AI 是怎么学会说话的

从一串数字到会写诗的机器：大语言模型工作原理

百姓AI

费曼式写法 · 由多个 AI 代理撰写与互相审校

导读

从一串数字到会写诗的机器：大语言模型工作原理

先别管神经网络

在翻开任何一页之前，我想先请你回忆一个遥远的下午：你一两岁的时候，坐在大人怀里，他们一遍遍指着灯说「灯」，指着狗说「狗狗」。没有人给你讲过语法，没有人给你下过定义，可几年之后，你不但会说话，还会顶嘴、会撒谎、会讲没人教过你的新句子。

语言这件人类最骄傲的本事，来的时候没有说明书。

这本书要讲的，是另一台「孩子」：一台只会做加法和乘法的机器，是怎么在读了人类几乎所有的文字之后，学会说话的。它能跟你聊人生、写情诗、改代码、一本正经地分析章鱼墨水治感冒时药店该怎么摆货架——而它干的全部工作，说出来只有一个：**猜下一个词。**

一个只会猜词的机器，为什么看起来像在思考？这就是贯穿全书的问题。

这本书怎么带你走

我们不背公式，不堆名词，只沿着一条因果链往下走：文字怎么变成数字，数字怎么带上意思，「学会说话」怎么被压缩成一场可以打分的考试，机器怎么在句子里抓住重点，它脑子里几十亿个旋钮怎么被自动拧到位，为什么大到一定程度它会突然「开窍」，以及——最后，也是最重要的——它到底有没有在思考，我们该怎么跟它相处。

每一章都可以单独读，但十章连起来是一条线：从「一台只会算数的机器」到「一个会写诗的伙伴」，中间那几座桥，我们一座一座搭给你看。

不需要任何预备知识。带上好奇心，从第一章开始就好。

第一章 · 一台只会算数的机器

先把所有神奇的东西拿走

在正式开讲之前，我想先请你做一件反直觉的事：把「智能」「理解」「思考」这些大词，全部从脑子里请出去。我们只从一个无聊到极点的事实出发——

计算机只会做一件事：按固定的规则，搬运和计算数字。

这不是谦虚，是字面意思。你手机里的芯片，每秒能做几十亿次操作，但每一次操作都逃不出这几样：把两个数加起来、乘起来、比一比谁大、把一个数从这里搬到那里。你刷的短视频、打的游戏、听的歌，拆到最底层全是这个。屏幕上的小猫视频，在你眼里是猫，在芯片眼里就是几千万个数字，按某种约定排列着——哪几个数字代表哪个像素的红绿蓝，而已。

那么问题来了：这样一台只会算数的机器，是怎么学会**说话**的？你跟 ChatGPT 聊天，它回答得头头是道，还会写情诗、改简历、讲笑话。这中间到底发生了什么？

这就是这本书要回答的问题。我们不背定义、不堆名词，就沿着一条线往下追：**从「搬数字」到「会说话」，中间到底搭了几座桥？**

程序：一张写得极细的说明书

先说清楚传统意义上的「程序」是什么，因为 AI 恰恰是对它的一次背叛。

一个程序，说白了就是你写给机器的一张**说明书**，细到令人发指的那种。比如你要让电脑帮你算全班平均分，你得写：先把这 50 个数全加起来，再除以 50，把结果显示在屏幕上。一步都不能少，一步都不能含糊——你要是忘了写「除以 50」，机器绝不会自己悟出来，它只会傻乎乎地把总分甩给你。

传统程序的逻辑是：**人想清楚规则，机器负责执行。**计算器是这样，Excel 是这样，微信抢红包也是这样——每一步都是某个程序员预先写好的「如果……就……」。

大白话

大白话：传统软件就像一个极其听话但毫无主见的实习生。你给的 checklist 上有的，他做得又快又准；checklist 上没有的，他连想都不会想。

这套打法统治了计算机行业几十年，战绩辉煌。但它有一道撞了几十年都撞不开的墙。

那道撞不开的墙：有些事你说不清规则

来看几个看起来 trivial（微不足道）的任务：

- 给你一张照片，告诉我里面有没有猫。
- 把「我喜欢这本书」翻译成英语。
- 读一段留言，判断这个顾客是不是在骂人。

这些事，三岁小孩都会。但请你试着把它们写成一张「如果……就……」的说明书？

「如果照片里有两个尖耳朵、几根胡须、一条尾巴，就是猫」？那背对着你的猫呢？只露出半个身子的猫呢？卡通猫呢？一只瘦得没型的橘猫和一只胖得滚圆的橘猫，像素上几乎没有共同之处，可你一眼就知道它们都是猫。

「喜欢」翻译成 like？那「我喜欢这本书很久了」里的「喜欢」和「他这个人挺讨人喜欢的」里的「喜欢」，语境完全不同。想穷举规则，你会发现规则背后还有规则的例外，例外背后还有例外的例外——上世纪七八十年代，几代最聪明的人沿着这条路死磕了二三十年，最后不得不承认：**这条路走不通。**

这就是传统编程的天花板：**凡是人自己也说不清楚规则的事，就没法写成程序。**而语言、图像、常识，恰恰全都是这种「说得出来、写不出来」的东西。你能认出猫，但你说不清你到底是根据什么认出来的——那套知识存在你的脑子里，却不存在你能写下来的任何地方。

换个思路：别写规则了，让机器自己找

撞墙撞久了，有人换了个问法。原来的问题是：

> 「我怎样把规则写得足够细，让机器照着做？」

换成：

> 「我能不能不写规则，只给机器看海量的例子，让它**自己**把规律找出来？」

这个思路就是机器学习——大白话说，就是**不教规则，只给例子，让机器自己悟**。就像你从来没给孩子讲过「猫的定义」，只是每次见到猫就指给他看：「这是猫。」看个几十次，他自己就会了。孩子脑子里长出来的那套「识别猫的办法」，不是你写进去的，是他从例子里自己淘出来的。

机器学习干的就是这件事：程序员不再写「如果……就……」，而是搭一个**里面有成千上万个旋钮的空架子**，再喂给它海量例子——「这是猫」「这不是猫」「这句话下一句通常接这句」——让它自己慢慢拧那些旋钮，拧到它的输出跟真实例子对得上为止。拧完之后，那套「规则」就有了，但它不是任何人写的，它藏在那几十亿个旋钮的位置里。

