Soft Thinking: 通过概率加权概念标记探索多条推理路径

```
# 参考来源：
1. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
https://arxiv.org/pdf/2201.11903v6
2. Soft Thinking: Unlocking the Reasoning Potential of LLMs in Continuous Concept Space
https://arxiv.org/abs/2505.15778
```

1. 背景：为什么要摆脱“离散 token”限制?

大语言模型（LLM）在复杂推理任务中大放异彩，而 **链式思考**（Chain-of-Thought, CoT）通过生成中间步骤进一步提升了推理表现。但标准 CoT 依然存在四大痛点：

1 离散符号受限

- 每步仅选出一个固定 token，表达粒度受限于人为定义的词表。

2 背离人类认知

- 神经科学研究表明，人脑推理依赖抽象概念而非单个词语。

3 早期决策 + 单路径

- 序列式采样迫使模型过早锁定一路径，一旦偏离就难以回溯。

4 高不确定任务易失误

- 多条可行路线时，模型可能走错分支，既浪费 token 也降低准确率。

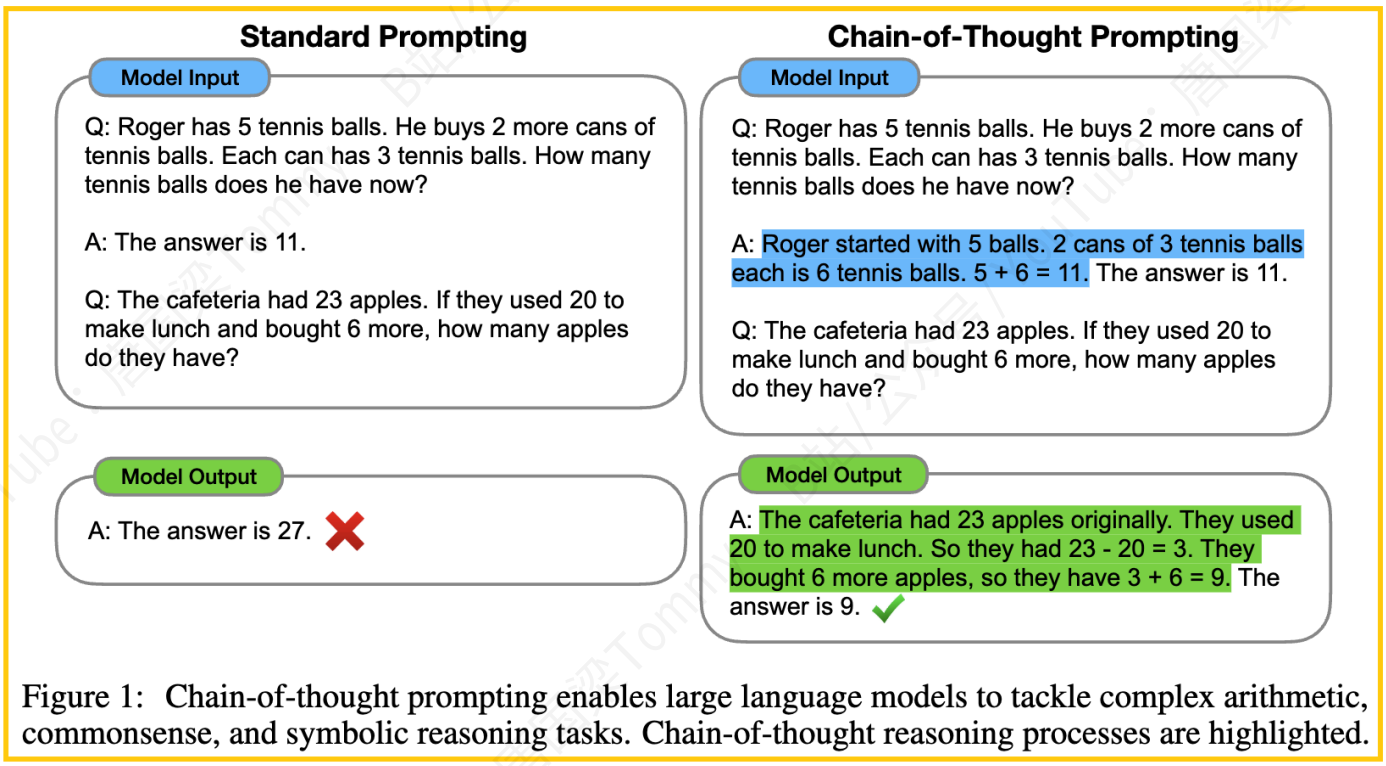
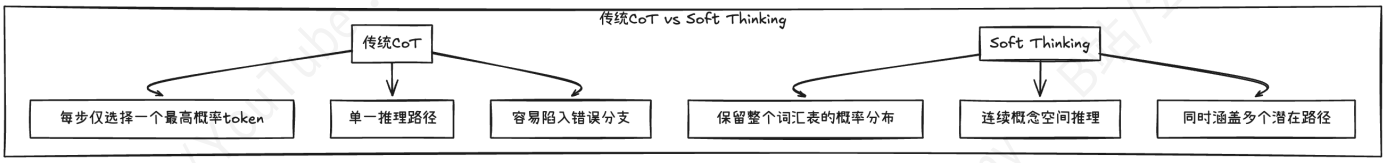


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

💡💡💡 **Soft Thinking：走向 连续概念空间**

为打破离散局限，论文提出 **Soft Thinking** —— 一个 **无需额外训练** 的推理范式。

- 为了解决LLMs在离散空间推理的限制，论文提出了 **Soft Thinking** 方法。这是一个**无需额外训练（training-free）**的方法，旨在让LLMs能够在**连续的概念空间（continuous concept space）**中进行推理。
- Soft Thinking的**核心创新点**在于，它**取代了标准CoT中离散的词语选择过程**。在标准的CoT中，模型在每一步推理后会生成一个对词汇表所有词语的概率分布，然后从中"硬"地选择（采样）一个词语作为下一步的输入。Soft Thinking则不同，它**保留了原始完整的概率分布**。



