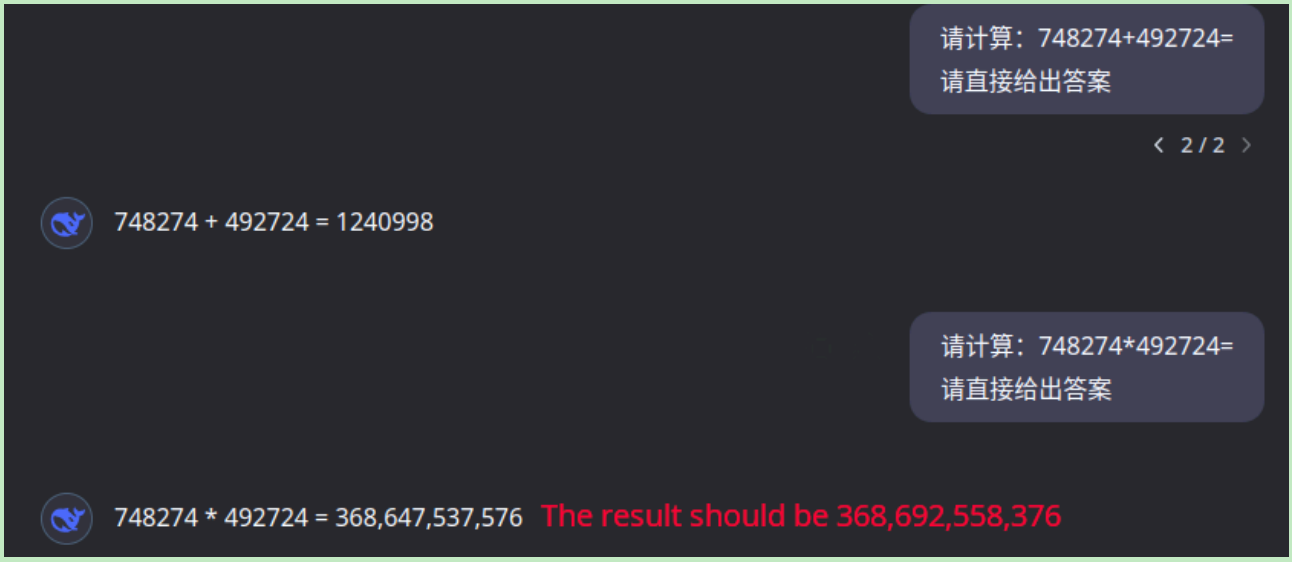
LLM 在加法和乘法任务中的表现差异与能量观点下的解释

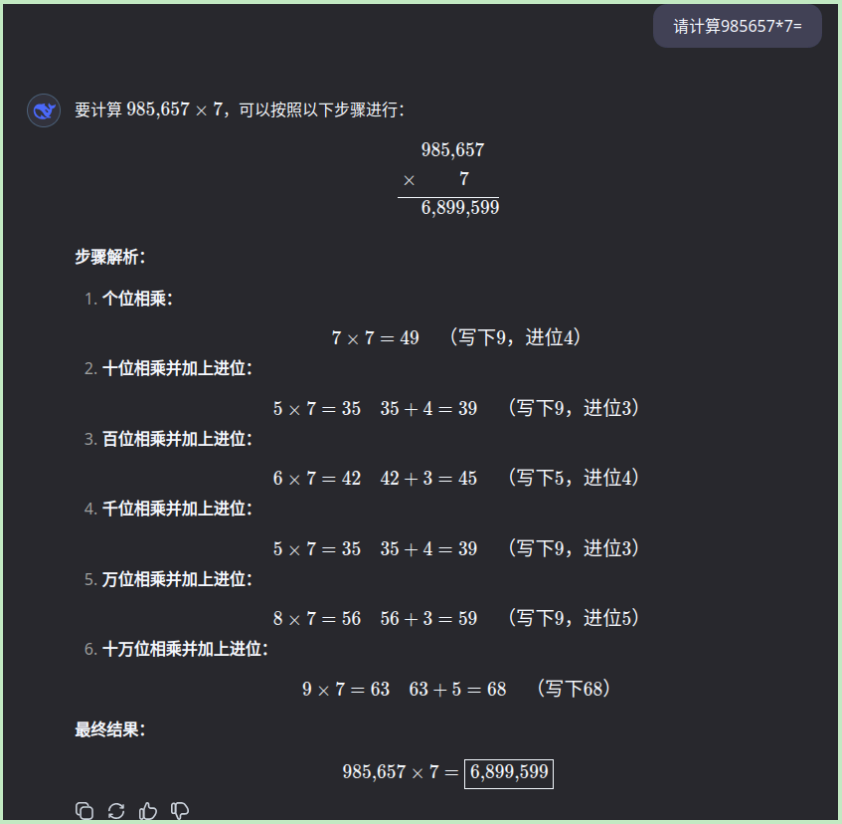
1. 发现的问题

我们发现，对于 LLM 来说，处理两个整数的加法和乘法问题，会有较大的表现差异。例如：

- 对于 6 位数+6 位数的加法，DeepSeek V3， 通义千问，GPT-4o 等都可以很容易的答对；然而对于同样规模的乘法,这些模型却很难给出正确答案。



- 同样对于乘法任务，6 位数乘以一位数的问题，DeepSeek V3， 通义千问，GPT-4o 等模型可以很容易的答对。
- 从 DeepSeek V3 给出答案的形式来看，DS V3 会倾向于将乘法转化为加法进行求解，这对于部分乘法任务来说，相比 GPT-4o，通义千问等倾向于直接给出乘法结果的模型，会有更好的表现。



2. 尝试作出的解释

2.1 背景条件

我们将 LLM 的范围，限制在使用 Attention 机制捕获上下文关联，并通过深度网络对自然语言的转移概率 $p(x_k \mid \boldsymbol{X}_{t<k})$ 进行建模的自回归模型。

2.2 行为解释

这里我们将会从“自回归模型”的视角，结合数学原理，阐述加法和乘法在模型视角下的区别和模型表现差异的可能原因。

我们所见的 LLM 的“计算”行为，实际上是在进行上述的自回归，即根据已经提供的“上文”，求解一个词表概率分布，并尝试选择最高概率的 token（贪婪搜索）或者保留 Top K 进行多步推理后选择累计转移概率最高的 path（集束搜索）。也就是说，模型并非是在计算一个数值，而是在根据已有信息，进行一个 token 的选择。