这就是整本书的地基。你今天听到的所有 AI 新闻——会聊天的、会画画的、会写代码的——底下都是同一句话：**不是人把规则写给了机器，而是机器从数据里把规律学了出来。**

这本书要搭的几座桥

思路是有了，但从「给机器看例子」到「机器会跟你聊人生」，中间还差着好几座桥。这本书会一座一座地搭过去：

1. 文字怎么变成数字？——机器只认识数，得先给语言「上户口」。（第二章）
2. 数字怎么带上「意思」？——为什么机器能算出「国王和女王的关系，就像男人和女人的关系」。（第三章）
3. 「学会说话」这件事，怎么被压缩成了一个朴素到家的任务：猜下一个词。（第四章）
4. 机器怎么在一句话里「抓住重点」，而不是傻乎乎地逐字往前读。（第五章）
5. 那些旋钮到底是怎么被自动拧到位的。（第六章）
6. 为什么模型大到一定程度，会突然「开窍」。（第七章）

7. 一个只会续写文字的机器，怎么被调教成了会听你话的助手。（第八章）

8. 最后，它到底有没有在「思考」？我们该怎么跟它相处？（第九、十章）

每一座桥都不需要你先修任何课，只需要你带一样东西：好奇心。出发吧。

第二章 · 把词切成数字

第一个麻烦：机器不认字

上一章我们说了，机器只认识数字。那好，你想让机器读一句话，第一个拦路虎马上出现：**「我今天很开心」这七个字，在机器眼里什么都不是。**得先想办法把它们变成数。

怎么变？最容易想到的办法是：给每个字编个号。建一本大字典，「我」是 1024 号，「今」是 3857 号……机器拿到的就不再是文字，而是一串编号，像发电报一样。

这个思路方向没错，但按「字」还是按「词」来编号，差别就大了。而这正是第一个值得停下来想的地方。

按词切，还是按字切？都有坑

先看「按词切」。英语好像天然按词切——空格都替你分好了。但真干起来麻烦不断：`eat`、`eats`、`eating`、`eaten` 是四个词还是同一个词？如果算四个，字典就得把它们全收进去；碰上没见过的新词（网络上每天造出来的梗、某家公司的名字），字典里查不到，机器当场抓瞎——这叫未登录词，大白话就是「户口本上没这人」。更要命的是中文，连空格都没有，「研究生命的起源」该切成「研究/生命/的/起源」还是「研究生/命/的/起源」？切词本身就得先理解语义，成了鸡生蛋蛋生鸡。

再看「按字切」。每个字一个编号，字典小（中文常用字几千个，足够了），永远不会有查不到的字。但代价是：单个字往往没啥意思。「蝴」字单独拎出来是什么意思？它只有跟「蝶」凑在一起才有意义。把意义全推给后面的模型去重新拼，等于白白增加了学习的难度。

按词切，字典爆炸还漏词；按字切，字字都是碎片。两头都不舒服。

实际答案：切成「零件」

工程上真正用的办法，是取一个中间态：**把文字切成常见的「零件」，而不是完整的词，也不是单个的字。**这种零件叫 token（中文常译「词元」），大白话说，就是**机器眼里的最小语言单位**。

怎么切？思路朴素得可爱：**哪些字符组合出现得特别频繁，就把它整个收进字典。**统计海量文本后发现「ing」老是出现，那「ing」就当一个 token；「un」老是出现在开头，也收；「吃」「饭」「今天」「人民」这些高频组合也整个收。而「eating」这种词，就拆成「eat」+「ing」两个 token。

> 大白话
>
> 大白话：这套切法像玩乐高。字典里存的不是所有可能的城堡，而是一批最常用的积木块。常见的东西整块整块地有，没见过的新词也能用小块现场拼出来——字典不大，还永远不抓瞎。

这么一切，效果立竿见影：

- 字典大小可控：主流模型大概收 5 万到 15 万个 token，中英文、标点、代码符号全在里面。
- 新词、网络梗、生僻名字都不再是问题——大不了多拆几块。
- 常见的词整块进出，不用让机器从偏旁部首重新悟意思。

一个直观的量级：对中文来说，一个 token 大约是「大半个字到一个词」之间；日常聊天一句话，通常十几二十个 token。你每次用 AI 按 token 计费，计的就是这个「零件」的个数。

编号只是门牌号，不携带任何意思

切完之后，每个 token 拿到一个编号。到这里，「我今天很开心」在机器眼里就变成了类似 `[1024, 3857, 66, 29013]` 的一串数。

但请注意一个极其重要、后面会反复用到的事实：**这些编号只是门牌号，本身不携带任何意思。**

门牌 1024 和 1025 是邻居，但「我」（1024 号）和碰巧排在 1025 号的那个 token 之间，意思上可能八竿子打不着。编号是人随手发的，跟语义毫无关系。这就好比通讯录里，排在你妈后面的那个人，跟你妈没有任何关系。

所以到现在为止，机器手里只有一串「门牌号」。它能做的全部事情，就是拿着号码去查表、数数、算概率。你说这离「理解语言」还差着十万八千里——没错。**编号解决了「机器**

能存下这句话」，但完全没解决「机器知道这句话什么意思」。

下一步，就是给这些干巴巴的门牌号注入「意思」。怎么注入？答案藏在一个出人意料的地方：让意思变成空间里的距离。这就是下一章的内容。

第三章 · 词语的地图

门牌号有了，但「意思」还没着落

上回说到，一句话切完 token 之后，在机器眼里就是一串门牌号：`[1024, 3857, 66, 29013]`。门牌号有个致命问题：它不带意思。「我」是 1024 号、「你」是 1025 号，纯属发牌时的巧合——从号码本身，机器看不出「我」和「你」关系很近，也看不出「我」和「航空母舰」关系很远。

可语言的核心恰恰就是关系。如果机器连「猫」和「狗」比「猫」和「扳手」更像都感受不到，那它永远不可能学会说话。这一章要解决的，就是这个问题：**怎么让数字自己长出「意思」来？**

