

给无名之物起名字

从一百个坏案例到帕累托累加：AI 时代最稀缺的元能力

竹小竹8167

费曼式写法 · 由多个 AI 代理撰写与互相审校

导读

从一百个坏案例到帕累托累加：AI 时代最稀缺的元能力

先问你一个问题：你有没有用过一个方法很多年，却始终叫不出它的名字？

这本书的起点就是这样一件事。有个方法，从吴恩达的机器学习课程里学来：挑出 100 个坏案例，按发生频率排序，从最高的开始逐个击破。它在训练模型时百试百灵，后来在完全不相干的领域里也不断露面——处理投诉、分析事故、辅导孩子作业。它如此好用，以至于用起来的时候总有一点说不出的别扭：你不知道该怎么称呼它。没有名字，就没法跟人推荐它，没法在恰当的时刻想起它，甚至没法确定不同场合遇到的它是不是同一个它。

直到有一天，在《大海捞针》这本书里，它终于有了名字：**帕累托累加**——一个单调递增的过程。名字落下的那一刻，一个模糊的熟人才真正变成了工具。

这本书要回答的贯穿全书的问题是：给东西起名字这件事，到底是一种什么能力？为什么在 AI 时代它变得稀缺？以及，一个普通人怎么训练它？

我们会一层层剥开。名字是一种压缩——它把一整包经验压成几个音节，像指针一样随取随用；但压缩是有损的，所以名字也会撒谎。命名创造的是「可操作的单位」——有了名字才能指认它、组合它、反驳它、迭代它。马丁·福勒把这套手艺做到了行业级规模，给重构动作、企业架构模式、领域特定语言——起名，等于送给整个软件行业一套通用语言。而在 AI 什么都能生成、什么都通货膨胀的时代，稀缺的不是内容，是把一团乱麻压缩成一个好名字的判断力。

读完这本书，你不会变成福勒。但你会带得走四件可以每天练的小动作——收集、聚类、起名、检验，和一副新的眼光：从此看到无名之物，你会像看到没整理的代码一样，手痒。

第一章 · 一百个坏案例

想象你训练了一个识别猫的图片分类器，准确率 90%。听起来不错？但剩下 10% 的错误藏在哪里，你一无所知。这时候大多数团队的本能反应是：换更复杂的模型、加更多数据、调参。这就像房间里灯坏了，你的对策是把整栋房子的电线全换一遍。

吴恩达在他的机器学习课程里给过一个反直觉的建议，几乎和本能相反：**先别动模型，先去看那些错的东西。**

把错误摆到桌面上

具体做法朴素得让人意外。从你的开发集（一组模型没见过、专门用来挑模型毛病、辅助改进的数据）里，把分类错误的样本挑出来，比如 100 个。然后，一个一个人眼去看，给每个错误贴一个标签：这张图其实是狗，被当成了猫；这张图光线太暗；这张图里猫只露出半张脸……

贴完标签之后做一件关键的事：**按出现次数从多到少排序**。比如你发现 100 个错误里，有 43 个其实是狗的照片，27 个是光线问题，15 个是猫被遮挡，剩下的零零碎碎。

排序一出来，下一步该干什么几乎是自己喊出来的。43% 的错误是狗惹的祸？那就专门去找一批狗的图片加进训练集。改完这一项，准确率可能就跳了两个点。然后收拾 27% 的光线问题，再跳一点。

大白话

人话版：别修全屋的电线。先把 100 个坏灯泡摆桌上数一数，哪种坏得最多就先修哪种。

为什么这招有效

它有效的原因藏在一条朴素的数学事实里：**错误不是均匀分布的，它们是扎堆的**。如果错误均匀散布在无数种原因上，每种原因只占千分之一，那你做什么都不会有明显效果，这个方法也就没用了。但真实世界里，错误几乎总是集中在少数几个原因上——少数几类数据、少数几个场景，贡献了绝大多数的坏结果。

所以「数一数、排个序、从头开始修」不是一个偷懒的近似，而是**对准了错误分布的重心开火**。修掉第一大原因，收益立竿见影；而且因为各原因是叠加的，每修掉一个，你的指标只会往上走，不会往下掉。

