熵驱动的记忆重构认知模型

Hongyou Wang

2026

Independent Researcher

摘要

现有以 Transformer 为核心的大规模语言模型，普遍隐含一个未经检验的前提：只要在给定输入条件下生成概率最高的文本序列，推理过程便可视为完成。在这一范式中，系统被默认要求在任何时刻、任何问题下都进入表达阶段，而“是否已经具备回答条件”并未作为一个显式、可计算的对象加以建模。大量实践表明，这一假设正是语义漂移、自相矛盾、表面合理但不可验证结论等问题的根源之一，而这些问题无法仅通过扩大模型规模或提升生成精度得到根本缓解。

与此形成对照的是，人类在面对复杂、不确定或信息不足的问题时，通常具备一种前语言层面的判断能力：意识到“当前尚未想清楚”，从而选择继续思考、补充结构或暂缓表达。这种能力并不等价于答案正确性的评估，也不依赖显式逻辑符号推演，而是一种对自身认知状态是否已收敛至可表达区域的判断。本文的核心目标，正是将这一能力形式化，使其成为认知系统中的一个显式机制。

本文提出一种以**可表达性熵**为核心判据、以**记忆重构（REWRIT）**为动力的认知推理模型。该模型不以“生成什么答案”为推理目标，而是将推理重新定义为：在给定查询约束下，通过结构加工推动系统内部状态进入一个风险可控、结构稳定、可被可靠表达的区域。语言生成仅被视为这一过程完成后的表达阶段，而非驱动推理本身的核心机制。

在模型中，系统状态由查询（query）与有限候选记忆集合共同构成，答案文本被明确排除在系统状态之外。为判断当前状态是否具备进入表达阶段的资格，本文引入可表达性熵这一结构性风险度量。该熵并非热力学熵或香农信息熵，而是一种面向认知与工程的状态不确定性刻画，综合反映三类风险来源：候选记忆对查询语义的覆盖不足、记忆之间的内部冲突，以及在历史路径中反复失败所体现的结构不稳定性。系统仅在熵值低于给定阈值 ε时，才被允许进入语言生成阶段。

当系统状态尚未进入可表达域时，模型触发 System 2 结构推理过程。System 2 并不生成答案文本，其唯一职责是通过 REWRITE 操作引入新的记忆结构，使整体状态在熵意义上严格下降。REWRITE 不要求逻辑完备或语义完整，其评价标准仅在于是否降

低了可表达性熵。为保证推理路径的稳定性与可追溯性，模型施加单调扩展约束：系统禁止删除或回溯既有记忆，只允许通过新增或重构结构来推进认知状态。

在长期运行中，有效的 REWRITE 模式可通过虚拟环境训练被抽象和沉淀为可复用的经验结构，从而逐步前移为快速检索阶段（System 1）即可调用的认知模板。为防止记忆无序膨胀，本文进一步引入记忆熵守恒约束，要求系统在持续学习与写回的过程中，总体记忆资源保持有界，通过抽象与压缩而非简单累积实现能力提升。这一约束保证了模型在长期运行下的容量稳定性、推理效率与结构自洽性。

从整体上看，本文模型在结构上严格区分检索、推理与表达三个阶段，并通过熵判别器实现阶段切换，使“是否该回答”成为一个显式、可调节、可学习的对象。该框架并不依赖特定的语言模型、检索机制或环境形式，因而可作为一种更底层的认知约束层，与现有生成模型、RAG 系统或执行模块协同使用。

本文不主张该模型立即替代现有大规模语言模型，而是提出一种不同的认知出发点：推理的本质不在于生成答案，而在于让系统状态变得可以被稳定表达。在这一视角下，语言生成只是认知过程的最后一步，而非承担认知判断与风险控制的核心角色。该框架为构建更稳健、可控、具备长期演化能力的智能系统提供了一种新的结构路径。

1 引言：为什么“能不能回答”比“怎么回答”更重要

现有大规模语言模型通常隐含一个默认假设：只要生成概率最高的文本序列，即可视为完成回答。在这一范式下，推理被等价为条件生成，系统被迫在任何输入条件下给出某种形式的输出。

然而，大量实践经验表明，模型失败的根源往往并非语言表达能力不足，而是在尚未形成稳定认知结构的情况下被迫进入表达阶段。其典型表现包括：语义漂移、自相矛盾、循环论证，或生成表面合理但无法验证、不可复用的结论。这类失败并不能通过提升语言模型规模或生成精度得到根本解决。

相比之下，人类在面对复杂或不确定问题时，通常具备一种前语言层面的判断能力：意识到“当前尚未想清楚”，从而选择继续思考、补充信息或暂缓表达。这种能力并不依赖于明确的逻辑符号推演，也不等价于答案正确性的评估，而是一种对自身认知状态是否成熟的判断。

本文尝试将这种自我状态判定能力加以形式化，使其成为一个可计算、可验证且可调节的机制，并将其置于推理流程的核心位置。

本文关注的核心问题是：

在不预设显式逻辑符号系统、也不依赖答案监督的前提下，系统如何判断当前是否已经具备进入表达阶段的条件？

围绕这一问题，本文采取以下基本立场：

推理不等同于文本生成。推理的本质在于推动系统内部状态逐步收敛到一个结构稳定、风险可控的区域，而非直接产出语言序列。

语言生成属于表达阶段而非推理阶段。它应当发生在系统状态已经通过“可表达性”判定之后，而不是承担判定本身的职责。

基于上述立场，本文提出一种以可表达性判定为核心的认知模型框架，在该框架中，“是否该回答”成为一个显式建模的问题，而非隐藏在生成概率分布中的隐含结果。

在引入本文模型之前，有必要首先澄清一个广泛用于描述人类认知过程的经典区分：System 1 与 System 2。

System 1 与 System 2：从认知隐喻到可执行结构

“System 1 / System 2”这一划分最早来源于认知心理学，用以描述人类在面对问题时两类截然不同的处理模式。该概念在 Daniel Kahneman 等人的工作中被系统化提出，用于区分快速、直觉、低成本的判断过程（System 1）与缓慢、审慎、高认知负载的反