我们不妨从数学原理上来观察理论上加法和乘法任务中结果某一位数字的选择域：

- 对于加法：

  我们假设有一个 K 位数字 N 可以被表示为：$N = \sum_{i=1}^{K}(n_i \times 10^i)$，以及另一个 S 位数字 M 可以被表示为：$M = \sum_{i=1}^{S}(m_i \times 10^i)$。为了简单起见，我们不妨假设 $K \geq S$，即 $\min(K,S) = S$，由于交换律的存在，这是可以接受的。

  于是有:

  对于 $N+M$ 的结果 R，其从低位向高位的第 j 位置上的数字 r_j 的可能取值为：

$$r_j = \begin{cases} (n_j + m_j) \bmod 10 & \text{if} \quad j \leq \min(K,S) \text{ and } c_{j-1} = 0 \\ (n_j + m_j + 1) \bmod 10 & \text{if} \quad j \leq \min(K,S) \text{ and } c_{j-1} = 1 \\ n_j \bmod 10 & \text{if} \quad \min(K,S)+1 > j > \min(K,S) \text{ and } c_{j-1} = 0 \\ (n_j + 1) \bmod 10 & \text{if} \quad \min(K,S)+1 > j > \min(K,S) \text{ and } c_{j-1} = 1 \\ n_j & \text{if} \quad j \geq \max(K,S)+1 \end{cases}$$