一个老语言学家的土办法

答案的思路，早在上世纪中叶就被语言学家想到了。有句名言，大意是：**「想知道一个词是什么意思，就看它平时都跟哪些词混在一起。」**

想想你是怎么学外语的。碰到一个生词「醍醐」，你不查字典，但你发现它老出现在这样的句子里：「醍醐灌顶」「如饮醍醐」……你再想想「灌顶」「如饮」这些词搭配的往往是智慧、美酒、顿悟之类的好东西，你大概就能猜到：醍醐是个好东西，跟精华、妙处有关。你从没见过定义，但光靠「它出没的场合」，就把意思猜了个八九不离十。

> 大白话
>
> 大白话：词的意思不在字典里，在它的朋友圈里。看一个人常跟谁来往，就知道他是什么人；看一个词常跟什么词做邻居，就知道它是什么意思。

这个思路妙就妙在：**「数邻居」是一件纯统计的事，机器最擅长。**不需要人告诉它任何规则，只要把海量文本倒进去，数清楚每个词周围都出现些什么词，「意思」就能从统计里自己浮出来。

把每个词放进一张巨大的地图

具体怎么做？思想实验是这样的：想象一张无限大的地图，我们要给每个词在地图上安排一个家。安排的原则只有一条——

用法相近的词，住得近；用法八竿子打不着的词，住得远。

机器的操作是：先随机发房子（每个词随便站个位置），然后拿海量文本当考卷，反复做一件事——「根据邻居猜中间这个词是什么」。猜错了就挪房子：把搭配对的词拉近一点，把瞎凑对的词推远一点。几十亿次调整之后，整张地图就稳定下来了。每个词最终的位置，用一组坐标记下来——比如 768 个或 4096 个数字——这串坐标就叫这个词的词嵌入（embedding），大白话说，就是**这个词在「意义地图」上的家庭住址**。

注意：没有任何人规定哪个坐标代表什么。地图是机器自己数出来的。但等它数完，人们去参观这张地图，发现里面的布局惊人地有道理。

这张地图上发生了什么

学成之后的词嵌入地图，有几个让人起鸡皮疙瘩的性质：

第一，意思近的词真的住得近。「猫」和「狗」是邻居，「猫」和「老虎」也不远，而「猫」和「扳手」隔着十万八千里。机器从来没人教过它什么是动物，但「都在类似的句子里出没」这一条，就足以让它们聚成一团。

第二，方向也是有意义的。最著名的例子：拿「国王」的坐标，减去「男人」的坐标，再加上「女人」的坐标，结果落点最近的词是——**「女王」**。换句话说，地图上从「男人」指向「女人」的那个方向和距离，跟从「国王」指向「女王」的几乎一模一样。性别、单复数、国家与首都（「中国–北京」约等于「法国–巴黎」），这些关系全都变成了地图上的方向和距离。

> 大白话
>
> 大白话：词的意思变成了「住址」，词和词的关系变成了「怎么走」。语义这件看不见摸不着的事，第一次变成了可以做加减乘除的几何。

第三，这一切完全是自学的。没有语言学家参与，没有词典被录入，就是「数邻居＋挪房子」这一个朴素机制，在足够大的文本上跑了足够多次之后，自己长出来的结构。

为什么「意思变成地址」这么关键

你可能会问：折腾这么一圈，图什么？图的是一个根本性的转变——

从此，「理解语言」被翻译成了「算数」。

门牌号时代，「猫和狗像不像」这种问题机器没法答，因为 1024 和 3857 之间无话可说。地址时代，这个问题变成了「两个坐标离多远」——这是芯片最拿手的活。两句话像不像、这句话该接哪句话、这个词在这个语境里偏哪个意思，全部可以化成坐标之间的运算。

到这一步，机器手里的料就齐了：它不再面对一堆干巴巴的编号，而是面对一张所有词都各就各位的意义地图。每个 token 进门，先查表换成坐标，然后——真正的表演才开始：机器要拿着这些坐标，去做那件听起来简单、实际上撑起整个 AI 时代的事：猜下一个词。下一章见。

第四章 · 就猜下一个词

一个朴素到可疑的任务

前面两章，我们把文字切成了 token，又给每个 token 发了一个带意思的「住址」。现在，主角登场。大语言模型干的全部工作，说出来你可能不信——

给定前面一段话，猜下一个 token 是什么。就这一件事。

不是「理解问题」，不是「组织回答」，不是「思考人生」。就是猜下一个词。你跟 ChatGPT 聊了一下午，它写的每一首诗、改的每一行代码、讲的每一个道理，拆到底层，全是一次又一次「猜下一个词」串起来的：猜一个，接上去，再猜下一个，再接上去，直到说完。

这听起来像个脑筋急转弯：这么简单的事，怎么可能撑起那么聪明的表现？这正是本章要掰扯清楚的。答案是：**「猜下一个词」这件事，难到离谱。**

想猜得准，你得懂整个世界

我们来玩这个游戏。给你前半句，你猜后半句：

> 「今天下雨，出门记得带——」

简单，「伞」。你甚至不用动脑子。但再看这个：

> 「《红楼梦》的作者是——」

要接「曹雪芹」，你光懂语法没用，你得**知道事实**。再看：

> 「小明把果汁倒进空杯子，杯子满了；他又拿起瓶子，发现里面是空的，因为果汁已经被倒进了——」

要接「杯子」，你得在脑内跟踪「果汁在哪儿」这件事的物理变化。再看：

> 「def quicksort(arr): if len(arr) <= 1: return arr; pivot = arr[0]; rest = arr[1:]; return quicksort([x for x in rest if x < pivot]) + [pivot] + ——」

要接下去，你得懂 Python、懂递归、懂快排的算法逻辑。

看出来了吗？「猜下一个词」是个无底洞：**想把各种文本里的下一个词都猜准，语法、常识、事实、逻辑、代码、人心，你全都得会。**一个能把猜词游戏玩到满分的考生，实际上就是把这个世界的规律压缩进了自己的脑子里。这就是大语言模型的核心洞见——

大白话

大白话：教机器「理解世界」这件事没法直接教，但「猜下一个词」是个可以打分、可以海量出题的考试。而把这场考试考好所需要学会的东西，恰好约等于理解世界。考试是手段，压缩世界知识是结果。

它不猜一个死答案，它给一堆候选打分

再具体一层。机器猜词的时候，不是拍脑袋蹦出一个字，而是给词表里**每一个** token 算一个「可能性」：接「伞」的可能 70%，接「雨衣」的 15%，接「防晒霜」的 0.001%……加起来正好 100%。这种「给所有候选各打一个分、分数加起来是 1」的东西叫概率分布——大白话说，就是**一张「接下来最可能是什么」的排行榜，榜上每个候选都标着把握有多大**。