思过程（System 2）。在这一语境中，System 1 与 System 2 并非具体算法或模块，而是一种对人类认知现象的功能性描述。

需要强调的是，原始的 System 1 / System 2 理论本质上是一种心理学隐喻：它并不试图给出可执行的计算流程，也未回答一个关键工程问题——系统应当如何判断何时从“快速反应”切换到“深度思考”。在多数实际应用中，这一划分往往被直接移植为经验性假设，例如“复杂问题使用慢思考，简单问题使用快思考”，但缺乏明确、可验证的切换判据。

在人工智能与大规模语言模型语境中，System 1 / System 2 的概念常被非正式地借用，用以指代“直接生成”与“多步推理”之间的差异。然而，这种用法通常停留在行为层面的类比，并未解决以下更基础的问题：

当系统面对一个具体 query 时，它如何判断当前状态是否已经成熟到可以进入表达阶段，或是否必须继续进行结构加工？

本文沿用 System 1 与 System 2 这一命名，但对其含义进行严格的工程化重定义，并刻意剥离其心理学语境中的主观描述成分。在本文模型中：

System 1（S1）：快速、无迭代、无状态的初始化检索机制。其职责仅限于在给定 query 条件下，从长期记忆中召回一组可能相关的记忆原子，用以构成初始系统状态。S1 不进行结构评估，不尝试解决冲突，也不承担“是否可以回答”的判断责任。

System 2（S2）：以结构成熟度为目标的显式推理过程。当系统状态未通过可表达性判定时，S2 被触发，其唯一目标不是生成答案，而是通过记忆重构（REWRITE）等结构性操作，降低系统的不确定性，使状态逐步收敛到可被稳定表达的区域。

在这一重定义下，S1 与 S2 的区别不体现在“速度快慢”或“计算量大小”上，而体现在其在整体推理流程中的功能角色：

S1 回答的是：“有哪些可能相关的结构？”

S2 回答的是：“在这些结构之上，是否已经具备表达条件？如果没有，还缺什么？”

更重要的是，本文不预设 S1 与 S2 的切换规则。系统是否从 S1 进入 S2，或从 S2 进入表达阶段，完全由后文引入的可表达性熵判定机制决定，而非由人工设定的任务复杂度、步数阈值或启发式规则控制。

通过这一方式，System 1 / System 2 不再只是认知层面的描述性概念，而被转化为一个由统一判据调度的、可计算的结构化推理框架。这为后续章节中对系统状态、熵判别器以及 S2 结构推理流程的形式化定义奠定了基础。

2 基本对象：为什么答案不应作为系统状态的一部分

2.1 Query（问题 / 任务）

记 query 空间为

$$\mathcal{Q}.$$

任意一次推理由一个 query 触发：

$$q \in \mathcal{Q}.$$

在本文中，q 被视为对系统状态的约束条件，而不是必须立刻映射为答案的输入。其作用在于限定系统需要收敛到的表达目标与语义边界，而非直接决定输出文本或输出形式。

需要强调的是：

- q不等价于“问题文本”；
- q可以是结构化任务、抽象条件、多模态信号或复合指令；
- 系统对 q的处理重点在于：围绕它构造一个可表达的内部结构状态。

2.2 qv 记忆（记忆原子）

记系统的长期记忆全集为

$$\mathcal{M}.$$

系统中的每一条记忆记为

$$m_i \in \mathcal{M},$$

其形式定义为一个五元组：

$$m_i = (q_i, v_i, z_i, c_i, s_i).$$

各字段含义如下。

- **q_i：记忆来源 query / 触发语境**

q_i用于标记该记忆在何类 query 语境下被激活或复用。它可以是原始 query 的压缩签名、聚类标识或结构化条件。

■ **v_i：记忆内容（结构性认知单元）**

v_i表示一条内部认知结构，用于承载断言、经验片段、中间结构、抽象关系或约束信息。本文刻意不将 v_i定义为自然语言文本，也不要求其满足任何显式逻辑语法。

在本文模型中：

- v_i是一种与自然语言可对应但不等价的内部表示；
- 自然语言仅是其在表达阶段的一种可能投影形式，而非其存在前提；
- 推理过程中对 v_i的操作不依赖语言生成或语言相似度。

因此，v_i 的语义并不通过语言结构直接体现，而是通过其在系统中的组合方式、几何关系以及对整体可表达性熵的影响间接体现。

这一设计的核心动机在于，将推理所依赖的认知结构与最终用于对外沟通的语言表达在语义上解耦，避免系统在尚未完成结构收敛时，被语言形式提前牵引进入表达阶段。

■ **z_i：向量表示（Embedding）**

$$z_i \in \mathbb{R}^d$$

是记忆 m_i的向量表示，用于支持：相似度计算、记忆检索（S1 阶段）、聚类与近邻搜索。

需要注意的是，z_i：

- 不承载真值或逻辑语义；
- 不直接参与推理决策；
- 仅作为几何意义上的可比表示空间。

■ **c_i：记忆资源占用（Memory Cost / Entropy Budget）**

$$c_i \in (0, +\infty) \text{ 或 } [0, +\infty)$$

表示该记忆在系统中占用的资源成本、熵预算或容量成本。

需要强调的是，c_i：

- 不表示该记忆的正确性、置信度或概率；
- 不参与推理方向选择；
- 仅用于长期记忆容量控制与记忆熵守恒约束。

记号统一说明：本文后续不再使用 ω_i表示资源占用，统一使用 c_i。

■ **s_i：记忆方向权重（Polarity / Directional Value）**

$$s_i \in [-1,1]$$

表示该记忆对推理或表达的方向性价值（偏好 / 极性）。它用于表达"支持 / 避免"这类方向信息，而不与 c_i的容量角色冲突。

解释语义如下：

- $s_i \approx 0$：几乎无意义 / 不确定 / 暂不使用；
- $s_i > 0$：倾向使用（支持性记忆）；
- $s_i < 0$：倾向避免（反向记忆 / 禁忌 / 反例 / 陷阱提示）；
- $|s_i|$：方向强度。

采用 qv 记忆原子的设计动机

采用上述 qv 记忆原子的主要原因在于：

- 本模型不预设显式逻辑符号系统（如 与 / 或 / 非）；
- 需要一种统一载体，能够同时承载事实、假设、经验与中间结构；
- 推理的核心目标不是直接生成文本，而是一个
  "结构组合 → 熵评估 → 结构重构（REWRITE）" 的过程。

