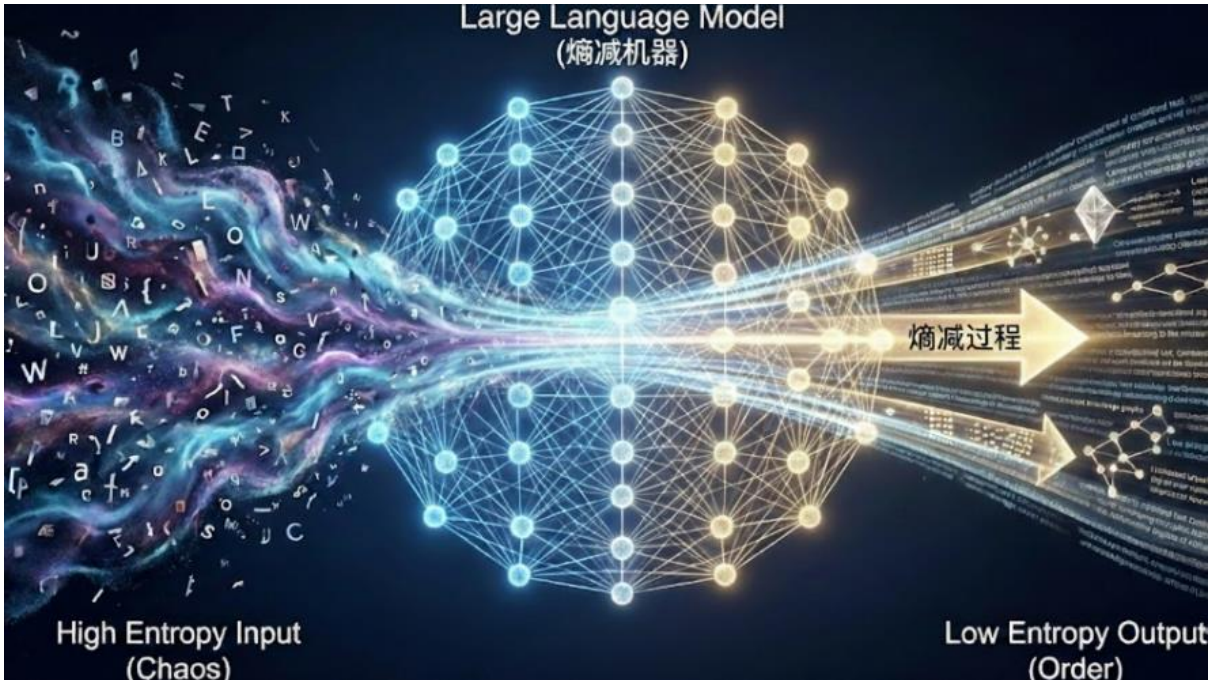


一切都是熵减

上篇：熵减的本质

2025 年 12 月 24 日

引子

几年前，我曾与南洋理工大学的一位物理学教授闲聊。他问我：″你觉得整个宇宙最基本的定律是什么？″

我想了想，说：″热力学第零定律？它定义了温度，是一切测量的基础。″

他摇摇头。

″那是第一定律？能量守恒，质能等价，宇宙的总账本永远平衡。″

他还是摇头。

″第三定律？绝对零度不可达，给宇宙设了一个下限？″

他笑了：″是第二定律。″

热力学第二定律：孤立系统自发地朝着热力学平衡方向——最大熵状态——演化。换句话说，宇宙的方向是熵增。秩序瓦解，结构消散，信息湮灭。这是时间之箭，是万物的宿命。

"但是，"我问，"如果宇宙的方向是熵增，那生命是什么？智能是什么？我们为什么能从混沌中建立秩序，从噪声中提取信息？"

他说："因为生命不是孤立系统。它通过消耗能量、向环境倾倒熵，来换取自身的局部熵减。生命是宇宙熵增洪流中的一个逆流漩涡——它不违反第二定律，而是利用第二定律。"

那一刻我突然意识到：智能的本质，就是熵减。

而语言模型，作为一种人造的智能，它的训练、它的推理、它的一切，也都是熵减。

从那之后，我逐渐意识到，熵这个概念贯穿了语言模型的方方面面——从训练到推理，从 Transformer 到 Diffusion，从长序列生成到隐空间推理。它像一条暗线，把看似不相关的现象串联起来。