2. Soft Thinking 的三个关键词

2.1 概念 token (Concept Token)

在任何中间推理步骤，LLM 会输出一个关于整个词汇表上所有标记的概率分布 p 。

这个概率分布 p 本身就被称为一个概念标记 ($ct := p$)

$ct := p \in \Delta^{|V|-1}$

与传统方法强制将分布塌缩到单个标记不同，概念标记保留了每个可能下一步的完整分布信息。这保留了每一步的"叠加态"信息。

2.2 连续概念空间

通过对词汇表中所有词元的嵌入向量按该概率加权求和，即可构造连续的概念空间。

$C = \left\{ \sum_k \alpha_k \, e^{(k)} \;\middle|\; \alpha \in \Delta^{|V|-1} \right\}$

其中 $e(k)$ 是第 k 个词汇项的嵌入向量，α 是概率分布。软思考在这个连续概念空间中进行推理。

Soft Thinking 利用这种概率加权插值，使概念 token 表示更加平滑丰富，超越了单个离散词汇的表达能力。

- **连续概念空间中的推理过程（Reasoning Process）：**

在Soft Thinking框架下，每一步推理生成一个概念 token 后，将其"再输入"模型继续推理。

具体过程是：

1️⃣ 模型先输出下一个词元的概率分布 p（定义为概念 token：ct）；

2️⃣ 然后计算该概念 token 的嵌入 $\tilde{e} = \sum_k p[k]\, e^{(k)}$ 并将其作为下一个输入向量。这样，模型实际上在连续概念空间中前进。

3️⃣ 只要当前概念 token 的最高概率词元不是结束标记（〈/think〉），推理过程就继续进行。一旦最高概率词元为“结束思考”，模型终止中间推理阶段，转入按普通方式生成最终答案的输出阶段。

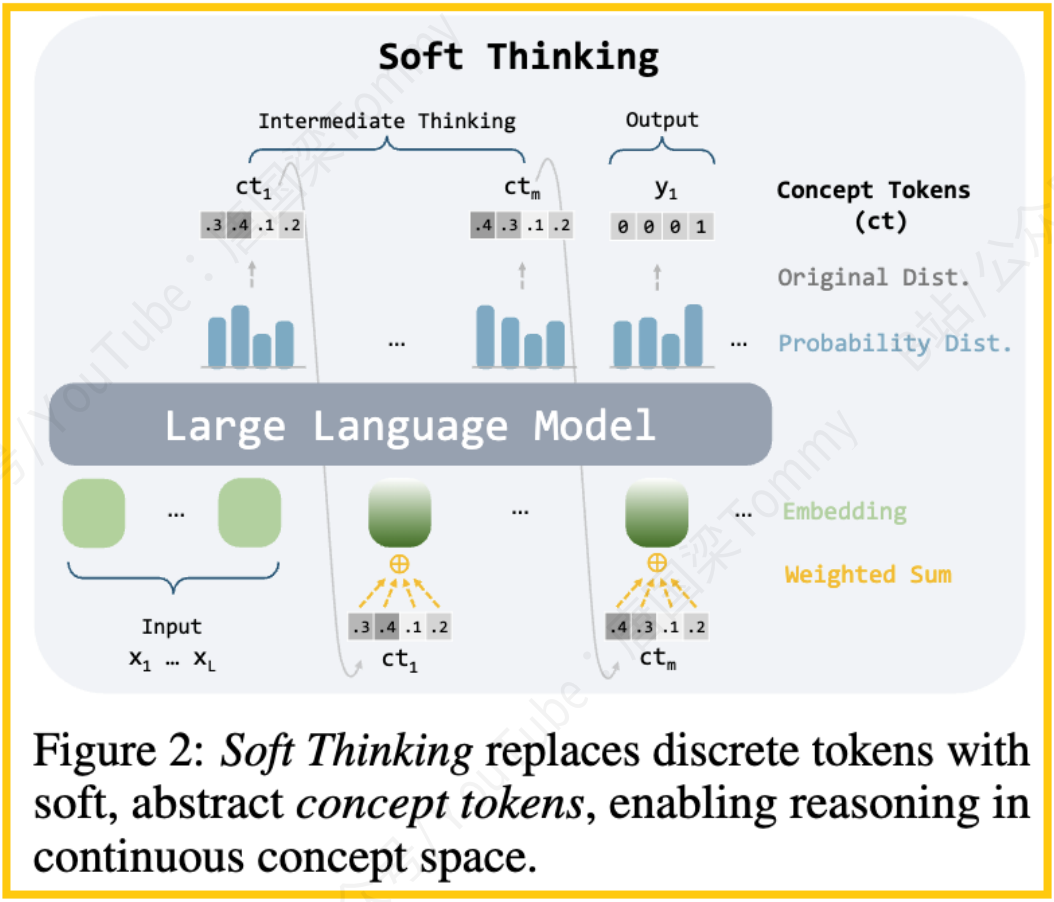


Figure 2: *Soft Thinking* replaces discrete tokens with soft, abstract *concept tokens*, enabling reasoning in continuous concept space.