然后它从排行榜靠前的候选里挑一个接上去，把新接的这个也算作「已有的话」，再算一轮排行榜，再挑一个……一个词一个词地滚下去，一篇文章就滚出来了。

这里藏着一个重要的细节：因为每次是从高分候选里「挑」而不是永远只取第一名，所以**同一个问题，它每次的回答可以不一样**。这也解释了 AI 身上那股「像人」的灵气——它不是

一台查表机，它每走一步都在一个带随机性的排行榜上做选择。Temperature（温度）这个参数调的就是这件事：温度低，永远选第一名，回答稳但死板；温度高，敢选排名靠后的，回答活但容易跑偏。

那台「打分机」长什么样

所以整个问题的关键，都收敛到一件事上：**那台「给词表每个词打分」的机器，内部是怎么构造的？**

它的名字叫神经网络——别怕这个词，大白话版本是：**一个由几十亿个「旋钮」组成的巨型打分器**。前面那段话的每个 token 的坐标进去，经过一层又一层的内部计算（每层本质上就是对坐标做大量的乘法和加法，旋钮的位置决定乘多少加多少），最后吐出那张排行榜。

这几十亿个旋钮，就是整本书第一章说的「机器自己拧出来的规则」。拧到位之后，语法知识、事实知识、逻辑知识，全都被压缩存储在这些旋钮的数值里——不是存成一条条能读出来的文字，而是存成一种「混在一起、直接影响打分结果」的形态。这就是为什么没人能直接打开模型指着某个旋钮说「这就是它知道曹雪芹的那一位」：知识是弥散在整个网络里的。

但这个打分器有个难关要过。看这句话：

> 「小狗追着球跑进了花园，它浑身都湿透了，因为外面正在下雨。」

读到「它」的时候，机器得知道「它」指的是小狗而不是花园；读到「湿透了」的时候，得联系到几十个词之前的「下雨」。早期的方法逐字往前读、读到后面忘了前面，就像一个人读书从不回头翻页。解决这个问题的那把钥匙，叫「注意力」——它是整个现代 AI 的引爆点。下一章专门讲它。

第五章 · 注意力：不再逐字读书

先回忆一下老办法的毛病

上一章末尾，我们给「打分机器」出了一道难题：

> 「小狗追着球跑进了花园，它浑身都湿透了，因为外面正在下雨。」

机器读到「它」的时候，必须想起几十个词之前的「小狗」。在 2017 年之前，主流的做法是让机器**从左到右逐字读**，读一个词，就把这个词的意思揉进一个「脑子里的小结」里，带着这个小结继续往下读。这就像一个记忆力有限的人读书：**不许回头翻页，只能边读边在心里做笔记**。短句子还行，文章一长，开头的内容在笔记里越揉越糊，读到结尾早就忘了开头——「它」是谁，再也对不上号。

这个毛病困扰了研究者很多年。直到 2017 年，谷歌的一篇论文提出了一个完全不同的读法，论文标题就敢叫《Attention Is All You Need》（你需要的只有注意力）。这个读法叫 Transformer，就是 ChatGPT 里那个「T」。它一出来，老办法几乎一夜之间全部被取代。它做对了什么？

注意力：每个词都抬头环顾全场

新读法的核心思想，一句话就能说完：**别逐字往前拱了，让每个词都抬起头，把整句话的所有词同时扫一遍，自己决定该重点看谁。**

拿「它」举例。处理「它」这个词的时候，机器会拿「它」去跟句子里的每个词都算一个「相关度」：「它」对「小狗」——相关度很高（小动物会湿透，合理）；「它」对「花园」——有点相关但不多；「它」对「因为」——几乎无关。算完之后，「它」的新含义就不再是干巴巴的「某个东西」，而是**「主要吸收了『小狗』的信息、附带一点『花园』的信息」的一个混合体**。代词指代谁，就在这一轮「环顾全场」里解决了。

这个「每个词环顾全场、按相关度吸收别的词的信息」的机制，就叫注意力（attention）。大白话说：**每个词都在问同一个问题——「为了搞懂我自己在这句话里的意思，我最该看哪几个词？」**

> 大白话
>
> 大白话：老办法是听汇报——信息一级一级传上来，传到后面全变了样。注意力是开全员大会——每个词直接跟所有词面对面，谁跟我相关我就多听谁的，没有中间商赚差价。

「应该看谁」是怎么算出来的

再往下钻一层，还是那老三样：乘法和加法。

每个词都带着自己的「住址」（第三章说的那串坐标）。注意力机制会用这些坐标算出词与词之间的相关度——直觉上，就是两个词的坐标「对得上」的程度越高，相关度越高。然后，每个词把所有词的坐标按相关度加权揉在一起：**谁跟我相关，我就多揉一点谁的信息进来。**揉完得到的新坐标，就是这个词「在当前语境下」的意思。

这解释了「一词多义」是怎么被处理的。「苹果」这个词，裸住址只有一个，但在「我咬了一口苹果」里，它环顾全场后吸收了「咬」的信息，新坐标就往「水果」的方向偏；在「苹果发布了新手机」里，它吸收了「发布」「手机」的信息，新坐标就往「公司」的方向偏。**词义不再是查表查出来的死东西，而是现场算出来的。**

而且机器不止雇一个「环顾员」。它同时派好多路注意力并行工作：一路专攻「谁指代谁」，一路专攻「谁修饰谁」，一路专攻「动作是谁发出的」……每一路叫一个「头」。几十个「头」各看各的重点，最后把各自的发现汇总。这就是为什么句子再绕，它也能把语法、指代、语义的关系同时抓住。

这个设计为什么引爆了一切

注意力机制的胜利，不只是「解决了指代问题」这么简单，它一次性扫清了三个老障碍：

- **长距离不再是问题。**第一个词和最后一万个词，在注意力看来距离一样——都是直接面对面。开头埋的伏笔，结尾照样能接上。

- **可以大规模并行。**老办法必须一个词读完才能读下一个，像单线程排队；注意力是所有词同时互相打量，成千上万个计算可以铺在显卡上一起跑。这意味着**可以用史无前例的规模去训练**——第七章讲的「大力出奇迹」，前提就是这个。
- **架构出人意料地通用。**同一套「环顾全场」的结构，不做大改动就能处理翻译、写作、代码、甚至后来连图片和蛋白质结构都用它。2017 年之后，AI 几乎所有方向的重磅进展，底层都站着这个架构。