它不只是在讲机器学习

这个方法的名字叫错误分析（error analysis），吴恩达把它当作训练深度学习模型的标准动作来教。但你把它抽出来看，骨架其实是这样的：

1. 收集一批「坏案例」，而不是对着抽象的总体焦虑；
2. 给每个坏案例找原因，归类；
3. 按原因的出现频率排序；
4. 从频率最高的原因开始，一个一个击破。

这个骨架里没有一丁点儿数学，也没有任何只属于神经网络的东西。换句话说，它完全可以搬出机器学习，搬进任何「有一堆问题要解决、但不知道从哪儿下手」的场景。

客服投诉太多？把投诉逐条归类排序，先修被投诉最多的那一项。孩子写作业总出错？把错题归归类，多半你会发现就两三种类型在反复出现。软件线上事故不断？把事故复盘报告里的原因数一数，排序，先消灭那个出现十几次的。

方法好用到这个程度，问题就来了——它叫什么？

第二章 · 一个没有名字的方法

先做个小实验。请你现在就想一件最近让你烦心的事：工作里反复出的纰漏、家里总也理不顺的乱象，什么都行。然后试着回答：这些问题里，哪一类出现得最多？

你会发现这个回答很难给出来。不是因为你笨，而是因为那些问题此刻在你的脑子里是一团**没有边界的雾**。「团队沟通不顺畅」——不顺畅在哪？是需求没说清，是交接总丢东西，还是评审流于形式？雾状的问题没法排序，没法排序就没法下手，于是你只能笼统地「加强沟通」，然后什么都不会发生。

说不出名字，就得反复付全款

上一章那个四步方法——收集坏案例、归类、按频率排序、逐个击破——好用到让人想天天用。但它有个隐蔽的前提：你得能把每一类原因**区分开**。区别的前提是，每一类原因在你脑子里得有一个边界，而边界往往是用语言画出来的。

没有名字的类别是什么样子？就是你每次遇到它，都要从头到尾重新理解一遍。

举个程序员的例子。九十年代做图形界面的人，反复写同一种代码：一个按钮，点了之后触发某个动作。每个人都手写一遍「点击事件回调」，代码长得都差不多，但谁也不知道别人也写过——因为这套东西没有名字。直到有人把它叫做「命令模式」，事情变了：两个程序员聊架构，一个人说「这里用命令模式」，另一个人脑子里立刻浮现出完整的结构、写法和适用场景。三秒钟完成了一次过去要半小时的传达。

大白话

人话版：没有名字的东西，你每用一次都要从头解释一遍；有了名字，一句话就能把它整个搬走。

无名之物的一个真实代价

心理学里有个著名的概念，叫错失恐惧症（FOMO, fear of missing out）：总怕别人在自己不在的时候经历了什么好事，于是不停刷手机。这个词在 2004 年前后被造出来并流行开

来。

在它被命名之前，这种感觉难道不存在吗？当然存在，每个人都体验过。但因为没有名字，它是一块说不清的背景噪音：你不知道它是什么，不知道别人也有，更谈不上管理它。有了名字之后，一切都可说了——「我刚才刷了一小时手机，其实是 FOMO 犯了」，你可以观察它、谈论它、拿它开玩笑，甚至抵抗它。

一个现象从「说不清的雾」变成「可以拿在手上的东西」，往往就差一个名字。这就是第一章那个方法的通用形态：你在各个领域反复遇到同一副骨架，却因为叫不出它的名字，每次都只能从头再认一遍。

识别 vs 命名：两种完全不同的能力

这里要拆穿一个容易混淆的地方。**能认出它**和**能叫出它**，是两种能力，而且后者比前者稀缺得多。

你已经在用了：你大概早就用「归类排序、先修大头」的方式处理过问题，只是没意识到这是个可迁移的方法。识别，是默会的能力，像骑车，身体会，嘴上说不出。命名，是把默会的东西拽到明处，给它一个可以被传递的符号。

默会的能力有两个天花板。第一，它传不出去：你会用，但你教不会别人，因为你也说不清你用的到底是什么。第二，它长不大：你无法把「那个方法」和「这个方法」放在一起比较、组合、改进，因为它们在你脑子里都没有把手，无从抓握。

哲学家迈克尔·波兰尼把这类「知道但说不出」的知识叫做默会知识（tacit knowledge）。命名，正是把默会知识变成明面知识的那道工序——这道工序，就是这本书要解剖的对象。

接下来的问题是：一个名字，到底是怎么做到「把经验整个搬走」的？得往压缩的方向看。

第三章·名字是一种压缩

程序员对「压缩」这个词应该很亲切：一个 zip 文件，把几百兆的东西压成几兆，要用的时候再展开。这本书要请你换一个角度看名字——**一个好名字，就是一段经验的压缩包。**