> 如图2所示，Soft Thinking 用概念标记（表示概率分布）取代了离散词语，并将这个概率加权计算出的连续嵌入作为模型的输入，驱动下一步的思考。与离散CoT通过采样单个词语丢弃其他路径信息不同，Soft Thinking 通过概念标记**保留了每一步完整的概率分布**。

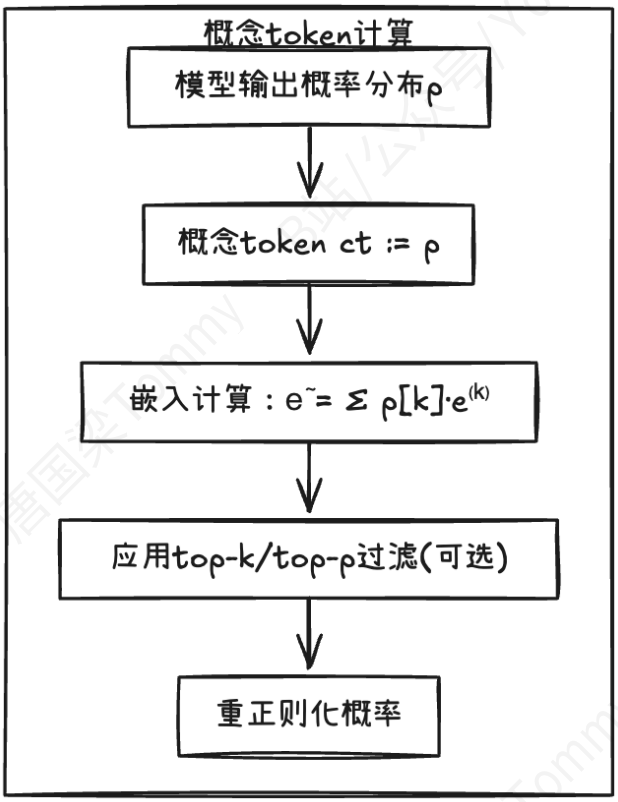
- **概念 token 的数学定义与嵌入计算：**

概念 token 即模型在当前时刻对下一个词元分布的概率向量 p，其对应的嵌入向量通过概率加权求和得到，即

$\tilde{e}_{\text{next}} = \sum_{k=1}^{|V|} p[k]\, e^{(k)} \in \mathbb{R}^d$

这里 $e^{(k)}$ 是第 k 个词元的嵌入向量，这个加权和得到的 $\tilde{e}_{next}$ 就位于连续概念空间 C 中。

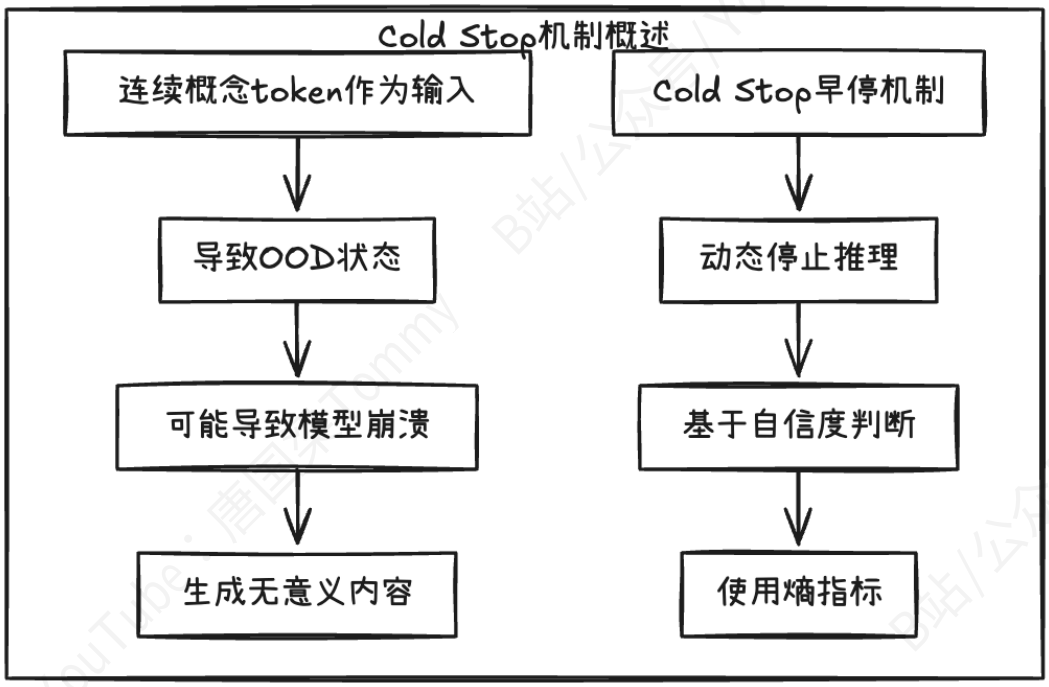
为了减少计算量，实际实现时通常会先对概率分布做 top-k/top-p 过滤，保留概率最高的若干词元，再重正则化并计算加权和。整个过程仅利用了原有模型的嵌入矩阵，无需额外学习参数。



2.3 Cold Stop（冷停止）

（1）问题背景

- 虽然概念标记使得抽象推理成为可能，但将连续的概念标记作为输入馈送给模型，实际上是将模型置于一个**分布外（Out-of-Distribution, OOD）**的状态。
- 这是因为LLMs通常只在离散词语序列上进行训练。如果推理过程过长，这种OOD输入可能导致**模型崩溃（model collapse）**，例如开始重复生成无意义的内容。



（2）解决方案

- 引入了基于熵的**Cold Stop**早停机制，这个机制会**动态地停止中间推理过程**，判断依据是模型何时变得“过度自信”。
- 具体来说，Soft Thinking 保留了每一步完整的概率分布，因此可以计算概念标记的**熵（Entropy）**：