至此，那台「打分机器」的构造就讲完了：token 变成坐标，坐标经过几十层「环顾全场＋加权揉合」，最后吐出下一个词的排行榜。**结构是人设计的，但里面几十亿个旋钮的具体数值，还是空的。**这些旋钮怎么从一堆随机数变成满腹经纶？那是训练的事——下一章，我们去看这本史上最贵的「错题本」。

第六章 · 一本错题本的故事

一台空机器和整座图书馆

前五章我们搭好了机器的结构：文字切成 token，token 变成坐标，坐标经过几十层「环顾全场」，最后吐出下一个词的排行榜。但这台机器刚出厂时是个**白痴**——里面几十亿个旋钮全是随机数，你问它「今天下雨出门记得带」，它可能告诉你下一个词是「冰箱」。

从白痴到博学，中间发生的事叫训练。这一章我们讲清楚训练到底是怎么干的。它的核心，朴素到像小学生的学习方法：**做题、对答案、改错题，循环几十亿次。**

一本史上最贵的错题本

训练用的「题库」，就是人类写过的海量文本：网页、书籍、论文、对话、代码——往少了说也是几万亿个 token。训练时，机器做的是这样一种填空题：

> 「床前明月光，疑是地上霜。举头望明月，低头思___。」

机器给出它的排行榜：比如「故」20%，「家」15%，「乡」5%……然后**对答案**：正确答案是「乡」。机器只给了 5%，错了。

接下来是关键动作，也是整个训练的灵魂：**算清楚每个旋钮该往哪个方向拧、拧多少，才能让「乡」的分数下次高一点。**然后把所有旋钮都按这个方向微调一丁点。这个动作的学名叫反向传播，别怕，大白话版本是：**从最后的错题出发，倒着往前追责，算出每个旋钮对这次错误各负多少责任，然后各改各的。**就像一锅汤咸了，有经验的厨师能判断出是盐放多了还是酱油放多了、各自多了多少——反向传播干的就是这件事，只不过锅里有几十亿种调料。

「下次改多少」的尺度叫学习率——大白话：**每次改错题的力度**。力度太小，学得太慢；力度太大，忽左忽右学不稳。这也是个要仔细调的东西，但本质上就这么点事。

然后下一道题、再下一道题……每道题错一点、改一点。一套流程走下来，整件事可以用三行话说完：

1. 出题：从海量文本里随手截取一段，盖住最后一个词。
2. 对答案：看机器给的排行榜，跟真实答案差多少（这个「差多少」叫损失，就是「错得有多离谱」的分数）。
3. 改错题：反向传播追责，把所有旋钮往减少损失的方向微调一点。

循环几十亿、上百亿次。这就叫预训练——大白话：**在正式上岗之前，先把人类全部文本当题库刷一遍**。

旋钮们是怎么从噪声变成知识的

见证奇迹的地方在这里。一开始，旋钮全是乱的，机器答题全靠蒙。但随着错题本越记越厚，一些旋钮开始「分工」：

- 刷过几千万道题之后，有些旋钮组合渐渐管起了**语法**——「了」后面大概不接名词，英文的 a 后面得接单数名词。
- 再往后刷，有些组合管起了**事实**——「《红楼梦》的作者是」后面，「曹雪芹」的分数悄悄爬到了第一。
- 刷得再久，有些组合竟然管起了**推理的套路**——「因为……所以……」「先……再……」这类逻辑链条的走向，也被拧进了旋钮里。

没有任何人规定哪个旋钮管语法、哪个管事实。分工是在一次次「改错题」里自己涌现出来的。这就像一支没人指挥的球队，打着打着，有人成了前锋、有人成了后卫——因为这样的配置赢球多。

> 大白话
>
> 大白话：预训练就是逼着一个学生把人类所有的书当填空题做。做第一万道时他只会背句子，做第一亿道时他摸到了语法，做到第一万亿道时，他已经把世界的规律压缩进了自己的神经里。没有人教过他任何一条规则，全是他从错题里悟的。

这台错题机器有多烧钱

说一个数量级，感受一下这件事的规模。训练一个当今第一梯队的大模型，要动用上万张顶级显卡，日夜不停地跑上几个月，光电费和硬件折旧就是几千万到上亿美元。训练全程，机

器大约要「读」完几万亿个 token——相当于把人类互联网上有质量的文字读好几遍。

这也解释了一个行业的基本格局：**预训练是少数大玩家才玩得起的游戏**。底子（基座模型）靠重金砸出来，后续的调教（下一章讲）反而便宜得多。于是行业里出现了「大厂炼丹、众人调香」的分工。

一个你可能没想到的问题：它为什么不止会背书

读到这里，一个自然的疑问是：刷了这么多题，机器会不会只是把题库背下来了？

答案是：确实会背下来一部分（这也是版权争议的源头之一），但绝不止于此。证据是，它能从容应对**题库里绝对没有的新句子**——你现编一个荒诞的场景：「如果章鱼的墨水能治感冒，药店该怎么设计货架？」它能给出像模像样的分析。背诵做不到这个，只有真正把规律拧进了旋钮、能现场组合运用，才接得住这种全新的题。

机器从「刷题」里长出来的，本质上是一台**世界规律的压缩机**：几万亿 token 的文本被压进几百亿个旋钮，压掉的是冗余，留下的是规律。这就是「智能」在这座桥上的样子。

但故事还没完。研究者在烧钱烧到一定量级时，发现了一个谁都不曾预料的现象：模型大到某个程度之后，会**突然**会一些它小时候完全不会的东西——不是慢慢变好，是「开窍」式的跳变。这就是下一章要讲的：大力为什么会出奇迹。

第七章 · 大力出奇迹

一个出乎意料的实验发现

时间回到 2020 年前后。研究者们本来在干一件很朴素的工程事：模型再做大一点、数据再多喂一点、训练再久一点，看看效果能好多少。大家的预期是「好一点、再好一点」，线性增长，一分投入一分收获。