名字是经验的指针

说「命令模式」这四个字的时候，你的声带只振动了不到一秒。但听到的人脑子里展开的是什么呢？一整套结构：把「请求」封装成对象，于是可以排队、可以撤销、可以记录日志，以及十几种具体的写法，外加「什么时候别用它」的教训。

四个音节，换来一大坨东西。这笔账怎么算都划算——而之所以能这么划算，是因为「命令模式」这四个字在程序员社群里，像一个**指针**一样，指向一份大家共享的经验。说话的人不用搬运，听的人自己展开。

大白话

人话版：名字不是标签，是超链接。点一下就下载一整包经验。

为什么非得压缩不可：脑子的带宽就这么多

这套「指针」机制不是修辞，它有硬的理由：人的工作记忆小得可怜。心理学里反复验证过，你同一时间能摆弄的东西，大约只有四个组块（chunk，工作记忆里一个可独立操控的信息单元）左右——四个，不是四十个。

这意味着什么？意味着你没法直接思考「我们团队所有的质量问题」。那是几百个具体事件，脑子装不下。你只能思考四个组块。问题的唯一出路是：把几百个事件**压缩成少数几个组块**——「主要是交接丢东西」「主要是需求没说清」。压完之后，四个组块正好装下，你的思考终于可以运转了。

所以第一章那个方法之所以有效，一半靠排序，另一半恰恰靠的是**归类**——而归类就是压缩，给每个堆起一个临时的名字。压缩不是命名的副产品，压缩就是命名的本职工作。

压缩的代价：有损

凡是压缩都有损。zip 解压出来的文件和原文件一模一样，因为那是无损压缩；而名字对经验的压缩是**有损的**——「命令模式」三个字展开不出每一行代码，它只留下骨架，扔掉了血肉。

有损不是缺陷，是设计。扔掉血肉，正是为了腾出带宽去摆弄骨架。但也正因为有损，名字天生带着风险：你手里的压缩包和别人的原文件可能对不上。一个人说「这是敏捷」，他想的可能是「两周迭代」，另一个人想的可能是「别写文档」——同一段压缩码，解压出两套完全不同的东西，后面所有的争吵都由此而来。

这就埋下了一个伏笔：名字既能搬运经验，也能偷换经验。它是全书最有力的工具，也是最常见的陷阱。第七章我们会专门审问它。

好压缩的两个指标

既然名字是压缩，好坏就有标准。软件工程师判断一个压缩算法看两条，挪到命名上惊人地合身：

- **压缩率**：一小段符号能展开出多少经验。「沉没成本」四个字能展开出「已经花掉收不回的钱不该影响后续决策」一整条行为准则，压缩率极高。「沉没成本谬误」更是连「这是错的」都编进了码里。
- **解压保真度**：不同的人解压出的东西不一致。「内卷」这个词保真度就差得惊人——有人说它是过度竞争，有人说它是没有产出的竞争，有人干脆拿它当「累了」的同义词。保真度低的名字，搬运经验时必然走样。

顺便说一句，这两个指标之间存在张力：压缩率越狠，保真度越容易掉。「内卷」原本是个精确的学术术语——没有发展的增长，被压进日常语言后就糊了。命名的手艺，很大程度上就是在两者之间找那个甜点。

现在工具备齐了。我们知道了名字是有损压缩、是共享经验的指针，知道了它的带宽理由和好坏指标。可以回到那个悬而未决的问题了：那个「归类排序、逐个击破」的方法，它到底叫什么？

第四章 · 帕累托累加

先讲一个一百多年前的老故事。1896 年前后，意大利经济学家维尔弗雷多·帕累托在研究财富分配时注意到一个奇怪的现象：他那个社会里，大约 80% 的土地掌握在 20% 的人手里。后来他在自家花园里又注意到，大约 20% 的豌豆荚产出了 80% 的豌豆。他把这条规律写下来，后来人们管它叫帕累托法则，或者更通俗的说法——二八定律：少数原因，造成多数结果。

从「二八」到「排序击破」

二八定律流传极广，广到快要变成一句正确的废话。为什么？因为大多数人只记住了「80/20」这个数字，没记住它真正的动作。它真正的动作其实是两句话：

1. 找出那 20% 的**大头**——是哪少数几件事造成了大部分结果；
2. 先打大头，因为同样的力气花在大头上，回报是小头上的好几倍。

回头看第一章那个方法：给 100 个坏案例归类、按频率排序、从最高的开始逐个击破——这就是帕累托法则的完整动作版。43% 的错误来自狗的图片，那「狗的图片」就是那 20% 的大头；先修它，性价比最高。吴恩达没有引用帕累托，但方法的内核是同一个。