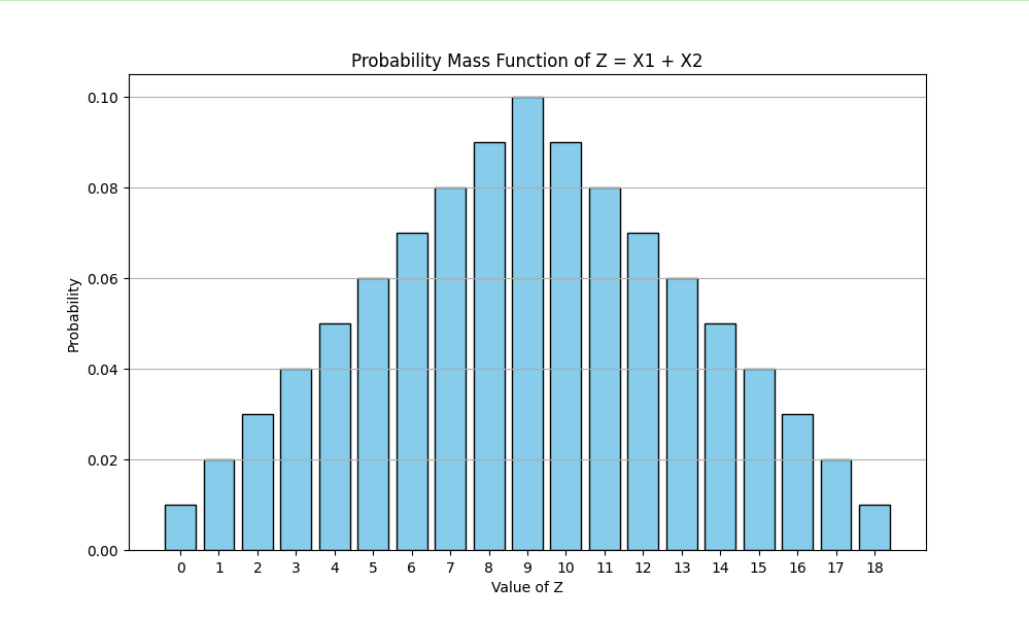
  此处 c_j 为进位标志，$c_0 = 0$，$c_j = \lfloor (n_j + m_j) \div 10 \rfloor$

  不难发现，对于加法来说，每一位上的可选值如果以正常的计算方法是可以被唯一确定的，而对于 LLM 来说，它只能从高位向低位输出，无法确定进位情况。但即便如此，每一位上的可选择范围也只有至多 2 个数字，即 $(n_i + m_i) \bmod 10$ 或者 $(n_i + m_i + 1) \bmod 10$。两个可选值在语义上似乎非常相近（有待验证），且主要由 n_i和 m_i 决定，（这里我们假设一个数字一定是一个单独的 token，而不存在多个数字组成一个 token 的情况）。

  而至于进位，理论上会由两个加数从 $i-1$ 到 0 所有的数字决定，然而实际情况是，$i-1$位置上的数字几乎已经给出了明确的指示，我们令进位存在的概率 $P(c_i = 1)$ 关于 n_i, m_i 的函数如下：

$$P(c_i = 1) = F(n_i, m_i) = \begin{cases} 1 & \text{if} \quad n_i + m_i \geq 10 \\ P(c_{i-1}) & \text{if} \quad n_i + m_i = 9 \\ 0 & \text{if} \quad n_i + m_i \leq 8 \end{cases}$$

显然，c_i 的具体概率分布取决于两个均匀分布随机变量n_i, m_i的和，我们可以从下图中观察到这个和的概率质量函数:



只有0.1的概率会让这个和落在不确定的区间，剩下0.9的概率下，这个和会让 c_i 表现出明显的二值性。

言下之意，即使模型只关注了 $n_i, m_i, n_{i-1}, m_{i-1}$ 四个值，在假设模型能够完全学习到规则的情况下，也已经有非常大的概率能够正确的处理少位数的加法问题，如果是 100 以下的加法，模型甚至能够完全正确的处理。

总结来说，在加法任务中，不仅只有一阶的隐藏信息，即进位，并且无论是进位还是结果主值的选择，都具有一个非常小的选择范围，这使得模型在这个任务上表现的非常好。

- 对于乘法

我们沿用上面的对于数字的假设，目标依然是研究结果 R 中每一位可能的取值。

1. 部分积计算： 对于每一位 i 和 j，满足 $i + j = k$，计算部分积 $n_i \times m_j$。 这些部分积的总和为：

$$S_k = \sum_{i=1}^{k} (n_i \times m_{k-i}) \tag{1}$$

其中我们认为， $k \in [1, K + S] \wedge \mathbb{Z}$

2. 进位处理： 来自第 $k-1$ 位的进位 c_{k-1} 需要加到 S_k 中。 因此，第 k 位的中间和为：

$$T_k = S_k + c_{k-1} \tag{2}$$

3. 结果位计算： 第 k 位的结果 r_k 是 T_k 对 10 取模：

$$r_k = T_k \bmod 10 \tag{3}$$

4. 进位更新： 新的进位 c_k 是 T_k 除以 10 的整数部分：

$$c_k = \left\lfloor \frac{T_k}{10} \right\rfloor \tag{4}$$

综合以上步骤，我们可以得到第 k 位数字 r_k 的通用计算公式：

$$r_k = \left(\sum_{i=1}^{k} (n_i \times m_{k-i}) + c_{k-1} \right) \bmod 10 \tag{5}$$

其中，初始进位 $c_0 = 0$，且进位 c_k 的更新公式为：

$$c_k = \left\lfloor \frac{\sum_{i=1}^{k} (n_i \times m_{k-i}) + c_{k-1}}{10} \right\rfloor \tag{6}$$