在这一设计下，记忆不再是"静态事实集合"，而是参与系统状态演化、对推理方向产生影响的动态结构单元。

2.3 系统状态

系统在任意时刻的状态定义为：

$$S = (q, M)$$

其中：

$$q \in \mathcal{Q}, M \subseteq \mathcal{M}, |\ M\ | < \infty.$$

需要特别强调的是：**系统状态中不包含答案文本**。

原因在于：

- 一旦将答案纳入状态，系统会过早围绕某一答案形态进行优化；
- 这将掩盖一个更根本的问题：当前结构是否已经成熟到“可以被稳定表达”。

因此，本文模型将“是否生成答案”视为状态判定的结果，而非推理过程的起点。

3 记号与参数总表

本节集中列出全文使用的核心函数、参数与阈值，并明确其语义角色与工程含义。

3.1 表示函数与向量空间

向量维度：

$$d \in \mathbb{N}.$$

Query 编码函数：

$$\phi_Q : \mathcal{Q} \to \mathbb{R}^d, z_q = \phi_Q(q).$$

记忆编码函数：

$$\phi_M : (q_i, v_i) \to \mathbb{R}^d, z_i = \phi_M(q_i, v_i).$$

说明如下：

- ϕ_Q与 ϕ_M可以共享或不共享参数；
- 本文不限定其训练方式或模型结构；
- 唯一要求是：输出向量位于同一可比较空间中，可用于相似度计算。

3.2 相似度与冲突判定

余弦相似度定义为：

$$\cos(u, v) \in [-1, 1],$$

用于衡量 query 向量与记忆向量之间的几何接近程度。

冲突指示函数定义为：

$$\chi(m_i, m_j) \in \{0, 1\}.$$

当 $\chi(m_i, m_j) = 1$时，表示在当前系统判定下，两条记忆被认为存在强冲突，难以在同一表达结构中共存。

说明如下：

- χ不要求基于形式逻辑；
- χ可由规则、统计模型或学习机制实现；
- 它只回答一个工程问题：这两条记忆放在一起是否会显著增加结构不确定性。

3.3 熵分解权重与阈值

熵分解权重定义为：

$$\alpha, \beta, \gamma \geq 0, \alpha + \beta + \gamma = 1.$$

其语义如下：

- α：对"相关性不足"的敏感度；
- β：对内部冲突的惩罚强度；
- γ：对历史路径风险的保守程度。

可表达阈值定义为：

$$\varepsilon > 0.$$

该阈值用于判定系统是否已进入可表达域。当系统熵低于该阈值时，系统被允许进入语言生成阶段。

3.4 迭代、容量与环境参数

最大推理步数：

$$T_{\max}.$$

该参数用于限制 System 2 推理循环的最深迭代次数，以防止无限计算。

记忆容量约束定义为：

$$\sum_{m_i \in \mathcal{M}} c_i \leq C,$$

其中 C表示系统允许的最大长期记忆资源上限。该约束是记忆熵守恒机制的基础。

环境经验筛选阈值定义为：

$$\varepsilon_{\mathrm{env}}.$$

该阈值用于判定虚拟环境生成的经验轨迹是否足够稳定、可抽象。只有当环境熵低于该阈值时，相关经验才允许写回长期记忆。

4 熵判别器（Entropy Discriminator）：结构成熟度的形式化判定

4.1 熵判别器的角色与函数定义

在本文模型中，熵判别器是唯一的停止判定机制。它不负责生成答案，也不参与语言建模；它只回答一个问题：

在当前 query 与候选记忆结构下，系统是否已经具备进入语言生成阶段的资格？

形式化地，熵判别器被定义为函数：

$$E: Q \times \mathcal{P}_{\text{fin}}(\mathcal{M}) \to \mathbb{R}_{\geq 0},$$

其中 $\mathcal{P}_{\text{fin}}(\mathcal{M})$表示 $\mathcal{M}$的有限子集集合。
$E(q, M)$表示系统在状态（q，M)下的**可表达性熵（expressibility entropy）**。

输入为：

- $q \in Q$：当前问题或任务；
- $M \subseteq \mathcal{M}$：当前候选记忆集合（有限）。

输出为：

- 一个非负标量 $E(q, M)$。数值越大，表示系统越不适合进入语言生成阶段；数值越小，表示系统越接近“可被稳定表达”的状态。

重要约束如下：

- 熵判别器的输入**不包含任何答案文本或语言生成结果**；
- 熵判别器评估的是**结构状态**，而非表述质量。

4.2 可表达性熵的认知含义

$E(q, M)$并非热力学熵或 Shannon 信息熵，而是一种面向认知与工程的状态不确定度量。它刻画的是：在 q的约束下，使用当前记忆集合 M进行表达时，系统将面临的结构性风险。

这些风险可归结为三类：

1. **覆盖不足（Coverage Gap）**：候选记忆与问题语义相关性过低，导致“答非所问”或信息缺口；

2. **内部冲突（Internal Conflict）**：候选记忆之间存在强冲突，导致拼凑式自洽或自相矛盾；
3. **路径不稳（Instability）**：即使覆盖充分、冲突较小，该结构在历史上仍反复失败或产生推理震荡。

熵判别器通过量化这三类风险，判断系统是否应继续推理（System 2），或停止并进入语言生成阶段。

4.3 熵函数的分解形式与权重

可表达性熵被定义为三个分量的加权和：

$$E(q, M) = \alpha \cdot E_{\text{cov}}(q, M) + \beta \cdot E_{\text{conf}}(M) + \gamma \cdot E_{\text{stab}}(q, M),$$

其中：

- $\alpha, \beta, \gamma \geq 0$，且满足 $\alpha + \beta + \gamma = 1$；
- 三个分量分别对应覆盖风险、冲突风险与稳定性风险。

尺度约定如下：为保证加权合理，约定各分量在适当归一化后位于 $[0,1]$区间；若某分量的原始取值超出该区间，应在进入线性加权前进行单调映射或裁剪。

4.4 覆盖熵 E_{cov}：相关性不足的度量

符号说明如下：

- $z_q = \phi_Q(q)$：query 的向量表示；
- 对任意记忆 $m \in M$，其向量表示记为 z_m；
- $\cos(u, v)$：余弦相似度，取值范围为 $[-1,1]$。