大白话

人话版：二八定律不是让你背「80/20」这个数字，是让你每次都先问：大头在哪？

为什么叫「累加」？这是关键的一半

帕累托法则管住了「先打哪个」，但那个方法还有另一半性质，帕累托本人没讲，而它恰恰是这个方法的骨头。有人读《大海捞针》时把这一半点了出来，还给它起了个名字——**帕累托累加**。

「累加」说的是整个击破过程的形状：**单调递增**。你修掉第一大原因，指标涨一截；再修第二大原因，又涨一截；再修第三大，再涨一截。每一步都让系统比上一步更好，没有任何一

步是在原地打转，更没有哪一步让事情变糟。你的进展是一条只往上走的楼梯——这就叫单调递增。

别小看这个形状，它解决的是改进工作里最常见的一种绝望：**按下葫芦浮起瓢**。很多改进之所以失败，是因为原因之间互相牵连——你修 A，B 坏了；修 B，A 又坏了；折腾三个月，指标回到原点，团队心力交瘁。这种系统的改进曲线是波浪形的，甚至是往下走的。

而「归类排序、逐个击破」选出来的那一串原因，绝大多数时候是**彼此独立、效果叠加**的：修狗的图片不会加重光线问题，修光线不会把狗变回来。于是每一步的收益可以稳稳地累加在一起。单调递增不是运气，是「按独立原因排序击破」这个动作的结构保证。

一个名字带来的三样东西

现在可以体会命名的威力了。在它有名字之前，你只是「觉得这个方法眼熟」；在它有名字之后——

- **你可以认出它**：看到一堆乱糟糟的问题，脑子里会自动弹出「这是帕累托累加的场子」，不用从头推导；
- **你可以检查它**：名字把方法拆成了两半——帕累托那一半问「大头找到了吗」，累加那一半问「每一步都单调递增吗」。如果你修完一项指标反而掉了，累加那一半立刻报警：你修的原因不独立，或者归类分错了；
- **你可以组合它**：它能和别的命名过的东西拼装——「帕累托累加」加「二八」加「坏案例清单」，就是一套完整的工作流。

同一个方法，命名前是每次从零再认一遍的熟人，命名后是一件挂在墙上、伸手就取的工具。这就是第二章说的「压缩包」在实际中兑现的样子。

不过到目前为止，我们看的都是一个人给方法起名字。把命名这件事做到行业级规模的，是一个叫马丁·福勒的人——下一章看他的手法。

第五章 · 福勒的礼物

软件行业有个流传几十年的段子：计算机科学里有两大难题，缓存失效、命名，以及差一错误。三个难题，两个跟名字有关。这个段子好笑，是因为它精确——所有程序员都知道，给变量起名字是日常工作中最频繁、也最能看出功力的动作。

但有一个人把「命名」从写代码的细枝末节，做成了一项行业级的基础设施。他叫马丁·福勒。

他给「整理代码」的动作起了名字

1999 年，福勒出版了《重构》。「重构」这个词他不是凭空造的——它九十年代初就在 Smalltalk 社区的小圈子里流传——但他做了一件关键的事：把「不改变外在行为、只改善内部结构」这件模糊的事，拆成一个个有名有姓的小动作。

比如提炼函数（Extract Method）：把一段塞在大函数里的代码抽出来，做成一个独立的小函数，起个名字。这个动作本身平淡无奇，每个程序员都凭直觉做过。但福勒把它写进了目录：叫什么名字，什么情况下该做，什么情况下不该做，怎么做才安全。一整本书，几十个这样的动作，每个都有名字、有边界、有操作步骤。

这本书改变了行业的说话方式。以前你说「这段代码太乱了，我想收拾一下」，没人知道你要干什么、干到什么程度算完；现在你说「这里该提炼函数，那里该用搬移字段」，每一句都是可执行、可检验、可争论的具体动作。**模糊的不适感，被翻译成了一张动词清单。**

大白话

人话版：福勒给「收拾代码」写了本字典。从此程序员争论代码时，用的是同一套词条。

他接着给企业的架构套路起了名字

2002 年的《企业应用架构模式》做的是同一件事，只是对象更大。企业软件里那些反复出现的骨架——数据怎么存、事务怎么管、界面和逻辑怎么分层——被他一个个起了名字：仓

储、工作单元、延迟加载、事务脚本、数据映射器……几十个模式，每个都配着「什么时候用、代价是什么」。

注意这里发生的连锁反应。这些模式在福勒命名之前当然存在——无数程序员独立写出过几乎一样的代码，就像第二章说的那些手写的点击回调。命名之后，它们从「各家私藏的土办法」变成了「行业共享的公共词汇」。一个新人入职，说「我们在这里加个仓储层」，他继承的是整个行业几十年攒下的经验，而不是他自己三个月的摸索。