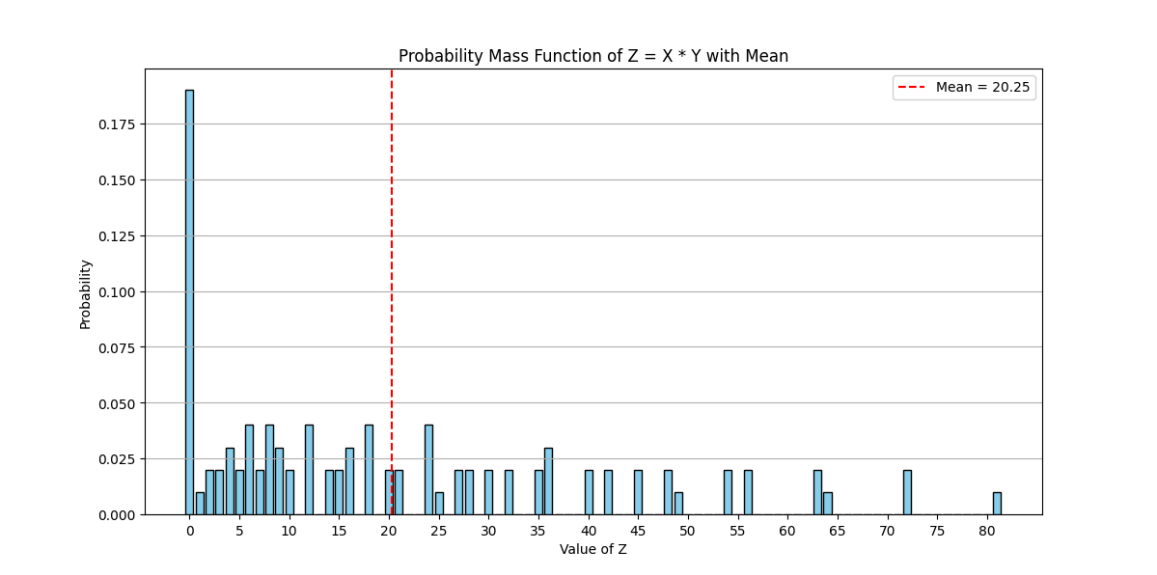
相比起加法任务中简单的线性进位机制，乘法运算呈现出随位置增长的进位选择空间

单个位置的部分积的总和 S_k 的取值范围是 $[0, 81k] \wedge \mathbb{Z}$。那么中间和 T_k 的取值范围就是 $[c_{k-1}, 81k + c_{k-1}] \wedge \mathbb{Z}$。

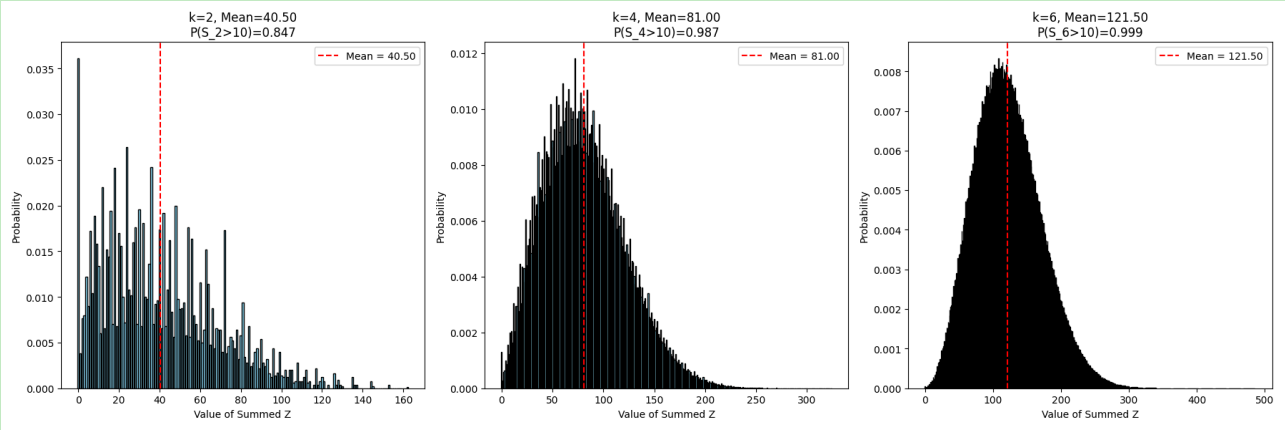
于是，对于进位 $c_k = \lfloor T_k \div 10 \rfloor$，其可选范围是 $\left[\left\lfloor \frac{c_{k-1}}{10} \right\rfloor, \left\lfloor \frac{81k + c_{k-1}}{10} \right\rfloor \right] \wedge \mathbb{Z}$

我们可以尝试对 S_k 的可能取值进行统计学的分析以得到更多结论：

假设 X, Y 为独立的，服从$[0, 9]$均匀分布的两个整数随机变量，下图呈现了 $Z = X \times Y$ 的概率质量函数:



于是 S_k 实际上是 k 个 Z 的和，下图展示了当 $k = 2, 4, 6$ 的时候，S_k 的概率质量函数:



从上文的 (1), (2), (4) 式子可以发现，进位 c_k 大约是 S_k 的 $\frac{1}{10}$, 对于 c_{k-1} 来说这个量级也成立。那么不难发现，粗略估计下，$k=2$ 时就有 84.7% 的概率会发生进，并且进位的选择空间相当大，而更加复杂的是，本次进位还会影响下一次进位的选择空间切的确具有明显的效果，例如当 $k=6$ 时，c_6 几乎会让 c_7 的选择空间整体向右移动 1 个单位。这种进位之间的链式影响，使得乘法的隐藏信息是对上下文极度敏感，且状态空间很大的。