定义最大相关性为：

$$\text{sim}_{\max}(q, M) = \max_{m \in M} \cos(z_q, z_m).$$

并定义覆盖熵为：

$$E_{\text{cov}}(q, M) = 1 - \text{sim}_{\max}(q, M).$$

其直觉解释如下：

- 若至少存在一条记忆与 query 高度相关，则覆盖风险较低；
- 若所有记忆均与 query 不相关，则覆盖风险较高。

边界与工程处理约定如下：

- 若 $M = \emptyset$，定义 $\mathrm{sim}_{\max} = 0$，从而 $E_{\mathrm{cov}} = 1$；
- 若余弦相似度可能为负，为保持单调性，可采用裁剪版本：

$$\cos^{+}(u, v) = \max(0, \cos(u, v)).$$

4.5 冲突熵 E_{conf}：结构内在矛盾的度量

冲突指示函数定义为：

$$\chi(m_i, m_j) \in \{0,1\}.$$

当 $\chi(m_i, m_j) = 1$时，表示两条记忆在当前系统语境下被视为**强冲突**。需要强调的是，χ 不要求显式命题逻辑（与 / 或 / 非）；它只需回答一个工程问题：这两条记忆是否难以在同一表达结构中共存。

定义如下。设 $|M|$为集合 M的元素个数，则：

$$E_{\mathrm{conf}}(M) = \frac{1}{|M|(|M|-1)} \sum_{i \neq j} \chi(m_i, m_j),$$

并约定：

- 若 $|M| < 2$，则 $E_{\mathrm{conf}}(M) = 0$。

χ的来源可以包括：

- 基于规则或约束（如互斥属性、唯一性约束等）；
- 基于统计或学习方法（二分类器或冲突概率阈值化）；
- 基于结构一致性校验（字段级或关系级冲突）。

4.6 稳定性熵 E_{stab}：历史路径风险的度量

稳定性熵用于刻画一种不同于“当前结构”的风险：即便当前结构看似合理，它是否在历史上反复走不通。

系统维护一个历史映射：

$$H(\mathrm{cluster}(q), \mathrm{sig}(M)) \to p_{\mathrm{succ}} \in [0,1],$$

其中：

- $\mathrm{cluster}(q)$：对 query 的粗粒度聚类或分桶，用于减少过拟合；
- $\mathrm{sig}(M)$：集合 M的顺序无关签名（如 ID 排序哈希或近似签名）；
- p_{succ}：从该模式出发，系统在有限步内达到 $E(q, M) \le \varepsilon$的经验成功率。

定义稳定性熵为：

$$E_{\mathrm{stab}}(q, M) = 1 - p_{\mathrm{succ}}.$$

若历史中不存在该键，则使用先验成功率：

$$p_0 \in (0,1).$$

更新方式如下：一次推理完成后，可采用指数滑动平均更新：

$$p_{\mathrm{succ}} \leftarrow (1-\eta)\, p_{\mathrm{succ}} + \eta \cdot \mathbb{I}[\mathrm{success}],$$

其中 η为更新步长，$\mathbb{I}[\mathrm{success}]$ 为成功指示函数（成功时取值为 1，否则为 0）。

4.7 可表达域与停止条件

定义可表达域为：

$$\mathcal{R}_\varepsilon = \{(q, M) \mid E(q, M) \le \varepsilon\}.$$

若当前系统状态 $(q, M) \in \mathcal{R}_\varepsilon$，系统即停止 System 2 推理并进入语言生成阶段。

阈值 ε的语义如下：

- ε较小：系统更保守、更稳健；
- ε较大：系统更激进、响应更快但风险更高。

4.8 熵判别器与“熵记忆”的关系

熵判别器内部可维护辅助记忆（如 $(\mathrm{cluster}(q), \mathrm{sig}(M))$对应的经验熵或成功率），但这些记忆不构成答案监督。它们学习的是：**哪些结构值得继续推理，哪些结构应尽早放弃或修正**，而不是“答案应该长成什么样”。

5 S2 推理与记忆重构（REWRITE）：以熵下降为目标的结构加工过程

5.1 S2 的定位：它不是“慢速答案生成”

在本文模型中，System 2（S2）并不是“更复杂的生成器”，也不是“深度搜索答案空间”的过程。S2 的唯一职责是：**在当前系统状态尚未进入可表达域时，通过结构加工降低可表达性熵**。

形式上，S2 是一个以熵为目标函数的状态演化机制，其输出不是答案文本，而是新的记忆结构补充。

换言之：

- S2 不关心“说什么话”；
- S2 只关心“加什么结构，能让系统更接近能说话”。

5.2 S2 的输入、输出与允许动作

5.2.1 输入

S2 在第 t步接收以下输入：

- 当前 query：

$$q \in \mathcal{Q};$$

- 当前候选记忆集合：

$$M_t \subseteq \mathcal{M};$$

- 当前熵值：

$$E(q, M_t).$$

其前提条件为：

$$E(q, M_t) > \varepsilon,$$

即系统尚未进入可表达域。

5.2.2 输出

S2 的输出是一个有限的新增记忆集合：

$$\Delta M_t = \{m_1, \dots, m_k\}, k \geq 0,$$

其中每个 m_j都是新的 qv 记忆原子。

重要说明如下：

- S2 允许输出空集（表示当前无可行降熵方向）；
- S2 不输出答案文本；
- S2 的输出将被并入系统状态，而非立即表达。

5.2.3 允许动作集合（Action Space）

S2 只允许一种类型的动作：**REWRITE**。

这意味着：

- 不能删除已有记忆；
- 不能直接修改已有记忆内容；
- 只能通过“新增结构”来改变整体状态。

5.3 单调扩展原

系统状态的演化必须满足：

$$M_{t+1} = M_t \cup \Delta M_t.$$

这一约束被称为**单调扩展原则**。其含义是：所有已有记忆始终保留；新结构以“补丁”方式叠加，而非替换。

其动机包括：

- **认知一致性**：人类理解问题时，通常不会通过“删掉旧知识”推进思考，而是引入更高层解释来化解冲突；

- **动力系统稳定性**：允许丢弃会引入强烈回溯与震荡，导致推理路径不稳定、难以收敛；
- **工程可控性**：单调扩展使演化路径可追踪、可复现，避免隐式破坏状态。