结果出来的曲线让所有人坐直了。一些能力——比如「给一个没见过的任务，只给两三个例子就照着做」——在小模型身上基本为零，模型一路变大，这项能力一路趴在零附近，看上去毫无希望；可一旦规模跨过某个坎，这项能力**突然之间就冒出来了**，像烧开的水，99 度时还是液体，100 度就沸腾了。这种现象被叫做涌现——大白话说：**量变积累到某个点，质变突然发生，新能力凭空登场。**

这不是一次两次的巧合。翻译、多步算术、逻辑推理、读懂没见过的笑话……一批能力都呈现这种「趴窝很久，突然开窍」的曲线。

为什么会「突然」？一个说得通的解释

先泼一盆冷水：科学界对「涌现」的成因至今没有标准答案，它仍是活跃的研究课题，甚至有人争论其中一部分「突然」只是测量工具不够细的错觉。但有一个解释框架，接受度比较广，而且很好懂：

很多能力不是一个旋钮管的事，而是一整套「零件」凑齐了才工作。

拿「多步推理」举例。要做好它，模型内部至少得先学会：读懂题意、记住中间结果、按步骤推进、检查前后矛盾。这几样「零件」在训练中各自缓慢进步，但只要缺一样，整体表现就是零——就像一台收音机，电阻、电容、天线各自慢慢备齐，但在最后一件焊上之前，它一声不响；焊上的那一瞬间，它突然就响了。从外面看是「突变」，从里面看是「零件终于凑齐」。

大白话

大白话：涌现像组装，不像发酵。发酵是今天酸一点明天酸一点；组装是最后一颗螺丝拧上之前，机器一直都是一堆废铁。

这也解释了为什么「突然」总发生在大模型身上：模型越大、数据越多，能装下的「零件」就越多、越精细，凑齐一整套的概率就越大。小模型不是不学，是零件箱太小，永远差着几颗螺丝。

规模定律：AI 行业的「摩尔定律」

比涌现更基础、也更让投资人疯狂的，是另一条规律：研究者们发现，模型的表现（用一个叫「损失」的分数衡量，上一章讲过，就是「错得有多离谱」）跟三个东西呈现出非常稳定的数学关系——**模型旋钮的数量、训练数据的量、投入的算力**。三者按比例一起涨，表现就沿着一条平滑的曲线可预测地变好，而且这条曲线横跨了好几个数量级都没有失效的迹象。这就是规模定律（Scaling Laws）——大白话：**「砸钱就能变强」不是玄学，是一条画得出曲线的经验规律。**

这条定律改变了整个行业的玩法。在它被发现之前，AI 进步靠灵感——谁的算法巧谁赢；在它被发现之后，AI 进步在相当程度上变成了工程学加资本——既然曲线可预测，那就建厂、买卡、堆数据，把曲线往前走。过去几年 AI 军备竞赛的疯狂投入，底层逻辑就是这条曲线给的信心。

当然，天下没有白吃的午餐。这条路线也在撞上它自己的天花板：高质量文本快被「读」完了，电和芯片越来越贵，「再砸十倍钱」换来的进步在放缓。行业里 2024 年之后的一个明显转向——从「把模型练得更大」转向「让模型在回答时想得更多」（后训练、推理时计算）——正是对这条曲线增速放缓的回应。这属于前沿动态，但骨架没变：**整个行业的节奏，仍然是被「投入—产出」的可预测曲线牵着走的。**

这一章之后，我们该害怕还是该兴奋

「大力出奇迹」给人两种相反的情绪。兴奋的人说：曲线还在往前走，奇迹还会继续。警惕的人说：一个连建造者都只能事后观察、无法事先预言它「什么时候会突然会什么」的东西，本身就意味着风险——你不能对你造的东西的能力边界两眼一抹黑。

两种情绪都有道理。但作为读者，你现在至少拥有了一样大多数人没有的东西：你知道「奇迹」不是魔法，它是一台组装到临界点的收音机——突然响起的那一刻令人惊讶，但每一个零件的来路都清清楚楚。

不过，此时的模型还不是你手机里那个有问必答的助手。它是一台「世界的压缩机」，一个博学的「续写机器」——你给它「如何制作蛋糕」，它可能续写成一篇教程，也可能续写成「下回分解」，因为它只负责把话接得像话，不负责「帮你」。从「会说话」到「会听话」，还差最后一次调教。下一章讲这个过程——它花的钱不到预训练的零头，却决定了你拿到的 AI 是什么性格。

第八章 · 从会说话到会听话

一个尴尬的产品事故

上一章结尾说了：刚刷完题的基座模型，是个博学的「续写机器」，但还不是助手。这个差别有多要命，看一个真实发生过的场景就懂了。

早期有人拿基座模型做产品，用户问：「怎么自制蛋糕？」模型回答：「怎么自制饼干？怎么自制面包？怎么自制披萨？」——它把问题续写成了一串更多问题。用户气笑了。但从模型的角度，它没犯错：在论坛帖子里，一个问题后面跟着一串相关问题，是再常见不过的文本模式。它忠实地完成了「把话接得像话」这个它唯一会做的事。

问题出在目标错位：**我们想要的不是「像话的续写」，而是「有用的回答」。**这两件事在文本世界里经常重合，但并不相同。把基座模型调教成助手的过程，就是把目标从「像话」掰到「有用」的过程。这就是对齐（alignment）——大白话：**让模型想要的，和我们想要的，对上。**

第一步：上示范课

第一道工序很直接，叫指令微调——大白话：**给模型上示范课。**

人类老师（加上 AI 辅助）精心写出成千上万对「问题 → 好回答」的范本：「怎么自制蛋糕？」配上 step by step 的靠谱教程；「帮我总结这段话」配上地道的摘要；「写首关于秋天的诗」配上一首像样的诗。然后用第六章那套一模一样的方法（做题、对答案、拧旋钮）继续训练模型，只不过这次的「题库」全部是这些示范。

刷完这些示范，模型就「悟」到了一个新模式：**哦，原来这种「有人问、我来答」的场合，正确答案长这样。**注意，机制上它干的还是「猜下一个词」，一点没变；变的是它见过的数据——现在它猜测的方向，从「论坛里通常怎么接」变成了「一个好助手会怎么接」。

大白话

大白话：预训练让模型读完了人类所有的书，成了饱学之士；指令微调是带它观摩了几万次「优秀客服怎么接待客人」。学问没变，但它学会了「上班的样子」。

示范课便宜、见效快，但有个天花板：人能写出的示范毕竟有限，而且「什么样算好回答」里那些更微妙的东西——语气舒服不舒服、答非所问没有、有没有在不懂装懂——很难一条条写成范本。