但需要考虑到的是，由于 k 的取值会超过任意一个因数的长度，此时我们认为超出长度的位置上因数值为0，那么实际上在 k 超过 $(K+S)\div 2$，即最终结果的一半位置的时候，由于高位补位 0 的存在，选择空间将会缩小。此外，由于最低位没有前置进位，其选择空间是 $[0,9]\wedge\mathbb{Z}$且非质数占优。

言下之意，我们可以认为乘法结果中每一位的选择空间非常大，但是靠近结果中间位置的选择空间是最大的，而靠近结果两端的选择空间是最小的。

由此我们终于可以对 LLM 在加法和乘法任务中的表现差异作出一个初步的解释：

Hypothesis-1:

在 LLM 的自回归模型中，加法任务中每一位数字的选择空间非常小，而乘法任务中每一位数字的选择空间非常大，这使得模型在加法任务中表现的非常好，而在乘法任务中表现的较差。

Hypothesis-2:

由于乘法任务中靠近结果中间位置的选择空间是最大的，而靠近结果两端的选择空间是最小的，我们可以期待看到的是模型在乘法任务中，两端的数字选择更加准确，而中间的数字选择更加困难。

3. 对“选择空间”的量化定义

3.1 熵

为了量化“选择空间”的大小，我们引入熵的概念。熵是信息论中的一个重要概念，用来衡量一个随机变量的不确定性。对于一个离散随机变量 X，其熵定义为：

$$H(X)=-\sum_{x\in X}p(x)\log p(x) \tag{7}$$

其中 $p(x)$ 是 X 取值为 x 的概率。

对于一个随机变量 X，其熵越大，表示其不确定性越大，选择空间越大。而当 X 的熵为 0 时，表示 X 的取值是确定的，选择空间为 1。

对应到 LLM 中，我们可以在单向注意力的条件下，定义某个位置上的“位置熵”，用于衡量在给定的前文下，该位置后选词的不确定性。我们可以定义第 i 位置的位置熵为：

$$H_i = -\sum_{x \in V} p(x \mid \boldsymbol{X}_{t<i}) \log p(x \mid \boldsymbol{X}_{t<i}) \tag{8}$$

其中 V 是词表，$p(x \mid \boldsymbol{X}_{t<i})$ 是在给定前 $i-1$ 个位置的词的情况下，第 i 位置的词为 x 的概率，即转移概率，也是 LLM 输出的概率分布。

3.2 能量

实际上我们还可以定义在一个特定的词表概率分布下，每个后选词的能量。对于一个给定的位置 i 和词表 V，我们定义第 i 位置上词 x 的能量为：

$$E_{i(x)} = -\log p(x \mid \boldsymbol{X}_{t<i}) \tag{9}$$

能量的高低直接反映了模型对某个词的选择倾向：

- 低能量：表示模型认为该词在当前上下文中出现的概率较高，是一个“自然”或“合理”的选择。
- 高能量：表示模型认为该词在当前上下文中出现的概率较低，是一个“不自然”或“不合理”的选择。

通过能量定义，我们可以将模型的概率输出转化为一个能量景观（energy landscape），其中低能量区域对应模型偏好的选择，而高能量区域则对应模型不太可能的选择。

3.3 能量和熵的关系，以及模型的自回归在热力学上的解释

通过能量和熵的定义，位置熵实际上是能量的加权平均，权重为每个词的概率。

于是，我们可以将模型的自回归过程理解为一个热力学系统的演化过程。这种关系为我们提供了一种新的视角来理解模型的生成过程：模型在生成每个词时，实际上是在能量景观中寻找一条低能量的路径，而熵则反映了这条路径的确定性。