5.4 REWRITE 的形式化定义

REWRITE 是 S2 唯一允许的推进操作，其目标是生成新的记忆，使系统熵严格下降。

5.4.1 熵下降约束（硬约束）

对任意一次 REWRITE，其输出 ΔM_t必须满足：

$$E(q, M_t \cup \Delta M_t) < E(q, M_t).$$

这是一个**必要条件**，而非软目标。

解释如下：REWRITE 不要求逻辑正确、不要求语义完整，甚至不要求可直接表达，但**必须在熵意义上改善系统状态**。

5.4.2 REWRITE 的语义角色

REWRITE 不等价于“生成结论”，而等价于：在当前认知结构中，引入一个新的解释方向，使整体结构更可表达。

因此，REWRITE 生成的 qv 记忆可以是：

- 条件化的（例如：“如果 X成立，那么 Y”）；
- 桥接式的（连接两个原本孤立的结构片段）；
- 抽象化的（将多个低层事实压缩为一个高层概念）；
- 约束性的（限制不合理的组合路径）。

5.5 REWRITE 的两类来源（逻辑产生的位置）

5.5.1 逻辑模板驱动的 REWRITE（经验型）

当系统中已存在可复用的逻辑记忆模板 ℓ时，REWRITE 可以直接调用：

$$m_{\text{new}} = \ell(q, M_t).$$

其中：

- ℓ是映射规则；
- ℓ来源于历史运行或虚拟环境训练；
- ℓ代表一种已被验证有效的降熵方式。

5.5.2 探索性 REWRITE（结构搜索型）

当无模板命中时，S2 进入探索性模式：对当前 M_t进行结构重组，生成候选记忆 m_{cand}，并通过熵判别器进行试探性评估。

形式化表示为：

$$\Delta M_t = \arg\min_{\Delta M \in \mathcal{C}(q, M_t)} E(q, M_t \cup \Delta M),$$

其中 $\mathcal{C}(q, M_t)$表示可生成的候选重构集合。

解释如下：

- 这是以熵为目标的结构搜索；
- 搜索空间由工程约束限定（如候选数量、搜索深度等）；
- 不要求全局最优，只要求局部下降。

5.6 S2 的失败情形与终止

S2 在以下情形下被认为**无法继续推进**：

- 对所有可生成的 ΔM，均有

$$E(q, M_t \cup \Delta M) \geq E(q, M_t);$$

- 达到最大迭代步数：

$$t = T_{\max}.$$

在上述情形下，系统可以选择：

- 停止推理并拒绝输出；

- 或标记该 (q, M_t)结构为高风险路径，供稳定性熵更新使用。

5.7 S2 输出与长期记忆的关系

S2 产生的新增记忆 ΔM_t具有双重角色：

- **短期作用**：立即改变当前状态，用于推动熵下降；
- **长期沉淀**：若在多次推理中被证明有效，可写回长期记忆库 $\mathcal{M}$，成为未来 S1 可直接召回的经验。

这使系统具备以下能力：

- S2 的计算成本可逐渐前移到 S1；
- 复杂推理路径可被“编译”为可复用结构。

5.8 为什么 S2 必然收敛或终止

在不做丢弃约束与熵下降约束共同作用下：

- $E(q, M_t)$构成严格下降或停滞序列；
- 熵的下界为 0；
- 且最大步数 $T_{\max}$有限。

因此，S2 必然在有限步内进入可表达域，或触发终止条件，从而保证系统不会陷入无限推理循环。

5.9 小结：S2 的本质

System 2 并不是“更慢的生成器”，而是一个在**记忆结构空间**中，以可表达性熵为能量函数，通过单调扩展逐步逼近稳定结构的加工过程。逻辑不是 S2 的前提，而是 S2 在长期运行中自然涌现的高效重构模式。

6 主流程：从输入到表达的完整推理路径

本节给出模型在一次实际推理任务中的完整运行流程定义。流程严格区分三个阶段：**S1 检索初始化 — S2 结构推理 — 语言表达（Generation）**，并通过熵判别器实现阶段切换。

6.1 推理问题的形式化描述

一次推理任务由 query 触发：

$$q \in \mathcal{Q}.$$

系统目标不是立即生成答案，而是构造一个有限记忆集合 $M^{\setminus *}$，使得：

$$(q, M^{\setminus *}) \in \mathcal{R}_{\varepsilon},$$

其中

$$\mathcal{R}_{\varepsilon} = \{(q, M) \mid E(q, M) \leq \varepsilon\}$$

为可表达域。

语言生成仅在该条件满足后发生。

6.2 阶段一：S1 初始化检索（快速联想）

System 1（S1）是一个无状态、无迭代的检索算子：

$$S_1 : \mathcal{Q} \to \mathcal{P}_{\text{fin}}(\mathcal{M}).$$

给定 query q，S1 返回初始候选记忆集合：

$$M_0 = S_1(q).$$

S1 的角色与约束如下：

- 只负责快速召回；
- 不保证覆盖充分；

- 不保证无冲突;
- 不进行熵优化;
- 工程上通常基于向量相似度、倒排索引或规则过滤实现。

6.3 阶段二：熵判别与阶段切换判定

对当前系统状态 (q, M_t)，计算其可表达性熵:

$$e_t = E(q, M_t).$$

根据阈值 ε进行判定:

- 若

$$e_t \leq \varepsilon,$$

则状态可表达，系统进入语言生成阶段;

- 若

$$e_t > \varepsilon,$$

则状态不可表达，系统进入 System 2 结构推理阶段。

这是系统中**唯一的阶段切换条件**。

6.4 阶段三：S2 结构推理循环（REWRITE 驱动）

当 $e_t > \varepsilon$时，系统进入 S2 推理循环。

S2 在第 t步的状态表示为:

$$(q, M_t, e_t), e_t = E(q, M_t).$$

单步更新规则

S2 在当前状态下执行 REWRITE 操作:

$$\Delta M_t = R(q, M_t),$$

并要求满足熵下降约束：

$$E(q, M_t \cup \Delta M_t) < E(q, M_t).$$

随后更新系统状态：

$$M_{t+1} = M_t \cup \Delta M_t,$$

并重新计算熵值：

$$e_{t+1} = E(q, M_{t+1}).$$

推理循环的形式化描述（伪代码）

t ← 0

M_0 ← $S_1(q)$

while (E(q, M_t) > ε) and (t < T_max):