  $H(p) = -\sum_{k=1}^{|V|} p[k] \log p[k]$

> 熵是衡量不确定性的一个自然指标，低熵（类似于物理学中的“冷”）表明模型对其预测非常自信，此时可以尽快结束思考。

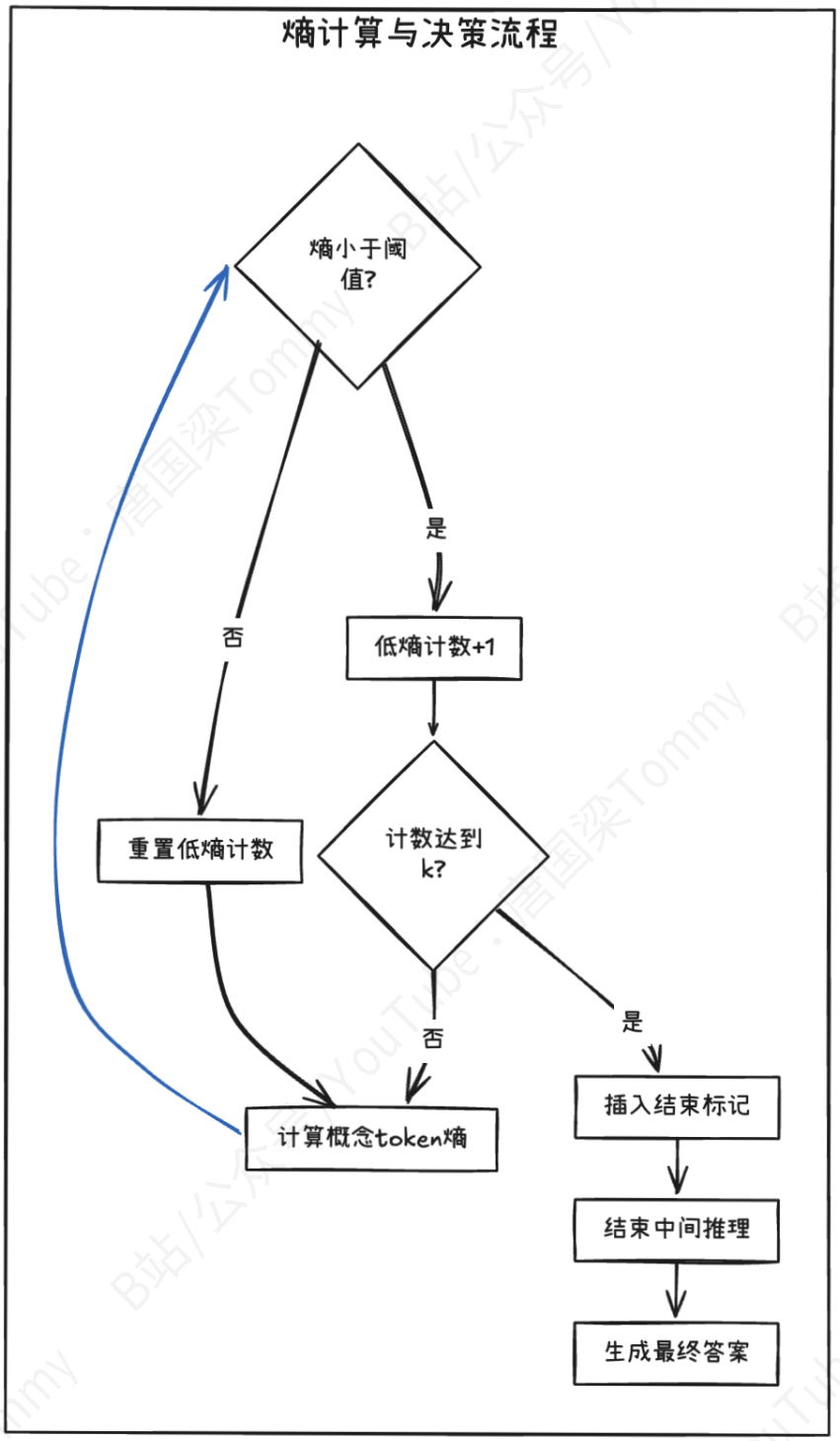
Cold Stop机制设定一个熵阈值 τ 和一个所需的连续低熵步骤数 k。

- **早停规则：**

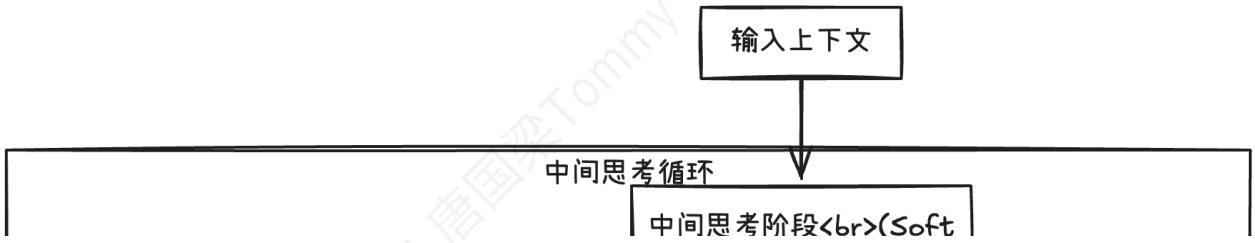
如果当前步骤的熵 $H(p) < \tau$，就增加低熵步骤计数器；否则重置计数器。