4. 尝试用熵和能量的观点来解释加法和乘法任务中的表现差异

对于加法任务而言，生成每一位的数字时，其选择空间较小，熵较低。在这种情况下，模型可以以较少的“能量”就将该位置的能量降低到一个最低的水平，即选出正确的结果数字。

而对于乘法任务，由于每一位的选择空间较大，熵也相应增大，这意味着模型的生成过程需要通过更多的推理和计算来达成结果。模型在乘法任务中的生成过程也需要更多的信息和计算步骤。而实际上由于生成模型原理的限制，输出一个乘法结果并不会比输出一个加法结果产生更多的计算，因此，模型在乘法任务中表现较差，尤其是在更高位的计算中，选择空间的复杂性和熵的增大使得生成过程不如加法任务那样顺利。

5. 热力学观点下对大语言模型领域中其他现象的解释

5.1 提示词工程（Prompt Engineering）与起点熵减

提示词工程通过精心设计的提示语（prompt）来约束模型的推理空间，实际上是在减少推理的起始信息熵。通过提供明确的结构、背景信息或语境，提示词将模型的选择范围限定在一个较小、较清晰的空间内，从而有效降低模型的选择不确定性。

- 熵减的机制：当模型输入的提示词更具约束性时，生成的文本路径不再是完全自由的，而是在提示词的引导下更加集中。这种结构化的提示使得模型在生成时能够更快速、更准确地缩小推理空间，减少不必要的冗余和复杂性，从而降低熵。
- 效果提升：通过在生成前就降低了起始熵，模型能够在较少计算资源的情况下，生成更加高效、精准的推理文本，尽管消耗的算力几乎没有变化。就像热力学中的“外力推动”一样，提示词工程通过对系统的初始状态进行引导，促使系统向低熵状态演化。

5.2 深度推理模型与熵减

深度推理模型并不像传统的提示词工程那样通过外部约束来直接减少起始熵，而是通过训练过程让模型逐步学会如何在高熵的环境下，生成低熵的输出。

- 和提示词工程不同，R1 这样的深度推理模型一般不需要提示词技巧，他们通过在训练中形成的推理倾向，输出大量的推理文本。一方面这些推理文本会约束后方生成 token 的不确定程度，另一方面这个过程的确消耗了更多的算力（能量）。在这二重作用下，深度推理模型在起始熵较高的情况下也能够获得理想的低熵输出。

5.3. 强化学习与推理文本的高熵特性

大段的推理文本虽然能够有效的约束后方 token 的选择空间，但是即便如此，推理过程通常包含多种潜在的解释和推理路径，因此具某种程度上可以认为具有较高的信息熵。在强化学习（Reinforcement Learning, RL）中，模型通过与环境的交互来优化其推理策略。通过奖励机制，强化学习能够指导模型逐步减少不必要的推理路径，最终朝着一个低熵、更加确定的输出方向发展。

- 推理文本的高熵与奖励机制：推理文本的高熵特性意味着生成过程中的决策空间较大。通过强化学习，模型不仅能够选择最优的推理路径，还能减少低效的选择和冗余信息的产生。在训练过程中，模型通过不断调整其行为策略，逐步收敛到更加符合语言规则和推理逻辑的路径，从而减少生成文本中的不确定性。
- 转移概率与真实语言的接近性：强化学习的优化目标使得模型学习到的转移概率分布更接近真实语言中的自然概率分布。通过训练，模型逐渐学会如何在推理过程中消除多余的选择，优化每个推理步骤的决策，从而使得生成的文本更加高效、精确，熵逐步降低。

5.4. 总结

从热力学角度来看，无论是提示词工程还是深度推理模型，本质上都在通过不同的方式减少系统的不确定性（熵）。提示词工程通过外部结构化的引导直接减少起始熵，使得模型在生成过程中能够更高效地收敛；而深度推理模型则通过大量的推理文本生成消耗更多能量的同时约束后方 token 的选择空间，从而在高熵的起点下保证生成更加高质量的文本。此外，强化学习的推理任务通过优化决策路径，使得生成的推理文本趋向于低熵状态，从而提升推理质量。