然后他把「语言」本身变成了命名对象

福勒做的第三件事更有意思：他开始给「语言」这种通常不被当作对象的东西起名字。2010 年他出版了《领域特定语言》，把领域特定语言（DSL，Domain-Specific Language）这个概念推到了主流视野里。

DSL 是什么？就是为一类特定任务专门造的小语言。正则表达式是处理文本的 DSL，SQL 是查询数据的 DSL，Makefile 是描述构建步骤的 DSL。它们早都存在，但「给这类东西一个统称」这个动作，让一件本来散着发生的事突然可以被谈论、被比较、被有意地去设计了。一个团队可以说「我们给计费规则造个 DSL」，而不是「我们写了个奇怪的解析器」——前者是一个决策，后者只是一次事故。

命名的行业级操作手册

把福勒的几本书放在一起看，他的手法其实是一套可复制的流程：

1. **扎根具体**：每个名字背后都是几十上百个真实案例，不是书斋里的抽象。「提炼函数」这个名字之所以有力量，是因为它精确对应你明天早上就会遇到的那段代码。
2. **给边界**：他不只说「是什么」，更说「什么时候不该用」。一个模式的使用代价写得和使用方法一样认真——这才让名字不变成口号。
3. **交给社群**：名字写进书里、放进目录，就成了公共财产。别人引用它、反驳它、修订它，词汇表自己就长起来了。

值得说清的是：福勒不是第一个给软件模式起名的人——1994 年「四人帮」那本《设计模式》已经列出过 23 个经典模式。福勒的独特之处在于规模和自觉：他把命名这件事系统性

地做了一遍又一遍，从代码动作做到企业架构再做到语言本身，等于向整个行业演示了**命名作为一种方法论**到底能做成多大。

他演示的结果你也看到了：程序员成了世界上命名密度最高的职业群体之一，而他们的协作效率，很大程度上就靠这套共享词汇撑着。问题是——名字对思考的改变，还能不能往更深处挖一层？下一章我们解剖名字的机制。

第六章 · 名字如何改变思考

第五章看完福勒的手艺，这一章往机制深处走一步：名字进了脑子之后，到底做了什么？我们拆出四个动作。每个动作你都能在自己身上验证。

动作一：有了名字，才能「指」它

先看一个日常场景。孩子摔倒哭了，你把他抱起来说：「膝盖摔破了，这叫疼，吹一吹。」就在这一句话里，你完成了一次命名的传教。下次他再摔，他不会只剩一团无法言说的恐慌——他会说「疼」。这个词让他能把感受**推到自己面前**，当成一个可以指认、可以讨论的对象。

哲学家给这个现象起了名字，很绕口：语言不只是描述思想的工具，它会反过来参与塑造思想。不必记这个学说，记它的日常版本就够了——**说不出的东西，你很难稳定地思考它**。思考需要抓手，名字就是最小的抓手。

动作二：有了名字，才能组合它

单个的词只是零件，组合的威力在句子层面兑现。对比这两句话：

- 「那个项目花了好多钱好多时间，大家舍不得砍，结果越陷越深。」
- 「这是沉没成本谬误。」

第一句是对的，但它只能一次描述一个项目。第二句是个零件，它可以被放进任何句式：「这里有没有沉没成本谬误？」「他犯了典型的沉没成本谬误。」「这个决策被沉没成本谬误绑架了。」一个名字，瞬间变成一套可复用的思考句式——你获得的不只是一个分类，而是一个**可以参与运算的算子**。

大白话

人话版：没名字的经验像散装零件，每次用都得现场造；有名字的经验像乐高积木，拿起来就拼。

动作三：有了名字，才能反驳它

这一条最反直觉，也最重要。**能被反驳，是一种奢侈品，而名字是它的入场券。**

没有名字的东西，你甚至无法对它说不。一种说不清的团队氛围让你难受，你只能「感觉不太对」——感觉没法被反驳，也没法被讨论，最后往往以「你太敏感了」收场。而一旦它有了名字——「这是模糊责任：任务没有明确的唯一负责人」——反对就有了靶子：「我不同意把问题归给模糊责任，这次明明是排期问题。」