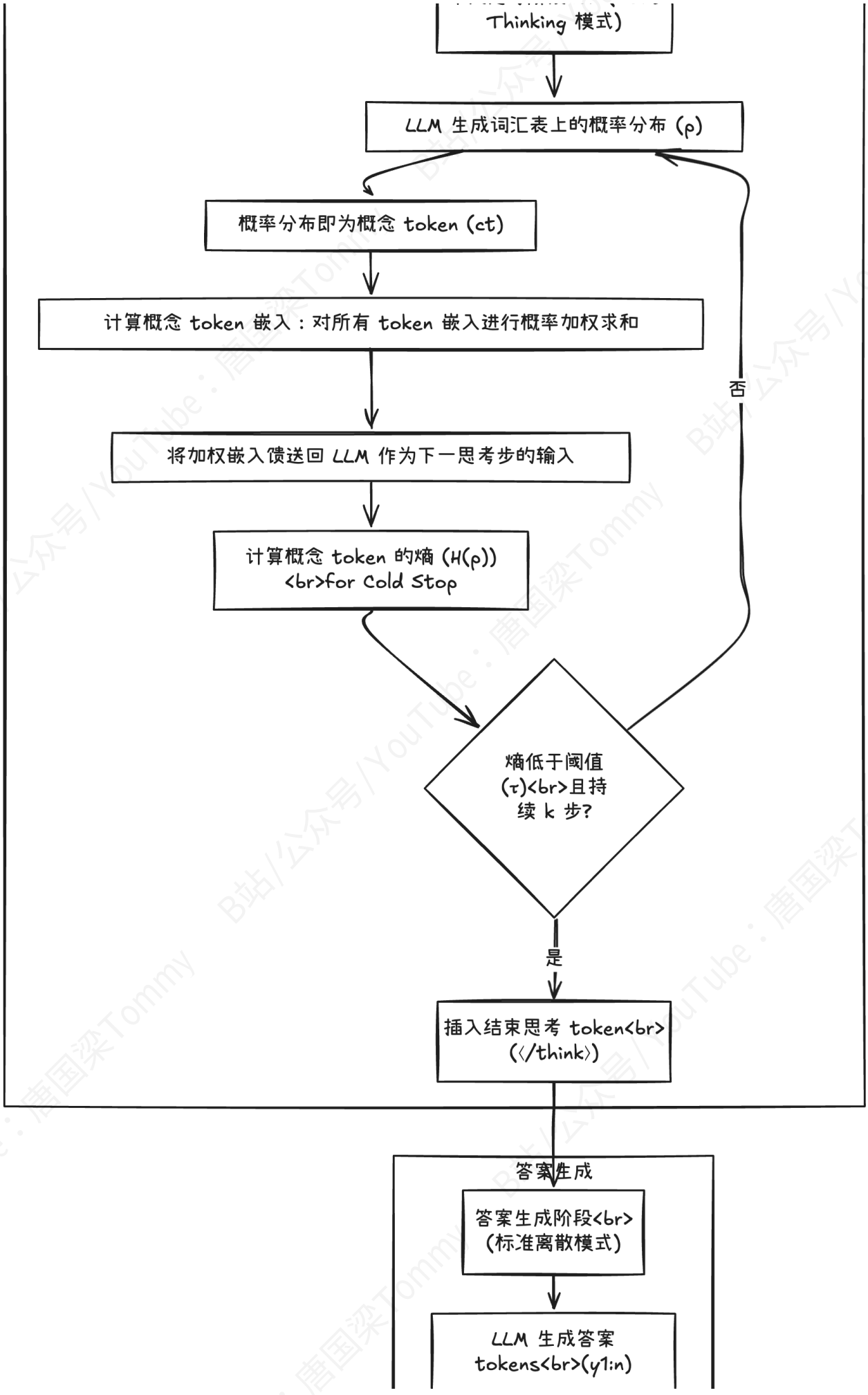
当计数器达到 k 时，强制插入一个“思考结束”词语 $\langle /think \rangle$，结束推理并开始生成最终答案。

这个机制避免了不必要的计算，并在处理OOD输入时减轻了模型崩溃的风险，提高了推理的鲁棒性和效率。

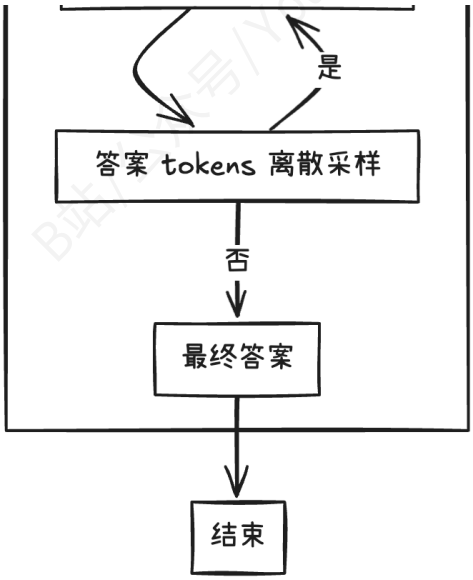

熵计算与决策流程
熵小于阈值?
是
否
低熵计数+1
计数达到k?
重置低熵计数
否
是
计算概念token熵
插入结束标记
结束中间推理
生成最终答案

3. 端到端流程图


输入上下文
中间思考循环
中间思考阶段
(Soft


Thinking 模式)
LLM 生成词汇表上的概率分布 (p)
概率分布即为概念 token (ct)
计算概念 token 嵌入：对所有 token 嵌入进行概率加权求和
将加权嵌入馈送回 LLM 作为下一思考步的输入
计算概念 token 的熵 (H(p))

for Cold Stop
熵低于阈值
(τ)
且持
续 k 步?
否
是
插入结束思考 token
(〈/think〉)
答案生成
答案生成阶段
(标准离散模式)
LLM 生成答案
tokens
(y1:n)



4. 实验结果与分析

实验选用了三款主流的开源LLM模型：**QwQ-32B**、**DeepSeek-R1-Distill-Qwen-32B**、**DeepSeek-R1-Distill-Llama-70B**。这些模型参数规模涵盖32B和70B，架构包括Qwen和LLaMA，训练范式也不同（RL训练和蒸馏训练），这保证了 Soft Thinking 方法的通用性和泛化能力。