    ΔM_t ← R(q, M_t)

    if ΔM_t = ∅:

        break

    M_{t+1} ← M_t ∪ ΔM_t

    t ← t + 1

终止情形

S2 推理循环可能以以下两种方式终止：

- **成功终止**：

$$E(q, M_t) \leq \varepsilon,$$

系统进入可表达域；

- **失败终止**：

$$t = T_{\max}或\ \Delta M_t = \emptyset.$$

失败终止的状态可被记录，用于更新稳定性熵中的历史成功率。

6.5 阶段四：语言生成（表达阶段）

当 S2 成功终止时，系统获得最终记忆集合：

$$M^{\backslash *} = M_t, E(q, M^{\backslash *}) \leq \varepsilon.$$

此时才调用语言生成器：

$$G: \mathcal{Q} \times \mathcal{P}_{\text{fin}}(\mathcal{M}) \to \mathcal{Y},$$

生成输出文本：

$$y = G(q, M^{\backslash *}).$$

6.6 语言生成器的约束与边界

语言生成器满足以下约束：

- 不参与 E的计算；
- 不触发 REWRITE；
- 不影响 S2 迭代；
- 仅负责将 $(q, M^{\backslash *})$表达为自然语言。

因此：

- 生成质量的好坏不影响系统是否“该说”；
- 生成错误不会污染推理路径；
- 推理与表达在结构上被严格解耦。

6.7 记忆写回与经验沉淀（可选）

一次推理完成后，系统可选择将部分新增记忆写回长期记忆库：

$$\mathcal{M} \leftarrow \mathcal{M} \cup \widehat{M},$$

其中：

$$\widehat{M} \subseteq (M^{\setminus *} \setminus M_0).$$

写回条件可包括：

- 在多次推理中反复出现；
- 显著降低平均熵；
- 通过虚拟环境或真实任务验证。

6.8 主流程的整体性质

该主流程具备以下性质：

- **可终止性**：熵严格下降 + 下界存在 + 最大步数约束 ⇒有限步终止；
- **可控性**：ε、$T_{\max}$、α、β、γ 显式控制系统行为风格；
- **模块解耦**：检索、推理、生成三阶段职责清晰，互不污染；
- **可演化性**：S2 的计算成果可逐步前移为 S1 的经验，加速未来推理。

6.9 小结

本节定义了一条从 query 输入到语言输出的完整、可执行的推理主流程：系统先通过熵判别确认**是否已具备表达资格**，再通过结构化推理降低不确定性，最终才进入语言生成阶段。这使“能不能回答”成为一个显式、可学习、可调控的数学对象，而不再是语言模型内部的隐

7 虚拟环境（Virtual Environment）：在低成本条件下学习降熵模式

7.1 为什么需要虚拟环境

在主流程中，推理能力高度依赖 System 2 的 REWRITE 是否能持续降低熵。然而，仅依靠真实任务驱动存在三个根本问题：

- **真实问题分布稀疏且昂贵**：高价值 query 出现频率低，且失败代价高，不适合作为探索场；
- **熵下降是结构性质，而不是答案性质**：若训练直接绑定答案监督，会把“表达成功”误学为“文本相似度最大”；
- **S2 的学习信号是延迟的**：REWRITE 是否有效，只有在多步熵演化后才能判断。

因此，需要一个可控、可重复、低成本的试验空间，用于批量生成“可学习的结构经验”。本文将这一空间统称为**虚拟环境（Virtual Environment）**。

7.2 虚拟环境的角色定位

虚拟环境在本模型中的定位是：一个用于产生**“结构—熵—演化轨迹”**的机制，而不是世界仿真器。

它不要求逼真还原物理或社会世界，也不要求生成真实可用的答案；它只要求能生成足够多的**状态—动作—结构变化序列**，使系统学会：**哪些结构变化在统计上更容易降低熵**。

7.3 虚拟环境的形式化定义

一个虚拟环境定义为四元组：

$$\mathcal{E}_{\mathrm{env}} = (X, A, f, g),$$

其中：

- X：环境状态空间；
- A：可执行动作空间；
- $f: X \times A \to X$：状态转移函数；

- $g: X \to O$：观测映射。

说明如下：

- X可以是抽象状态（如符号结构、约束集合、关系图）；
- A可以是操作符（如合并、条件化、桥接、抽象）；
- 不要求 f可逆或确定。

7.4 从环境轨迹到推理经验

虚拟环境运行生成一条轨迹：

$$\tau = (x_0, a_0, x_1, a_1, \dots, x_T).$$

轨迹本身不直接进入推理系统，而是作为经验原材料。

系统通过经验抽象算子：

$$G_{\text{env}}: \tau \to m_{\text{env}},$$

生成一条或多条 qv 记忆：

$$m_{\text{env}} = \langle q_{\text{env}}, v_{\text{env}}, z_{\text{env}}, c_{\text{env}}, s_{\text{env}} \rangle.$$

其中：

- $c_{\text{env}} \in (0, +\infty)$ 或 $[0, +\infty)$：经验写回后的资源 / 容量占用（与 c_i同语义）；
- $s_{\text{env}} \in [-1, 1]$：方向性价值（支持 / 避免），与 s_i同语义。