科学史上这样的例子比比皆是。「燃素」曾被用来解释燃烧，后来被证明根本不存在——但请注意，**它是在有了名字之后才被推翻的**。一个理论只要被命了名，就把自己放到了可以被检验、被证伪的位置上。命名的过程，就是把一个模糊的直觉变成一个明确的靶子的过程。靶子才有资格被射中。

动作四：有了名字，才能迭代它

最后一步最实际。名字让经验可以像软件一样 versioning——被引用、被修订、被替换。

「食物链」这个名字最初只是描述谁吃谁，后来生态学家沿着它长出了「食物网」「营养级」「生物富集」——每一个新名字都精确地修订或扩展了前一个的边界。如果当初那团现象始终没有名字，这些修订根本无处发生，因为你没法修订一个「那个东西」。

软件行业把这个循环走得最快：模式被命名、被大量使用、被发现有坑、被修订或废弃——「上帝对象」成了反面教材，「单例」从良方变成需要慎用的东西。词汇表像活的代码库一样在演化。**命名的终极产物不是词典，是一个会自我修正的知识系统。**

四个动作合起来

指认、组合、反驳、迭代——四个动作有个共同点：它们都是**操作**。这揭示了命名的本质：命名不是给事物贴一张死标签，而是把一个东西变成**可操作的单位**。可指认，可拼装，可攻击，可进化。

反过来说，一个从没被命名的东西，对你的思考系统来说是不存在的——不是它不重要，是你的系统没有它的驱动程序。

但别急着拥抱命名。能把东西变成可操作的单位，意味着也能把它变成可操纵的道具——名字会撒谎，而且撒谎的时候面不改色。下一章，审问名字。

第七章 · 名字会撒谎

第三章说过，名字是对经验的有损压缩。这一章处理有损的另一面：**压缩会走样，而走样的名字不是无害的缩写，是会说谎的道具**。它撒谎时面不改色，因为它自己也不知道自己在撒谎。

地图不是疆域

波兰裔学者阿尔弗雷德·科日布斯基在 1931 年提出过一句后来被引用烂的话：**地图不是疆域**。地图是对疆域的压缩——有用的压缩，没有它你寸步难行。但地图上的边界在疆域里往往找不到：你在两省交界的公路上行驶，看不到任何一条线。线是画在纸上的，不在地上。

名字就是地图上的线。「内向」和「外向」之间，在真实的人身上找不到一条分界——人格是连续的谱系，名字却把它剪成了两截。记住这条：**疆域是连续的，名字是离散的**。每一个名字，都是一次把连续的世界切开的动作。切得好，世界变清晰；切得不好，切出来的碎片就开始冒充世界本身。

大白话

人话版：名字永远比它指的东西简单。问题不在简单，在于你忘了它简单过。

谎言之二：名字偷偷夹带结论

更危险的谎，是名字把一个有争议的判断，伪装成一个无可争议的事实。语言学家管这类词叫伪装成事实的评价——它的语法形式是陈述，实际内容却是观点。

看几个样品。「拖延症」：一个人推迟做一件事，原因可能有一万种——优先级不对、任务模糊、完美主义、纯粹累了。「拖延症」这个名字一刀切过去，把所有原因压成一个「症」，好像病灶找到了，接下来只需「治疗」。但你回想第四章：原因分类不对，帕累托累加根本启动不了——你连第一大原因是什么都被名字挡在门外了。

「情商低」是同类。「他情商低」听起来像诊断，实际上把「没听懂暗示」「不会拒绝人」「表达太直接」这些完全不同、需要完全不同对策的问题，糊成了一团。名字本该是压缩经

验的指针，这类名字却是**终止思考的开关**——名字一出口，讨论就结束了。

谎言之三：名字会过时，但名字不会自己让位

第三个谎言最隐蔽：名字描述的是它被造出来那一刻的世界，可世界一直在动，名字却留在原地站岗。

「人工智能」这个词是 1956 年造出来的。它描述的那个雄心——造出通用的、像人一样思考的智能——此后经历了几十年的高潮与寒冬。今天的媒体用「AI」称呼从语音识别到大语言模型的一切。这个名字既太大（暗示着通用智能）又太糊（什么都往里装），于是每天都在制造误解：有人以为 AI 要毁灭人类了，有人以为 AI 就是高级搜索。不是大众笨，是这个压缩包的保真度本来就低——第三章的两个指标，「AI」这个名在保真度上不及格。

软件行业对此有个自救的传统，值得学：持续地给旧概念改名，给新概念正名。「分时」让位给「客户端-服务器」，「数据挖掘」让位给「机器学习」——每一次改名都是一次保真度的修复。行业没有假装名字可以一劳永逸，而是把改名当成了常规维护。

