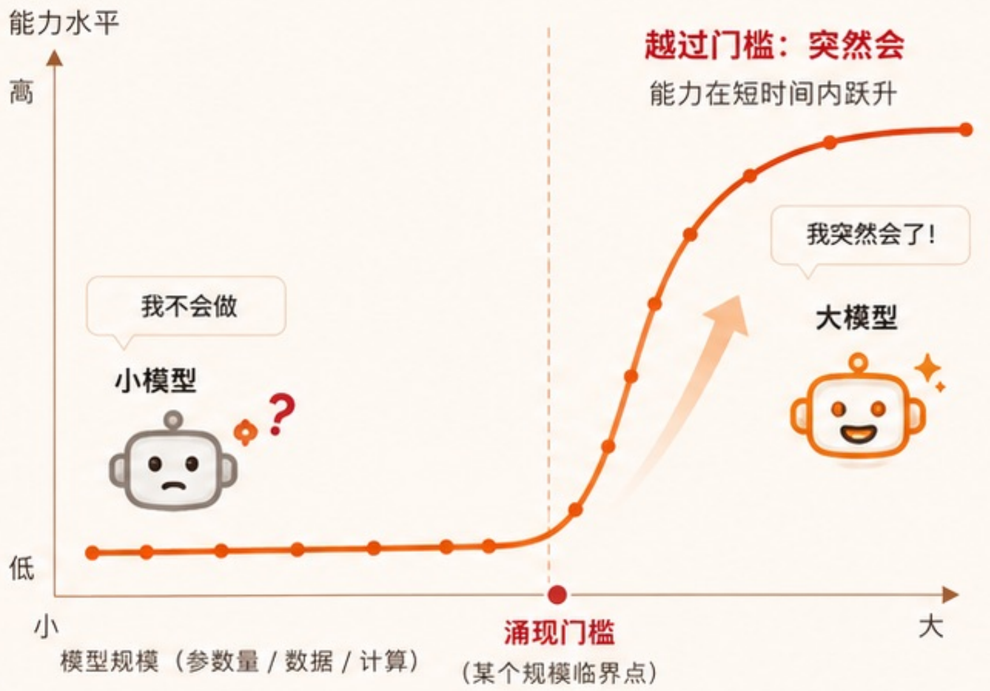
2 三个经典案例

多位数加法、多步推理、
举一反三（给几个例子当场学会）。

3 启发：规模可能带来质变

很多能力是积累到某点忽然开窍——
和孩子成长一样。（有学术争议）

能力随规模的典型变化曲线（示意）



三个经典案例

多位数加法

例：2783 + 5967 = ?