关键点在于：m_{env} 表达的是**“在某类结构条件下，某种重构方向往往有效”**，而不是“答案是什么”。

统一性说明：m_{env} 与一般记忆 m_i属于同一类对象，均属于 $\mathcal{M}$。下文在需要强调来源时使用下标 env，否则一律视为 m_i。

7.5 环境熵与经验筛选（防止噪声污染）

并非所有环境轨迹都值得写入长期记忆。为此引入**环境熵判定**：

$$E_{\text{env}}(\tau).$$

该量刻画轨迹是否具备以下特性：

- 结构演化稳定；
- 存在持续的降熵趋势；
- 可被压缩为低复杂度经验。

写回条件定义为：

$$E_{\text{env}}(\tau) \leq \varepsilon_{\text{env}}.$$

仅当满足该条件时，由 τ生成的 m_{env}才允许进入长期记忆库 $\mathcal{M}$。

7.6 训练目标：学习“熵下降能力”而非答案

训练对象不是语言生成器 G，而是以下结构组件：

- REWRITE 策略 R；
- 冲突判定函数 χ（可选）；
- 稳定性映射 H；
- 经验抽象算子 G_{env}。

核心训练目标定义为：**最大化新增结构带来的期望熵下降**：

$$\max \ \mathbb{E}_{(q,M)}[E(q,M) - E(q,M \cup \{m_{\text{env}}\})].$$

其解释如下：

- 若某一经验在多种结构状态下都能稳定降低熵，则其期望收益较高；
- 若只在个别轨迹中有效，其期望收益会被平均抵消。

7.7 训练如何反哺主流程

虚拟环境训练的成果通过以下方式反哺主流程：

- **S1 增强**：有效经验写回长期记忆，使 S1 更易召回接近可表达域的初始结构；

- **S2 加速**：逻辑模板命中率提高，减少探索性 REWRITE 搜索；
- **熵判别更稳健**：稳定性熵中的成功率逐步收敛，使系统更早规避高风险结构。

7.8 训练与推理的严格解耦

训练阶段**不改变推理规则本身**：

- 熵定义 E不变；
- 停止阈值 ε不放宽；

训练只改变：**哪些 REWRITE 更容易被提出、哪些结构更容易被识别为稳定**。

7.9 小结：为什么系统可以“越用越会想”

推理产生结构轨迹；
结构轨迹被抽象为经验；
经验通过环境熵筛选；
有效经验写回长期记忆；
长期记忆反过来加速未来推理。

系统不是“越训练越会说”，而是：**越运行，越擅长让自己进入“可以说”的状态**。

8 记忆熵守恒：长期认知系统的稳定性约束

8.1 问题动机：为什么“记得越多”反而是风险

在允许持续学习与记忆写回的系统中，最直观的风险并非推理失败，而是**记忆失控**：记忆条目数量随时间增长；冗余、相似、低价值记忆不断堆积；检索与推理成本上升；系统最终被自身历史拖慢甚至“淹没”。

传统系统通常采用显式删除或外部压缩（如离线重训、人工清洗）来应对这一问题。本文明确拒绝这两种方式：删除会破坏推理路径的单调性与可解释性；外部压缩则破坏系统内部的自洽闭环。

因此，本文引入一种更严格的长期约束：**记忆熵守恒**。

8.2 记忆熵的定义（不是记忆条目数）

记忆熵并不等价于记忆数量 $|\mathcal{M}|$。

在本文模型中，每一条记忆 m_i都携带一个资源权重：

$$c_i > 0,$$

该权重综合刻画了存储成本、检索影响、推理干扰度以及长期维护复杂度。

系统的记忆总熵（或总资源占用）定义为：

$$S(\mathcal{M}) = \sum_{m_i \in \mathcal{M}} c_i \,.$$

并施加如下硬约束：

$$S(\mathcal{M}) \le C,$$

其中 C表示系统在工程上可承受的最大记忆容量上限。

8.3 守恒原则的核心思想

记忆熵守恒的核心思想在于：**系统允许记忆形式发生变化，但不允许无代价的记忆增长**。

系统可以引入新记忆、改变表达形态、重构结构，但所有变化必须满足总资源约束。换言之，能力提升必须通过**结构效率的提高**来实现，而不能依赖简单的记忆累积。

8.4 为什么禁止删除，而允许“方向重写”

在第 5 节中，系统明确禁止删除操作，但允许通过 REWRITE 引入新结构。

二者的根本差异在于：

- **删除**：直接移除记忆 m_i及其对应的 c_i，导致历史路径信息不可逆丢失；
- **REWRITE / 抽象**：用新的高层记忆 m'替代一组低层记忆 $\{m_i\}$，同时满足

$$c(m') \leq \sum_i c(m_i),$$

即信息被压缩，但总体熵预算未被突破。

8.5 抽象压缩作为守恒机制

当系统检测到：

$$S(\mathcal{M}) \to C,$$

即记忆总熵接近容量上限时，系统不会触发删除，而是触发**抽象压缩操作**。

定义压缩算子：

$$\mathcal{A}: \{m_{i_1}, \dots, m_{i_k}\} \to m_{\text{abs}},$$

并要求满足：

$$c_{\text{abs}} \leq \sum_{j=1}^{k} c_{i_j}.$$

在功能层面，要求：

$$\forall q, E(q, M \cup \{m_{\text{abs}}\}) \approx E(q, M \cup \{m_{i_1}, \dots, m_{i_k}\}).$$

其含义是：抽象后的记忆在熵意义上保留了原记忆集合的“降熵能力”，但显著降低了结

构复杂度。

8.6 记忆“方向”的概念（方向性记忆）

需要特别指出的是：在本文模型中，记忆不仅仅是“是否存在”的对象，还携带**方向性**。

在第 2.2 节中，方向性已作为

$$s_i \in [-1,1]$$

纳入每一条记忆 m_i的定义，其语义如下：

- $s_i \approx 0$：几乎无意义 / 不确定 / 暂不使用；
- $s_i > 0$：倾向使用（支持性记忆）；
- $s_i < 0$：倾向避免（反向记忆 / 禁忌 / 反例 / 陷阱提示）；
- $|s_i|$：方向强度。

需要强调的是：与资源占用 c_i不同，s_i **不参与容量守恒约束**。容量约束仅由 c_i决定，而 s_i用于表达“该结构在推理中更像支持项还是避险项”。

这使得**负向记忆**可以在系统中合法存在，同时不破坏记忆熵守恒机制的工程边界。

8.7 守恒与学习并不矛盾

记忆熵守恒并不会抑制系统学习能力，反而迫使系统学习**更高效的结构表达方式**。学习的动力从“记得更多”转向“记得更好”。

这正是 System 2 中 REWRITE 机制在长期运行中所承受的持续压力来源。

8.8 记忆熵守恒与人类认知的类比

在人类认知中，童年记忆碎片繁多；随着经验增长，这些碎片被压缩为概念、原则与直觉。具体细节并未完全消失，而是被“折叠”进高层结构之中。

记忆熵守恒正是对这一现象的形式化表达。

8.9 记忆熵守恒对系统整体的影响

在引入记忆熵守恒约束后，系统在长期运行中具备以下性质：

- **容量有界性**：记忆规模不会无限增长；
- **推理效率可控**：S1 / S2 的复杂度不会随时间失控；
- **结构自洽性**：推理路径不会因历史清洗而断裂；
- **学习方向明确**：系统被迫向更抽象、更通用的结构演化。