对比的基线方法有两个：

1. Standard CoT Thinking：标准的、生成显式中间步骤的思维链方法。

2. Standard Greedy CoT Thinking：在推理过程的每一步都使用贪婪解码（greedy decoding）的标准CoT方法。

评估指标有两个：

- **Pass@1 准确率**：衡量模型生成正确答案的能力。Pass@1 等于正确解决的问题数量除以总问题数量。
- **生成长度（针对正确解）**：衡量模型推理的效率。统计模型生成正确答案时所使用的词语数量。这两个指标有助于评估计算成本和性能之间的权衡。

4.1 定量结果

4.1.1 数学基准

数学基准包括：Math500、AIME 2024、GSM8K、GPQA-Diamond。这些任务难度各异，从小学算术到研究生级别的问答都有涉及。

	Accuracy ↑					Generation Length ↓				
	MATH 500	AIME 2024	GSM8K	GPQA Diamond	Avg.	MATH 500	AIME 2024	GSM8K	GPQA Diamond	Avg.
	QwQ-32B [13]									
CoT Thinking	97.66	76.88	96.67	64.17	83.84	4156	12080	1556	8095	6472
CoT Thinking (Greedy)	97.00	80.00	96.57	65.15	84.68 (↑ 0.84)	3827	11086	1536	7417	5967 (↓ 7.8%)
Soft Thinking	**98.00**	**83.33**	**96.81**	**67.17**	**86.32** (↑ 2.48)	**3644**	**10627**	**1391**	**7213**	**5719** (↓ 11.6%)
	DeepSeek-R1-Distill-Qwen-32B [38]									
CoT Thinking	94.50	72.08	95.61	63.10	81.32	3543	9347	875	6218	4995
CoT Thinking (Greedy)	93.00	63.33	95.30	59.09	77.68 (↓ 3.64)	3651	8050	1048	8395	5286 (↑ 5.8%)
Soft Thinking	**95.00**	**76.66**	**95.83**	**64.64**	**83.03** (↑ 1.71)	**3373**	**6620**	**785**	**4722**	**3875** (↓ 22.4%)
	DeepSeek-R1-Distill-Llama-70B [38]									
CoT Thinking	94.70	70.40	94.82	65.34	81.31	3141	8684	620	5500	4486
CoT Thinking (Greedy)	94.61	**73.33**	93.60	66.16	81.92 (↑ 0.61)	**2877**	9457	606	**4443**	4345 (↓ 3.1%)
Soft Thinking	**94.80**	**73.33**	**94.90**	**66.66**	**82.42** (↑ 1.11)	3021	**6644**	**597**	4470	3683 (↓ 17.9%)

Table 1: Comparison of *Soft Thinking* and various baseline methods on accuracy and generation length across mathematical datasets. Best results are highlighted in **bold**.

(1) Pass@1 准确率提升

- 在数学推理任务上，QwQ-32B 模型的平均 Pass@1 从 83.84% (CoT) 提升到 86.32% (Soft Thinking)，**显著提高了 2.48%**。
- 在颇具挑战性的 AIME2024 数据集上，提升更是达到了 6.45%。对于 DeepSeek 模型，Pass@1 也有 1.11% 到 1.71% 不等的提升。

(2) 词语效率显著提高

- 在数学推理基准上，Soft Thinking 使得 QwQ-32B 的词语使用量减少了 11.6%，DeepSeek-Qwen-32B 减少了 22.4%，DeepSeek-Llama-70B 减少了 17.9%，这些都是与标准 CoT Thinking 相比的结果。

4.1.2 编程基准

编程基准包括：HumanEval、MBPP、LiveCodeBench。特别是LiveCodeBench使用了2024年8月至2025年1月发布的近期问题，以确保评估的公平性，避免数据污染。

	Accuracy ↑				Generation Length ↓			
	HumanEval	MBPP	LiveCodeBench	Avg.	HumanEval	MBPP	LiveCodeBench	Avg.
	QwQ-32B [13]							
CoT Thinking	97.63	97.49	62.00	85.70	2557	2154	9986	4899
CoT Thinking (Greedy)	95.73	96.50	57.35	83.19 (↓ 2.51)	**2396**	**2069**	**7034**	**3833** (↓ 21.8%)
Soft Thinking	**98.17**	**97.66**	**62.72**	**86.18** (↑ 0.48)	2638	2157	7535	4110 (↓ 16.1%)
	DeepSeek-R1-Distill-Qwen-32B [38]							
CoT Thinking	97.25	95.13	57.33	83.23	3095	2761	8376	4744
CoT Thinking (Greedy)	87.19	87.54	43.36	72.70 (↓ 10.53)	**2294**	**1703**	**4702**	**2900** (↓ 38.9%)
Soft Thinking	**97.56**	**95.33**	**59.50**	**84.13** (↑ 0.90)	2713	2534	6255	3834 (↓ 19.1%)
	DeepSeek-R1-Distill-Llama-70B [38]							
CoT Thinking	97.71	94.77	56.94	83.14	2711	2386	8319	4472
CoT Thinking (Greedy)	92.07	91.82	48.02	77.30 (↓ 5.84)	2192	1979	5438	3203 (↓ 28.3%)
Soft Thinking	**98.17**	**94.94**	**58.42**	**83.84** (↑ 0.70)	2498	2214	6512	3741 (↓ 16.3%)

Table 2: Comparison of *Soft Thinking* and various baseline methods on accuracy and generation length across two coding datasets. Best results are highlighted in **bold**.