给名字做安检

既然名字会撒谎，成熟的命名者就得给名字过安检。三道关卡，都来自前面几章：

1. **边界关**：这个名字把连续的世界切在了哪？切口合理吗？（「内向/外向」的切口在谱系中间，合理；「拖延症」把万种原因切成一团，不合理。）
2. **夹带关**：这个名字里藏着评价吗？把「拖延症」还原成「推迟了任务」，把「情商低」还原成具体行为——还原之后还能否成立？
3. **新鲜度关**：这个名字造出来时描述的世界，还在吗？「AI」这张 1956 年的地图，今天该补画了。

过了安检的名字才敢用，但也只是「敢用」，不是「全信」。科日布斯基的提醒要常挂在墙上：地图不是疆域，名字不是事物——**地图画得越好，越要记得它是地图。**

现在，工具、威力、风险都摆齐了。剩下的最后一个是：这套能力，在今天这个正在被 AI 改写的时代，到底价值几何？下一章算这笔账。

第八章 · 为什么 AI 时代更稀缺

现在算这本书的核心账：命名这种能力，为什么在 AI 时代反而更值钱了？直觉的答案恰恰相反——AI 什么都能写，起个名字还不是一秒钟的事？

这个直觉混淆了两件事。**生成一个名字，确实变便宜了；判断哪个名字是对的，变贵了。**我们一层层看。

内容在通胀，注意力在通缩

经济学里最朴素的一条：供给暴涨的东西，价格崩盘。AI 让「生成内容」的成本趋近于零——方案、总结、代码、名字，要多少有多少。一份会议纪要，过去要人写半小时，现在一秒出三版。内容进入了通胀时代。

但硬币的另一面：**阅读、辨别、吸收注意力的成本一分没降**，反而因为内容泛滥而更贵了。一个人一天还是只有二十四小时，能认真读完的东西还是那么多。当供给端无限膨胀、需求端纹丝不动，稀缺品就换了位置——从「生产内容」变成了「判断哪份内容值得读」。

大白话

人话版：自来水不要钱了，挑水喝的品味就值钱了。AI 时代最不缺的是内容，最缺的是把一团内容压成一行标题的那只手。

命名恰好是「压缩」这门手艺本身

这时候回头看第三章：命名是什么？是压缩——从一堆具体经验里提炼骨架、扔掉血肉、压成一小段能装进别人脑子的符号。

AI 的生成和命名的压缩，方向正好相反。AI 擅长的是**展开**：给一个词，它展开成一千字。命名要做的是**收拢**：给一千个具体案例，压成一个词。展开可以自动化——因为语言模型本质上就是个展开机器。收拢没法完全自动化，因为收拢的前提是你知道**该扔掉什么**，而「该扔什么」取决于你对问题本身的判断，这是目前还需要人来拍板的部分。

所以不是 AI 不能参与命名——它能起出一百个候选名，个个通顺。瓶颈从「造出名字」挪到了「从一百个通顺的名字里挑出那个对的」。**起名的手变便宜了，看名字的眼力变贵了。**

共享语言，变成了人机之间的接口

还有一个更实际的变化。越来越多人干活的方式变成了：人给 AI 下指令，AI 产出，人验收。这套协作能不能跑顺，取决于人能不能把需求**说清楚**——而说清楚的前提，是你自己知道那个东西的骨架是什么、边界在哪、什么算对什么算错。

换句话说：你给 AI 下的每一个模糊指令，都会原样变成模糊的结果。「写个方案」和「用帕累托累加的思路分析这 100 个投诉」是两个世界——后者之所以有效，正是因为你脑子里有那个压缩包，能把它解开来给 AI 看。命名不再只是人与人的通用语言，它正在变成**人与 AI 之间的 API**。没有自己词汇表的人，在这个接口上只能用默认参数干活。

判断力是怎么来的：还是那两个字，案例

先泼一盆冷水：判断力没法直接学。「知道该扔什么」这种能力，没有教材，因为每一个命名场景要扔的东西都不一样。它只能从一个地方长出来——**大量具体案例**。

这正是吴恩达方法的另一副面孔。你练命名的方式，和训练模型的方式同构：攒一批真实案例（哪怕很小），亲手归类，亲手起名，然后拿去用——用对了说明压缩得对，用走样了说明边界切错了，回头修。福勒那本模式目录，也是几十年项目案例喂出来的，不是想出来的。