第二步：打分行

于是有了第二道工序，思路换了个方向：**与其手把手示范，不如让模型自由发挥，人来当裁判。**

做法分两层。第一层：让模型对同一个问题生成好几个不同回答，人类裁判只管排序——「这个比那个好」。这种「二选一」的判断，比亲手写出完美答案容易得多，可以海量地做。攒下几十万条排序之后，训练一个小的「裁判模型」去学人类的口味：什么样的回答人会打高分。这个裁判模型就叫奖励模型——大白话：**一个吃透了人类喜好的自动打分器。**

第二层：让语言模型对着奖励模型「刷分」。生成一个回答，裁判打分，分高就拧旋钮强化这个方向，分低就削弱，循环往复。这套「按打分调整行为」的练法叫强化学习——大白话：**做对给糖、做错没糖，久而久之就学会挑能拿糖的做法。**这两层合起来，就是大名鼎鼎的 RLHF（基于人类反馈的强化学习）。

大白话

大白话：指令微调是临帖——照样子写；RLHF 是投稿——写完给编辑看，编辑说哪个好就往哪个方向多写。一个教「形」，一个教「神」。

性格是调出来的

经过这两道工序，模型的「性格」就定型了：你熟悉的那个礼貌、周全、爱分点、会说「这是一个很好的问题」的助手，性格底色就是在这一步被塑造的。回答多长、多谨慎、多爱用列表，都是打分环节里人类裁判口味的沉淀。

这里也藏着一些你可能已经察觉到的副作用。裁判普遍喜欢「完整、详细」的回答，模型就渐渐变得啰嗦；裁判容易给「顺着用户说」的回答打高分，模型就染上了「讨好」的毛病——你说地球是平的，它都可能先夸你「这个角度很有趣」。这些不是模型有性格缺陷，而是**打分器的口味，原封不动地长进了模型里**。奖励模型是人类偏好的镜子，镜子有瑕疵，照出来的行为就有瑕疵。

近两年，这第二道工序还长出了一个重要的新分支：不只教「答得让人满意」，还直接教「把题做对」——尤其在数学和代码这种「答案有对错」的领域，直接用「答案对不对」当糖来训练模型多想几步。这条线催生了 2024 年以来那批「会思考」的模型。它和「说话得体」是两种糖，喂出了模型的两种本事。

一座桥搭完了

到这里，从「一台只会算数的机器」到「你手机里的 AI 助手」，桥全部搭完：文字变数字（二章）、数字带意思（三章）、全部工作收敛为猜词（四章）、注意力抓住上下文（五章）、刷题刷出知识（六章）、规模带来涌现（七章）、调教塑造性格（本章）。

桥搭完了，但最重要的问题反而浮出了水面：这样一个造出来的东西，它给出的回答里，哪些是可靠的、哪些是天生的坑？它真的在「思考」吗？下一章，我们掀开盖子看它的局限——了解它的边界，比惊叹它的能力重要得多。

第九章 · 它真的会思考吗

先回答那个大问题：它在思考吗

走到第九章，我们已经拆完了整台机器。现在可以正面回答那个所有人都关心的问题了：AI 到底会不会思考？

诚实的回答分两层。第一层，**机制层面，它没有「想」这个动作**。它不做计划、不设目标、不自我怀疑；它就是一层层算坐标、一个个猜词，每一步都是几十亿次乘加。你让它解数学题时它「思考」的样子，和你说「晚安」时它回「晚安好梦」的样子，在机制上是同一种操作，没有任何一个瞬间它「停下来认真想了想」。

但第二层，**表现层面，它又确实能完成我们一向认为需要思考才能做的事**：规划、推理、纠错、类比、在陌生领域里举一反三。为什么会这样？因为「猜下一个词」这场考试逼它把人类思考的痕迹全学了去——人类写出来的文本里，藏着人类解决问题的过程；把几十亿人的解题过程压缩进旋钮之后，机器手里就有了一个「思考套路」的仓库。它不是在思考，但它调用思考套路的样子，和思考很难区分。

> 大白话
>
> 大白话：钢琴自动演奏机没有音乐细胞，但它肚子里那卷打了孔的纸，是从真正的钢琴家手底下录来的。AI 就是那台演奏机，人类全部文本就是那卷纸。琴声是真的，「懂音乐」存疑——但当琴声好到能开音乐会时，「它懂不懂」在某种程度上变成了哲学问题，而「它能干什么」是工程问题。

对你日常使用而言，记住一句务实的话就够了：**把它当作一个读完了人类所有文本、但没有亲历过任何一件事的博学者。**它的强和弱，都能从这句话里推出来。

它最大的坑：一本正经地胡说八道

如果你只用 AI 做过一件事，请记住这一节。AI 有一个与生俱来的缺陷，叫幻觉——大白话：**它会用百分之百自信的语气，编造百分之百错误的内容，而且编得滴水不漏。**

为什么是「与生俱来」？回到它的本质就懂了。它的全部训练目标是「把话接得像话」，而不是「说真话」。在它的世界里，「《XX 研究》表明……」这个句式之所以高分，是因为真实文本里这句话后面通常跟着靠谱内容——它学的是「这句话**应该**长什么样」，而不是「这句话**是不是**真的」。让它推荐参考文献，它能给你编出格式完美、期刊像样、但从没存在过的论文——因为「格式完美的参考文献」正是它学了无数遍的文本模式。

几个特别容易踩坑的地方：

- **冷门、具体的事实**：人名、日期、书号、小地方的法规——训练文本里出现次数少，它更容易「记不清就现编一个像的」。
- **最新的信息**：模型的知识停在训练数据截止的那一天，之后发生的事它一概不知，但你问它，它未必会说「我不知道」。
- **精确计算**：它不是计算器，是「预测下一个数字字符」的机器，长串乘除经常算错。

所以一条铁律：**AI 说的重要事实，尤其是要拿去用的，必须自己核实。**把它当一个记性极好但偶尔会脸不红心不跳编故事的超级实习生——它的初稿价值千金，它的定稿需要你把关。

它的另几个边界

除了幻觉，还有几条边界值得心里有数：

它没有「亲身经历」。它读过所有关于骑自行车的书，但从未摔过跤。这意味着它特别擅长「人们写下过的东西」，而对「写下来很少、全靠身体经验的东西」——比如此刻你家猫的脾气——一无所知。

它记不住你，除非系统帮它记。每次新对话，它都是一张白纸——模型本身不在聊天中「学习」，你和它聊一下午，关掉窗口它就全忘了。你感觉它「记得你」，那是外面的系统把历史记录每次重新塞给了它。

它的「记性」有容量。一次能装进脑子的文本长度（上下文窗口）是有限的，聊得太长，开头的话会被挤出窗外——表现为「聊着聊着忘了你前面说过什么」。

它有立场残留的偏见。它学的是人类文本，人类文本里的偏见——地域的、性别的、职业的——会或多或少渗进它的回答。它在重大问题上被调教得谨慎中立，但文本深处打捞起来的

偏见不会完全消失。

那为什么还这么有用?