(1) Pass@1 准确率提升

- 在编程基准上，Soft Thinking 也实现了稳定提升，平均 Pass@1 提高了 0.48% 到 0.90%。

(2) 词语效率显著提高

- 在编程任务中也观察到了类似的趋势：词语使用量减少了 16.1% 至 19.1%。这种**大幅减少词语使用量**不仅降低了计算和推理成本，也表明模型能够通过更简洁高效的推理过程达到正确答案。

小结：

- 实验结果表明，Soft Thinking 在**同时提升推理性能（准确率）和词语效率（生成长度）**方面取得了成功。
- 这种“双赢”得益于其核心思想：利用能够编码更丰富语义信息的概念标记。在每一步推理中，概念标记能够在一个“词语”内表示和处理更抽象或复合的概念。因此，模型只需要更少的推理步骤就能达到正确答案，直接体现为更少的词语使用量。
- Soft Thinking 的效率提升**并非简单地依赖于更激进的剪枝或走捷径，而是反映了模型推理过程的根本性增强**。通过在每一步保留所有可能下一步词语的完整概率分布，Soft Thinking 实现了多条推理轨迹的“软性”聚合，有效地**拓宽了模型的探索空间**，同时又避免了显式枚举带来的组合爆炸。

4.2 定性结果

- **示例-1**

Question: 43 * 34 = ?

CoT Thinking (157 tokens):
First, I recognize that multiplying 43 by 34 can be simplified by breaking down the numbers into more manageable parts. I'll decompose 34 into 30 and 4.\n\nNext, I'll multiply 43 by 30. Since 30 is three times 10, I can first multiply 43 by 3 to get 129, and then add a zero to account for the 10, resulting in 1,290.\n\nThen, I'll multiply 43 by 4, which equals 172.\n\nFinally, I'll add the two results together: 1,290 plus 172 gives me the final answer of 1,462.

Soft Thinking (96 tokens):
To solve 43 multiplied by 34, I'll start by breaking down the numbers into their place values. \n\nFirst, I'll multiply 43 by 4, which gives me 172.\n\nNext, I'll multiply 43 by 30, resulting in 1,290.\n\nFinally, I'll add these two products together: 172 plus 1,290 equals 1,462.

Figure 3: A comparison between standard CoT and *Soft Thinking* on a multiplication problem. We select the token with the highest probability at each step of *Soft Thinking* for readability and interpretability. Full distribution is visualized in Figure 4. Red text denotes repetitive, useless words.

图3展示了一个简单的乘法问题（43 * 34）上标准CoT和Soft Thinking的输出对比。研究人员为了可读性，在Soft Thinking的每一步选择了概率最高的词语进行展示。结果发现，**Soft Thinking 的输出保持了高度的可读性和可解释性**。虽然两种方法都得到了正确答案（1462），Soft Thinking 生成的解释显著更简洁（96个词语对比157个词语）。这表明Soft Thinking在提高词语效率的同时，能够保留逻辑结构。

- **示例-2**

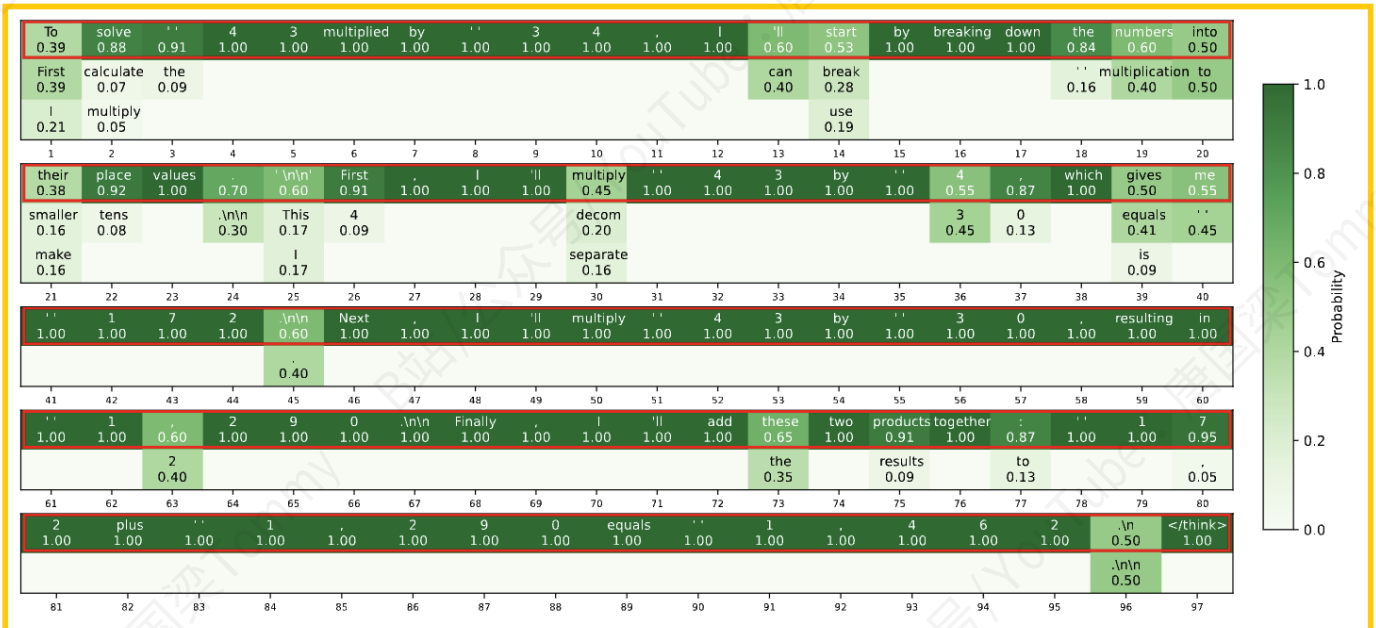


Figure 4: An example illustrating the probability distribution of our proposed *Soft Thinking* method. At each step, top-k token candidates and their probabilities are shown. Red boxes indicate the selected tokens that form the final generated sequence for readability and interpretability.