小模型： 常出错

大模型： 突然准确

多步推理

例：经过多步逻辑推导得出结论

小模型： 易中断/出错

大模型： 突然连贯正确

举一反三

例：给几个例子，当场学会新题型

小模型： 只能记住样例

大模型： 能泛化到新情境

有些能力不是慢慢长出来的，而是在规模越过门槛后突然显现。

推理模型：让 AI 学会「先想再答」

养成记 · 推理模型

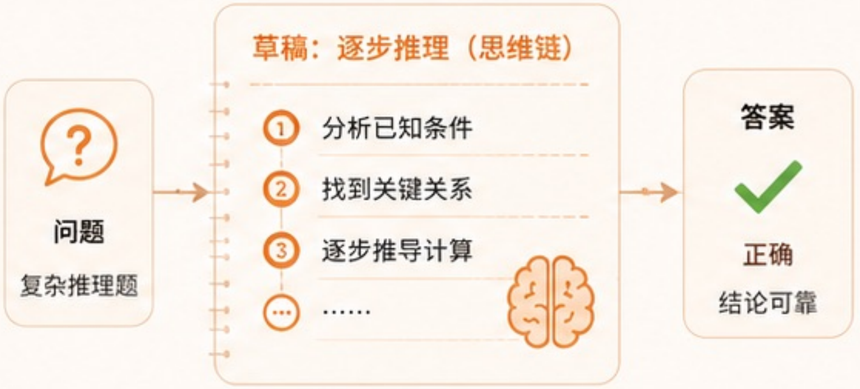
1 普通模型 = 脱口而出

看到问题直接吐答案，难题易错。

速度快，但容易被表面信息误导。

2 推理模型 = 先打草稿

回答前把思路一步步写出来，再给结论，错误率大降。



先想清楚，再给答案，准确率显著提升。

3 代价：慢、贵

看菜下饭——查事实用普通模式，解难题才开「深度思考」。

查事实 / 简单问答 → 普通模式

复杂推理 / 难题攻克 → 深度思考（推理模型）

更慢
需要更多时间

更贵
消耗更多算力

推理模型的关键，不是知道更多，而是愿意多想几步再作答。

MoE：把一个大模型拆成一群「专家」

养成记 · MoE（拓展）

1 普通模型：每题全员上

调动全部参数，又慢又贵。

2 MoE：分诊台 + 专家

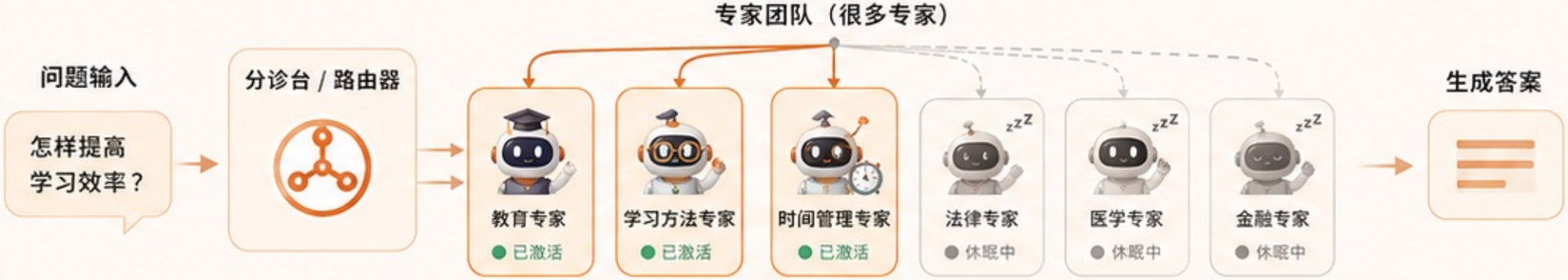
路由判断该归谁管，
只叫醒相关的少数专家。

3 总参数大、实际干活的小

总容量大、每次只叫醒少数专家。

普通模型：每题全员上 调动全部参数、又慢又贵。

MoE：分诊台 + 专家 路由判断该归谁管，只叫醒相关的少数专家。



成本对比（示例）	训练成本降低	推理成本降低	推理速度提升
	30% ~ 50%	50% ~ 80%	1.5 ~ 3 倍

* 以上为常见范围，实际效果随任务与模型而异。

MoE 的聪明之处，是让模型**总容量很大**，但每次只动用其中**一小部分**。

幻觉：它为什么「一本正经地胡说」，怎么办

养成记 · 幻觉

1 为什么会编

它追求「读起来可信」，
不是「事实正确」；
不知道也不会停下。

2 老师的铁律

事实 / 数据 / 引文
必须自己核实；
越具体的细节越要警惕。

3 三步核查

标事实 / 建议 → 查人名
年份数据引文 → 教师定稿。

AI 的自信回答（可能带幻觉）

根据最新研究，2023 年《清华管理评论》发表的文章《生成式 AI 对组织创新的影响》指出：…（省略）

可以参考下列文献：

李明, 张华. 生成式 AI 对组织创新的影响[J]. 清华管理评论, 2023, 45(3): 12–28. DOI: 10.1234/tjmr.2023.03012

看起来非常专业，但这可能是编造的引文。

人工核查清单（实践版）

1 标事实 / 建议
先标出模型给出的关键事实、数据与引文。

2 查人名年份数据引文
逐一核对作者、年份、期刊 / 会议、DOI / 链接、引用内容是否真实存在。

3 教师定稿
由教师审核、修订、定稿，对外发布或提交。

AI 可以帮你起草，但**事实核查**与最终把关，仍然必须**由人来完成**。

本讲小结

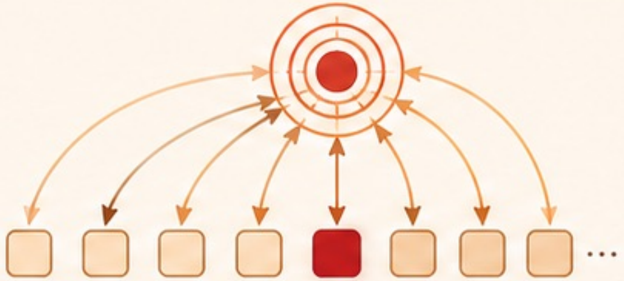
三句话带走

1 大模型干的就是「预测下一个词」

规模与训练到位，
就涌现出推理、创作、对话
的全部本事。

2 Transformer 的灵魂是 Attention

让每个词动态关注前文
最相关的部分，无视距离。



3 模型像孩子一样被「养」出来

读万卷书 → 拜师学规矩 →
在反馈中懂事。

如果只带走三句话：预测下一个词、Attention、三阶段训练。

收束

概念地图 + 给老师的一段话

这台机器是如何「想」与「说」的

Token
把文字切成
基本单位

→

Embedding
把 Token 变成
数字向量

→

Attention
决定“看哪里”
与“重视谁”

→

Transformer
用多层结构
综合信息

→

生成
预测下一个 Token
直到说完

自回归生成：用已有的内容，预测下一个 Token

这台机器是如何「学」与「成长」的

预训练
海量数据学习
通用语言规律

→

微调
在特定任务上
学会更好完成

→

对齐
让模型更安全、
更可控、更有用

→

涌现
能力在规模中
突然出现

推理
更长思考链，
更强理解力

MoE
专家分工协作，
更高效扩展

幻觉
不确定时也会
生成似是而非

给老师的一段话

一句心里话

二十年后，
学生会跟你聊 Attention、
聊涌现、聊对齐——
那时，你聊得上。

你已经做到了

不必成为专家，
但你已经听得懂
他们在说什么。

今天你**不必成为专家**，但已经能看懂这台机器的大致运行方式与**成长路径**。