8.10 小结

记忆熵守恒为本文模型提供了长期稳定性的理论下界。它确保系统在持续学习、持续写回、持续推理的前提下，仍能保持结构紧凑、路径可追溯、复杂度受控。

在这一约束下，学习不再是“多记一点”，而是**“用同样的熵预算，表达更多的世界”**。

9 讨论（Discussion）

9.1 关于“熵”作为核心判据的合理性

本文将“是否进入可表达状态”的判定统一抽象为**可表达性熵** $E(q,M)$，这是一种**结构层面的风险度量**，而非语言层面的不确定性。

其主要优势在于：

- **不依赖具体输出形式**，天然支持多模态或非语言系统；
- 将“是否该说”从“说什么”中剥离，避免语言模型在结构尚未成熟时过早承担推理责任；
- 为推理过程提供一个**连续、可比较、可优化的标量目标**，使推理具备工程可调性。

但需要明确指出：

- $E(q,M)$的具体形式并非唯一；
- 不同任务、不同风险偏好可能需要不同的分解项或权重配置；
- 熵并不试图“解释一切”，它只是一个在工程与认知之间具有高度可操作性的判据。

本文的立场是：在工程实现与认知抽象之间，熵是一个足够通用且可执行的中介量。

9.2 参数极端化下的系统行为

模型中的关键参数对系统行为具有决定性影响，其极端取值能够揭示系统的内在机制边界。

- 当 $\varepsilon \to 0$时：
  系统极端保守，几乎拒绝所有输出；S2 推理深度显著增加，稳定性最高，但响应成本较大。
- 当 $\varepsilon \to 1$时：
  系统极端激进，行为逐渐接近传统语言模型；S2 很少触发，但结构风险显著增加。
- 当 $\alpha \gg \beta, \gamma$时：
  系统主要关注“是否相关”，而对内部冲突与历史风险不敏感，容易产生**表面相关但结构混乱**的输出。
- 当 $\gamma \gg \alpha, \beta$时：

系统强烈依赖历史经验，路径依赖与保守性增强，适合高风险、低容错场景，但创新能力下降。

这些极端情况表明：本文模型并非参数无关的黑箱系统，而是一个**可通过参数显式调节认知风格**的结构化推理框架。

9.3 与现有范式的关系

从整体范式上看，本文模型并不试图替代现有主流方法，而是提供一种更底层的认知约束层。

- **与 Transformer / 大规模语言模型**：
  后者以生成概率最大化为核心；本文以前置的"是否可表达"判定作为约束，语言生成被严格后置。
- **与 RAG（Retrieval-Augmented Generation）**：
  RAG 中检索结果直接参与生成；本文中，检索结果必须先通过熵判别与结构重构，才能进入生成阶段。
- **与 Planner–Executor 架构**：
  Planner 通常依赖显式任务分解或规划语言；本文的 S2 不依赖显式规划符号，其"逻辑"以降熵模式的形式隐式存在。

因此，本文模型更适合作为一种**认知结构层**，与生成模型、检索系统或执行模块组合使用，而非独立替代它们。

9.4 逻辑的"非显式性"与可解释性问题

本文刻意避免预设显式逻辑符号系统（如 与 / 或 / 非），这一选择带来了明显的取舍。

其优势在于：

- 逻辑结构可以从经验中自然涌现，而非被硬编码；
- 系统不受特定形式逻辑限制，具有更强的泛化潜力。

其代价在于：

- 单次推理过程的可解释性不如显式逻辑系统；
- REWRITE 的生成动机可能难以用传统逻辑语言完整描述。

本文的立场是：

可解释性应体现在“为什么某种结构变化降低了熵”，而非“是否符合某种预设逻辑语法”。

9.5 潜在局限与开放问题

尽管本文框架在结构上具有一致性与可执行性，但仍存在若干开放问题：

- **熵函数设计的任务依赖性**：不同领域需要不同的冲突判定方式或稳定性度量；
- **REWRITE 搜索空间的规模控制**：如何在保证降熵能力的同时限制搜索复杂度；
- **虚拟环境的构造方法**：如何设计既足够抽象、又不过拟合特定任务分布的环境。

这些问题为后续研究提供了明确且可操作的方向。

10 结论（Conclusion）

本文提出了一种以**可表达性熵**为核心判据、以**记忆重构（REWRITE）**为动力的认知模型框架。

该框架的核心特征包括：

- 将“是否进入语言生成阶段”显式建模为一个**可计算、可学习的状态判定问题**，而非隐藏在生成概率分布中的隐含结果；
- 通过 System 2 结构推理，在记忆空间中沿**熵下降方向**进行单调扩展，使系统状态逐步收敛到可被稳定表达的区域；
- 通过虚拟环境训练，使在统计意义上有效的降熵结构逐步沉淀为可复用经验，从而将计算成本前移至快速检索阶段；
- 通过**记忆熵守恒约束**，确保系统在长期运行、持续学习与持续写回的条件下，仍保持容量有界、推理可控与结构自洽。

与以生成概率最大化为中心的主流模型范式不同，本文强调：**推理的本质不在于生成答案，而在于让系统状态变得“可以被稳定表达”**。语言生成只是认知过程的最后一步，而不应承担认知判断与风险控制的核心职责。

本文并不主张该模型立即替代现有的大规模语言模型，而是将其视为一种**认知结构层面的补充与约束机制**。在这一视角下，该框架既可以作为独立的推理内核存在，也可以与现有生成模型、检索增强系统或执行模块协同工作，为构建更稳健、更可控、具备长期演化能力的智能系统提供一种新的结构路径。

参考文献

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need*. In **Advances in Neural Information Processing Systems (NeurIPS 2017)**.

[2] Simon, H. A. (1955). *A Behavioral Model of Rational Choice*. **The Quarterly Journal of Economics**, 69(1), 99–118.

[3] Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.