列了这么多局限，不是劝退，而是校准。一个工具的局限越清楚，它的能力越可用。这台机器的正确打开方式，是搞清楚它在哪些事上几乎不会失手：

- **语言活儿**：总结、改写、翻译、润色、起名字、换语气——这是它的主场，幻觉风险极低。
- **结构化的脑力杂务**：列提纲、做比较、拆任务、写初稿——把「从零到一」最痛苦的那一步外包给它。
- **解释与学习**：让它把任何概念讲给你听、反复追问、换着角度问——它是有史以来最有耐心的老师，且「讲清楚一件事」恰恰是它最不容易编错的场景。
- **代码与文书**：格式严、套路多的文本，它写得又快又好；但涉及具体事实、数字、引用的部分，照铁律，核实。

一句话：**创意和初稿交给它，事实和定稿留给你。**理解它的工作原理，最终是为了知道什么时候可以放手信它、什么时候必须上手查它。

最后一章，我们把视角再拉高一点：这台机器正在被装上「手脚」，从「会说」变成「会做」——以及普通人在这个时代最该养成什么习惯。

第十章 · 学会和 AI 相处

给这台「说话机器」装上手脚

到这里，你已经知道 AI 助手的全部家底：它博学、会说、偶尔胡说、没有记忆、没见过世界。一个自然的问题是：这样的东西，怎么就变成了新闻里那个「能帮你订机票、写程序、做研究」的 AI？

答案出人意料地简单：**装上手脚。**

回想一下，模型唯一的本事是「输出文字」。但如果它输出的某段文字，不是给人看的，而是**给一个程序看的指令**呢？比如它输出「调用搜索引擎，关键词：明天北京天气」，外面的程序收到这串字，真的去搜了，把结果再喂回给它，它接着回答你。这一下，局面全变了——

模型不会查实时天气？让它调用搜索。模型算数会错？让它调用计算器。模型记不住事？给它接一个数据库，每次对话前把相关资料先塞给它。模型的每一个短板，都可以用一个外部工具补上。这个思路叫工具调用——大白话：**模型负责动脑子决定「该干什么」，工具负责动手把它干成。**它自己够不着的，就指挥够得着的。

> 大白话
>
> 大白话：一个绝顶聪明但卧床不起的人，配上电话、电脑和一群跑腿，照样能开公司。模型就是那个大脑，工具调用就是它的电话和跑腿。

再往前一步：让模型不只调一次工具，而是**循环**——「想清楚要干什么 → 调用工具 → 看结果 → 再想下一步 → 再调用」，直到把一件事办完。这种「能自己一步步把任务做下去」的 AI，就是现在常说的Agent（智能体）——大白话：**不只回答问题，还能领任务、跑腿、交差的 AI。**2025 年前后 AI 应用的主线，基本就是这条路：大脑还是那台猜词机器，但接上手脚、配上循环之后，它从「聊天对象」变成了「能干活的同事」。你现在正在读的这本书，就是这样一个 Agent 写出来的。

普通人最该养成的四个习惯

原理讲完了，落到地上。这本书的读者不需要会训练模型，但值得把下面四个习惯刻进日常：

一、把 AI 当实习生，不当算命先生。它精力无限、博闻强记、随叫随到、从不抱怨，但会犯错、会讨好、会不懂装懂。好用法是带实习生那套：交代清楚背景和目标，验收它的产出，重要的活给它分小步走，别一句话扔过去指望它读心。

二、给它上下文，它才给得出好答案。模型每一轮能看到的只有你塞给它的东西。「帮我改改这段话」和「这段话是发给客户的第一封邮件，客户是谨慎的德国采购，帮我改得更简洁专业」——同一个模型，产出天差地别。**你给的背景，就是它眼里全部的世界。**

三、事实要核实，创意可放手。第九章的铁律再强调一次：凡是要拿去用的具体事实、数字、引用，必须查证；凡是语言、结构、创意类的活儿，放心交给它。分不清哪类的时候，默认查证。

四、把它当「对话对象」，不当「搜索引擎」。它最大的价值在一来一回的追问里：「再简单点」「举个反例」「如果预算砍一半呢」「用我奶奶能懂的话再讲一遍」。一次提问得到一个平庸答案就放弃的人，和追问五轮拿到惊艳产出的人，用的其实是同一台机器。区别在用法，不在机器。

回到最初的问题

这本书从一个无聊的事实出发：计算机只会搬数字。十章走下来，我们看到了这条链——

文字切成 token，token 落进一张意义的地图；「学会说话」被压缩成「猜下一个词」这场可以打分的考试；注意力让每个词环顾全场；几十亿次「做题、对答案、拧旋钮」把人类文本里的规律压进一台机器；规模大到某个点，能力突然开窍；再经示范课和打分行两轮调教，这台「世界的压缩机」学会了当助手；最后接上工具，它开始会做事。

那么，最初那个问题——**「一个只会算数的机器，怎么就学会了说话？」**——现在有答案了：它不是被谁教会了规则，而是在足够大的数据、足够巧的结构、足够狠的算力之下，**把人类写在文字里的世界，压缩进了自己身体里的几十亿个旋钮**。它会说话，是因为我们说过的话都在里面。

某种意义上，你每次和它对话，都是在和整个人类文明的文本遗产对话——经过压缩、重组、能即时回应你的那种。这台机器到底是镜子、是工具、还是别的什么，历史还没写完答案。但至少从现在开始，它对你不再是黑箱。

这就够了。去用它吧。