图4展示了Soft Thinking在中间推理步骤中词语的概率分布。研究人员高亮显示了概率最高的三个词语。在探索性推理阶段（如图4中的步骤1-3, 13-14, 18-20），词语分布看起来更均匀，反映了存在多种可行的推理路径。而在进行精确计算时，词语分布变得接近于独热（one-hot），表明文本元素有助于路径探索，而数值组件处理精确计算。

5. 总结

Soft Thinking vs 传统CoT的区别：

- 传统的链式思维（Chain-of-Thought, CoT）在每一步推理中仅采样一个离散词元（token），因而模型被迫过早做出单一路径选择，容易陷入错误分支。
- Soft Thinking引入了"概念 token"（concept token）的概念，在每个步骤保留整个词汇表的概率分布，不再仅选择一个最高概率词元。这等于在连续的概念空间中进行推理，允许模型同时涵盖多个潜在路径，提高了探索多样推理轨迹的能力。

阶段	CoT（传统）	Soft Thinking
中间推理	每步采样单 token → 早早走到固定路径	输出完整分布 → 概念 token；信息"叠加"
表达能力	受限于离散 token	连续概念空间，语义更细腻
效率控制	可能长链啰嗦	Cold Stop 监测熵，遇低熵快收束
结果	准确 or 误入歧途	同时保持多路径信息，收敛到正确答案且更精简

6. 案例实战

6.1 环境配置与模型下载

```
# 第1步：创建虚拟环境
conda create -n st_venv python=3.11 -y
conda init bash && source /root/.bashrc
conda activate st_venv
conda install ipykernel
ipython kernel install --user --name=st_venv


# 第2步：模型下载

#  （方法-1）从HuggingFace下载
# 设置国内镜像环境变量（可选，解决网络问题）
# export HF_ENDPOINT=https://hf-mirror.com

# 安装 git lfs
apt-get update
apt-get install -y curl vim sudo
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh | sudo
bash
sudo apt-get install git-lfs
```

```
git lfs install

# 这个模型下载路径可以自定义，下面是我的路径。
cd /root/autodl-tmp/
git clone https://huggingface.co/Qwen/Qwen3-1.7B


#  （方法-2）从ModelScope下载
# 参考官网下载指南：https://www.modelscope.cn/models/Qwen/Qwen3-0.6B/files
# 有一点需要注意，下载模型时，需要指定缓存路径，不然就使用默认的路径。
pip install modelscope
modelscope download --model Qwen/Qwen3-0.6B --local_dir /workspace/model/Qwen3-0.6B



# 第3步：下载数据集
pip install --upgrade huggingface_hub
# 如果不能科学上网，请设置下面的endpoint
export HF_ENDPOINT=https://hf-mirror.com
cd /root/autodl-tmp/
huggingface-cli download --repo-type dataset --resume-download HuggingFaceH4/aime_2024 --
local-dir /root/autodl-tmp/aime_2024

# 第4步：安装项目依赖包
pip install torch transformers accelerate jsonlines math_verify openai torch_memory_saver

# 预计编译\安装大概20分钟
pip install flash_attn --no-build-isolation

# install SGlang (0.4.6.post1) tailored for soft thinking
cd sglang_soft_thinking_pkg
pip install -e "python[all]"
cd ..
```

6.2 原创案例演示

参考我实现的代码案例

6.3 SGLang 推理引擎“无缝”支持 Soft Thinking

> 论文在附录 A.3 给出了完整的工程改造方案，下面按 **改动范围** → **关键代码** → **工作机制** 的顺序展开说明。

（1）配置层：让 SGLang 能切换到 *soft-thinking* 模式

文件	新增字段/参数	作用
`model_config.py``server_args.py`	`enable_soft_thinking: bool` `max_topk: int` `think_end_str`	通过命令行 `--enable-soft-thinking --max-topk 15 --think-end-str </think>` 等开关整条推理链

（2）采样器：从“单 token”到“概念 token”

- **修改点**：`sampler.py`
  1. 若开启 *soft-thinking*，执行 `torch.topk(logits, k=max_topk)` 得到 `topk_probs / topk_indices`；概率归一化后一并写入输出对象。
  2. 仍返回一个 `batch_next_token_ids`（通常取 argmax）供 tokenizer 记录历史，与旧逻辑兼容。
  3. 计算熵 `entropy = -Σ p log p` 并随同结果输出，为 Cold Stop 提前准备信号。

效果：一次 forward 就拿到“完整分布 + 熵”，不用再二次遍历或重新 softmax。

（3）嵌入层：把概率分布映射到连续概念空间

- **文件**：`vocab_parallel_embedding.py`
- **新增函数** `weighted_forward(topk_probs, topk_indices)`：

```
topk_embeds = self.embedding(topk_indices)   # [B, K, D]
new_embed   = (topk_probs.unsqueeze(-1) * topk_embeds).sum(dim=1)
```

  仅对 top-k 子集做一次矩阵-向量乘法，O(k·d) 计算量可忽略。
- 多卡场景下还实现 `weighted_forward_tp` 做张量并行聚合。

（4）模型前向：接收“连续嵌入”

- **文件**：`models/llama.py`，`models/qwen2.py`
- **核心分支**

```
if batch.topk_probs is not None:
    hidden_states = self.embed_tokens.weighted_forward(...)
else:
    hidden_states = self.embed_tokens(input_ids)
```

  —— 即只在“思考阶段”走 `weighted_forward`，答案阶段仍走标准 embedding。
- **零权重改动**：只换入口向量，Transformer 主体完全复用。

（5）解码调度：Cold Stop = 早停机制

文件	关键逻辑
`schedule_batch.py``scheduler_output_processor_mixin.py`	维护 `low_entropy_steps` 计数；当 `entropy < τ` 连续 ≥ `length_threshold` 时：`self.output_ids[-1] = think_end_str_id` —— 强制写入 `</think>` 并切换到答案模式

设计要点

- 阈值 `τ`、计数 `k` 均可通过 `sampling_params.py` 配置；
- 终止后 **不会**影响最终答案的温度/采样策略。