这篇文章，是我对这些思考的整理。希望能与大家分享，也期待批评指正。

一、训练：学习熵减的规则

1.1 从混沌到秩序

训练开始时，模型的参数是随机初始化的。这些参数没有结构，没有意义，是纯粹的噪声。模型面对任何输入，输出的都是接近均匀分布的猜测——熵极高。

训练结束时，参数凝结成了精巧的结构。面对"中国的首都是"，模型会以极高的置信度输出"北京"——熵极低。

这个从随机到有序的过程，就是参数空间中的熵减。

1.2 数据：熵减的能量来源

热力学告诉我们：局部熵减需要消耗能量，并向环境倾倒更多的熵。语言模型的训练同样如此。

数据是"能量"。每一个训练样本都携带着关于世界的信息——这是人类文明几千年积累的低熵结构。模型通过拟合这些数据，把数据中的结构"吸收"进参数中。

计算是"做功"。GPU 燃烧电能，执行梯度下降，一步步把参数从混沌推向有序。

热量是"倾倒的熵"。数据中心的散热系统把废热排向环境。模型内部的熵减了，环境的熵增了。总账依然平衡。

这就是训练的热力学本质：消耗能量，吸收数据中的结构，在参数中实现熵减。

1.3 Loss：熵减的度量

我们用交叉熵损失（Cross-Entropy Loss）来训练语言模型。这个名字不是偶然的——它直接度量的就是模型输出分布的熵与真实分布的差距。

Loss 下降，意味着模型的预测分布变得更尖锐、更确定、更接近真实答案。这就是熵减。

训练曲线从高 Loss 收敛到低 Loss，就是熵减过程的可视化。我们盯着那条下降的曲线，其实是在观察一个系统如何从混沌走向有序。

1.4 训练学到的是什么？

表面上看，模型在学习预测下一个 token。但更深层地看，模型在学习的是：熵减的规则。

什么样的上下文应该导向什么样的确定性？什么样的模式值得被压缩和记住？什么样的结构是语言和世界的规律？

训练完成后，这些规则被编码进了参数。模型成为了一个熵减机器——给它一个高熵的输入空间，它知道如何把熵减下来。

二、推理：执行熵减

2.1 从可能性空间到确定输出

推理开始时，面对一个 prompt，可能的回复几乎是无限的。这是一个巨大的、高熵的可能性空间。

推理结束时，模型输出了一个具体的、确定的序列。从无限可能收敛到一个答案——这是熵减。

每生成一个 token，模型都在做一次局部熵减：从词表的几万种可能性中，选出一个。一步步地，高熵的可能性空间被压缩成低熵的确定序列。

2.2 条件熵的收束

更精确地说，模型在做的是降低条件熵 $H(x_{t+1}|x_{\leqslant t})$。

刚开始时，面对一个 prompt，可能性的空间是巨大的——条件熵极高。随着生成的推进，每一个写下的 token 都在对未来施加约束。

当然，熵的变化并非单调递减。在某些时刻——比如故事的转折点、引入新概念时、或者"The winner is..."这样的悬念处——不确定性会短暂飙升。但整体而言，上下文的积累就像收紧的网，不断排除不相关的可能性路径，将巨大的可能性空间逐步坍缩。

到了句末，标点落下的那一刻，这一步的不确定性终于归零。

2.3 训练与推理：两个层次的熵减

现在我们可以看清全貌：

训练是在参数空间做熵减——从随机参数到有结构的参数。这是一次性的（或者说，是缓慢迭代的），学到的是熵减的规则。

推理是在输出空间做熵减——从高熵的可能性空间到具体的序列。这是每次请求都发生的，执行的是训练好的熵减规则。

一个学习规则，一个执行规则。两个层次，同一本质：熵减。

三、表示空间决定熵减的效率

3.1 同一问题，不同空间，不同的熵

现在进入本文最关键的洞察：熵减的效率取决于你在什么空间做熵减。

同一个问题，在不同的表示空间中，呈现出的熵是不同的。

设想一个高维的语义状态 z。当我们把它投影到低维的 token 空间得到 $x = \pi(z)$ 时，会发生什么？

多个不同的语义状态 z_1、z_2、z_3 可能都对应同一个 Token 前缀——这是投影造成的混叠。如果你只看 Token 序列，"下一个词是什么"充满不确定性——熵很高。但如果你能在高维的语义空间看问题，在每条具体的轨道上，下一步其实是近乎确定的——熵很低。