AI 在这里的角色很微妙：它是极好的陪练。你可以把五十个用户反馈丢给它，让它试着聚类命名，然后你来挑刺——它的每个错误分类都是一次免费的「保真度」教学。但陪练不能替你上场：最后拍板哪个名字是对的、敢不敢把它用出去，这一步留给人。**压缩可以外包，压缩率由你签字。**

账算完了：生成通胀、辨别稀缺、命名是压缩的手艺、人机接口要词汇表——四件事指向同一个结论。接下来的问题只剩一个：这只手，具体怎么练？最后一章，训练手册。

第九章 · 命名肌肉训练手册

前八章把「命名是什么、为什么稀缺」讲完了。这一章只做一件事：把训练方法拆成四个可以今天就上手的动作。它们不是并列的四选一，是一条流水线——收集、聚类、起名、检验，循环转起来。

动作一：收集——攒你自己的坏案例

一切命名训练的原料都是同一个东西：**具体案例**。没有案例的命名是空转——坐在书桌前想「我该给什么起名字」，什么都想不出来，因为名字不是想出来的，是从案例里长出来的。

做法简单到不像训练：建一个清单，专门记「反复出现、让你烦、但你说不清为什么」的事。今天开会被同一个问题绕回去了，记一条；这周第三次解释同一件事，记一条。每条只写两三句话，必须带具体场景——「周三评审，需求文档改了四版还是因为没说清验收标准」，而不是「沟通有问题」。后者没法归类，前者能。

大白话

人话版：命名训练的第一课不是起名，是捡石头。捡够五十块，玉才可能出现。

动作二：聚类——让案例自己站队

攒到二三十条，开始归类。这一步的关键纪律是：**先看案例，再定类别**，而不是先画好格子往里塞。格子先行的分类是你对世界的预设立场，很容易把不同质的东西硬捏成一团——第七章「拖延症」就是这么来的。

实操上，把案例一条条读，遇到「这两条其实是同一件事」的就并成一堆，堆多了再给堆起临时名。名字这时候要丑一点、具体一点都没关系：「评审会上说不清验收标准」「群里 @ 了没人认领」「文档改了没人同步」——这种名字现在越具体越好，抽象的留给后面。

如果你的案例超过五十条，自己分不动了，这正是上一章说的陪练场景：把案例丢给 AI 让它先分一版，你不抄它的答案，专挑它的错——它把哪两条硬并了、哪个类别名字取得含糊，每个错误都暴露一个你自己没看清的边界。

动作三：起名——过一遍福勒的工序

堆分好了，给真正重要的堆起正式的名字。工序就是第五章拆出来的福勒流程，简化成三个问题：

1. **它做了什么动作？** 好名字优先描述行为，不描述情绪。「验收标准缺失」比「评审乱象」好——前者告诉你修什么，后者只告诉你不爽。
2. **它的边界在哪？** 写一句话：什么情况算它，什么情况不算。写不出这句话，说明你对这类案例的理解还没到位，回去再聚类。
3. **它什么时候不该用？** 像福勒写模式的代价一样，给自己每个名字配一条「别乱用」。这会逼着你把名字从口号降级成工具。

起完名，过一遍第七章的三道安检：边界关、夹带关、新鲜度关。名字是给别人用的压缩包，出厂前质检。

动作四：检验——用走样来校准

最后一个动作最容易被跳过，也最关键：**把名字拿去用，然后观察它在哪里走样。**

检验的方法是故意的压力测试：拿三个新案例来，不看答案先问自己「这算不算『验收标准缺失』？」——三个都判断一致，名字的保真度就合格；有一个犹豫，说明边界线上有模糊地带，回去修订那句边界说明。

更进一步，把名字交给别人用：讲给同事听，只说名字和边界，让他拿新案例去套。他套错的地方，就是这个名字会撒的谎。这和第三章的「解压保真度」是一回事，只不过这次你是在实测，而不是在估分。

循环起来：这就是你自己的帕累托累加

现在把镜头拉远，你会看到这套训练本身就在演示这本书的主题：

你的坏案例清单，就是第一章的那 100 个 bad case；归类排序，就是帕累托那一半；每一次起名—使用—修订，指标只涨不跌，就是累加那一半。训练命名能力的过程，用的恰好是被训练出来的那个方法——**用命名训练命名，用帕累托累加改进帕累托累加。**

最后一句话留给开书时那个场景：下次再有一个用了多年却叫不出名字的方法，别再让它从你的世界路过。停下来，给它起个名字。一个名字改变不了世界，但它会改变你看到的世界——从一团雾，变成一排可以点开的、清清楚楚的压缩包。

给无名之物起名字 · 竹小竹8167