熵没有凭空增加。那些"多出来"的不确定性，是投影造成的混叠假象。

3.2 Token 空间的三重诅咒

Token 空间——语言模型输出分布的那个离散词表——有三个结构性缺陷，让它不适合直接做熵减：

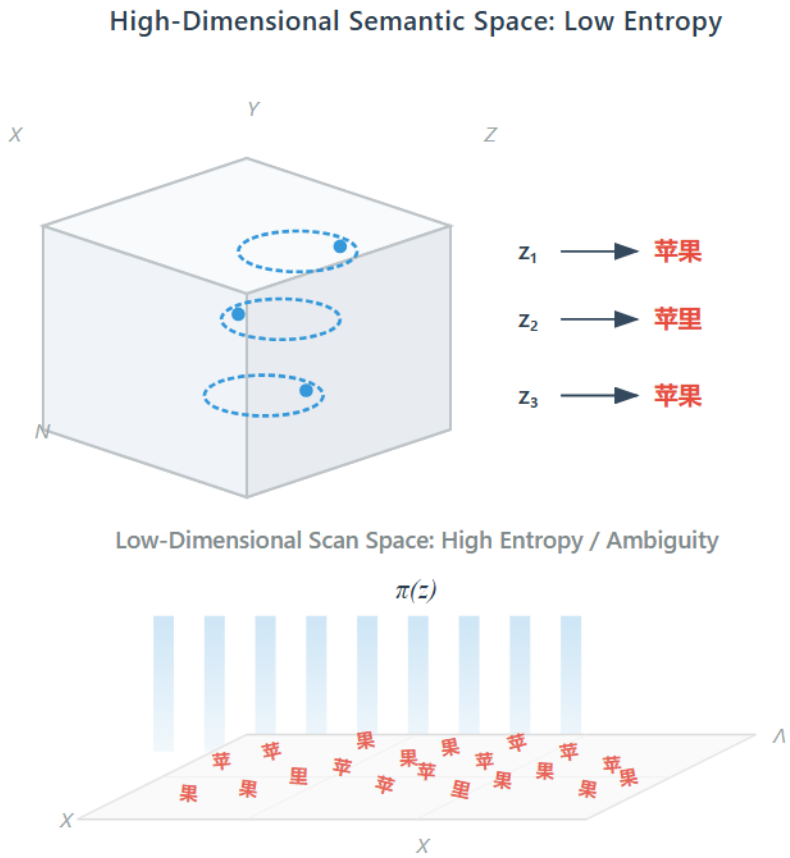
离散性。Token 是离散符号，不是连续向量。语义的微妙差异被粗暴地量化，精度丢失。

度量失配。编辑距离不等于语义距离。"cat"和"hat"只差一个字母，意思毫无关系；"huge"和"enormous"拼写完全不同，却是近义词。Token 空间的度量结构与语义结构严重错位。

双向混叠。这是最致命的缺陷，而且是双向的：

一方面是多对一：同一个意思有无数种说法。"我很高兴"、"我很开心"、"I'm happy"——不同的 Token 序列指向同一个语义，造成表示冗余，熵被虚高。

另一方面是一对多：同一个 Token 在不同语境下有不同含义。"bank"可以是银行，也可以是河岸；"苹果"可以是水果，也可以是公司。这是真正的混叠——不同的语义被压缩到了同一个符号上，必须依赖上下文才能消歧。



升维的核心作用，就是解开这种一对多的混叠：把在 Token 空间重叠的不同语义，在高维空间中拉开。同一个"bank"，在"river bank"的上下文中和"Bank of America"的上下文中，会被映射到高维空间的不同位置。

在这样的空间做熵减，就像在一张模糊的、充满重影的照片上辨认人脸——不是人脸本身难认，而是你拿到的数据把难度人为拉高了。

3.3 Transformer 的智慧：升维-演化-投影

自回归 Transformer 的成功，在于它找到了正确的熵减姿势：不在 token 空间硬做熵减，而是先换到一个更友好的空间。

第一步，升维（Lift）。Embedding 层把离散 token 映射到高维连续向量，开始解混叠——把被压在一起的不同语义在高维空间分开。

第二步，演化（Evolve）。多层 Transformer block 在高维空间计算。这里的核心是 Attention 机制。

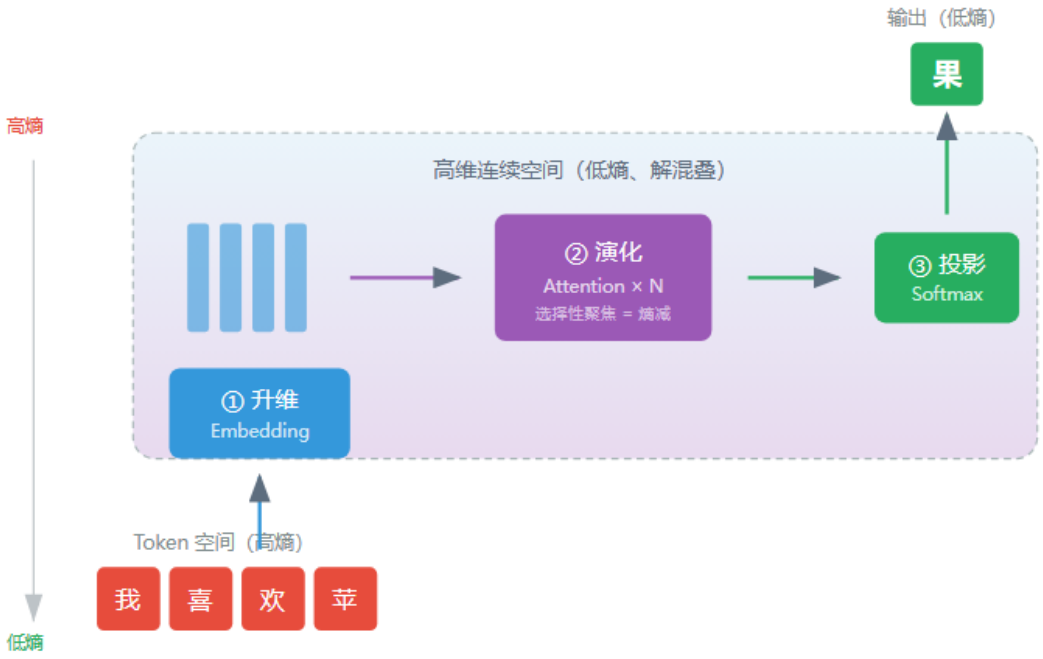
Attention 做了什么？它让每个位置可以"看到"所有其他位置，然后动态地决定：当前位置应该关注哪些上下文？这本身就是熵减——从所有可能的上下文中，选择性地聚焦到最相关的部分。

更重要的是，Attention 的计算发生在高维空间。Query、Key、Value 都是高维向量，相似度在高维中计算，信息在高维中聚合。整个过程不受 Token 空间离散性和混叠的限制。

这就是为什么 Attention 如此强大：它在一个解混叠的、连续的、高维的空间里做信息的选择和整合。熵减发生在正确的地方。

第三步，投影（Project）。最后一层把高维表示投影回 token 分布。熵减已经完成，这一步只是"翻译"成人类可读的符号。

这就是 Transformer 的秘密：它不是在 token 空间做熵减，而是先升维到一个熵本来就低的空间，在那里顺畅演化，最后再投影回来。熵减发生在正确的地方，所以高效、自然、稳定。



四、小结

让我们回顾这一篇的核心要点：

1. **一切都是熵减。**训练是在参数空间做熵减，从随机到有结构；推理是在输出空间做熵减，从可能性到确定性。

2. **表示空间决定熵减的效率。**同一问题在不同空间呈现不同的熵。高维、连续、解混叠的空间中熵低，熵减自然；低维、离散、混叠的空间中熵被人为抬高，熵减困难。

3. **Transformer 选对了空间。**升维-演化-投影的三阶段结构，确保熵减发生在正确的地方。

在下一篇中，我们将看到：如果选错了空间会怎样？Diffusion LM 的探索，从热力学的视角看，或许走了一条更为艰难的路。

（上篇完，中篇待续）