大模型技术及开发

大模型部署

主讲人：陈小军

时间：2025.5.xx

01

大模型解码

2025

解码策略

- 解码：基于输入（提示）生成输出的过程
- 自回归解码：逐词生成下一个词
  - 目前大模型主要采用Transformer 解码器架构

输入： 模型 M，输入词元序列 $\boldsymbol{u}$
输出： 输出词元序列 $\boldsymbol{y}$

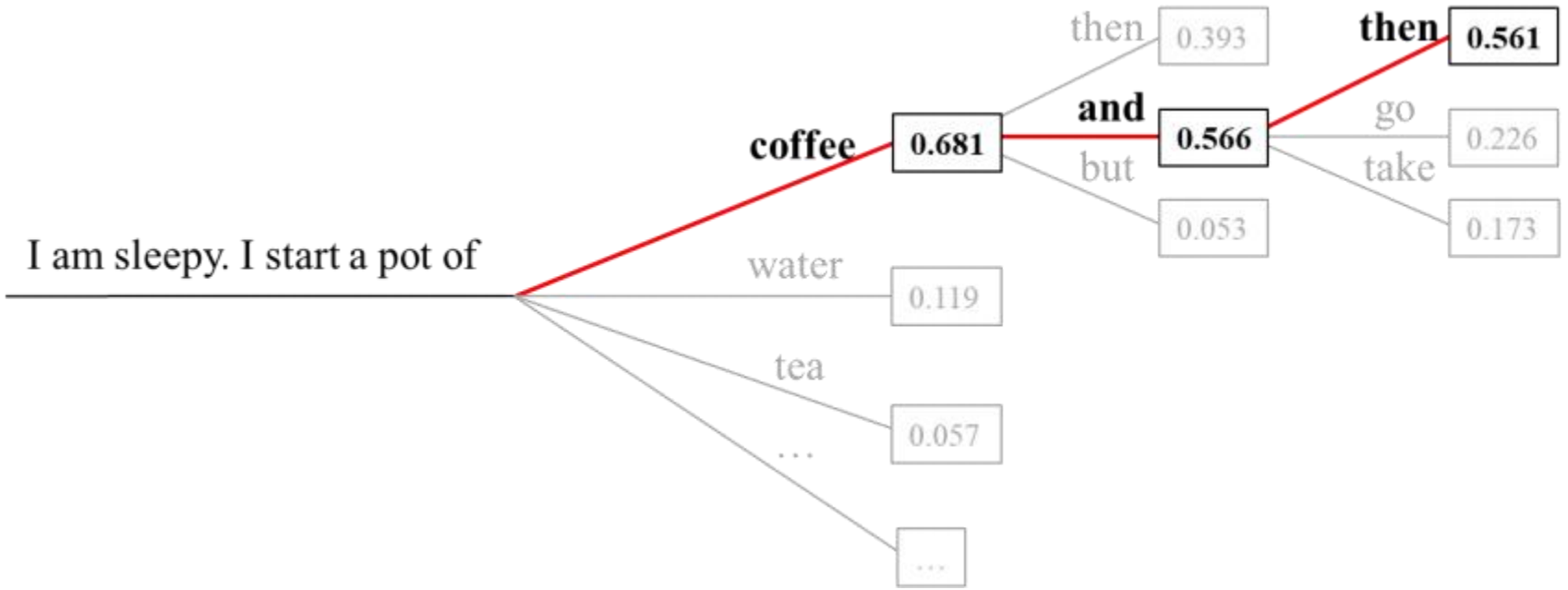
```
repeat
    P = M(u) # 生成下一个词元的概率分布
    u' ~ P # 从分布中采样得到下一个词元
    u ← u ⊕ [u']
until u' 是结束词元或者 u 的长度超过预设长度.
y ← u
```

伪代码：自回归解码流程

解码策略

- 贪心搜索：每个生成步骤中都选择概率最高的词元

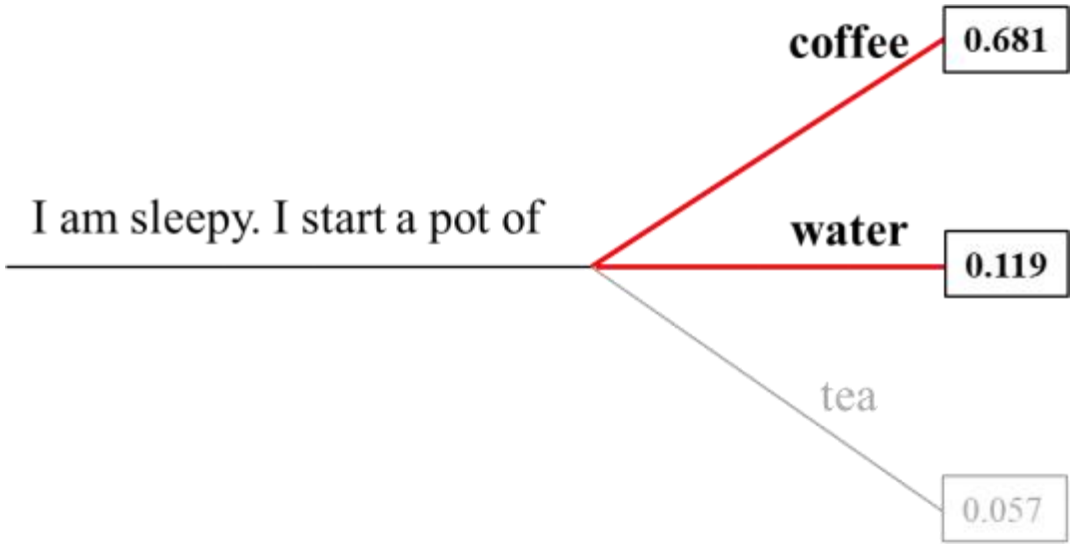
$$u_i = \arg\max_u p(u \mid u_{<i})$$



图中红线路径是贪心搜索最后选择的路径

贪心搜索的改进

- 束搜索：每步保留前n 个具有最高概率的句子
  - 缓解贪心搜索陷入“局部最优”的问题

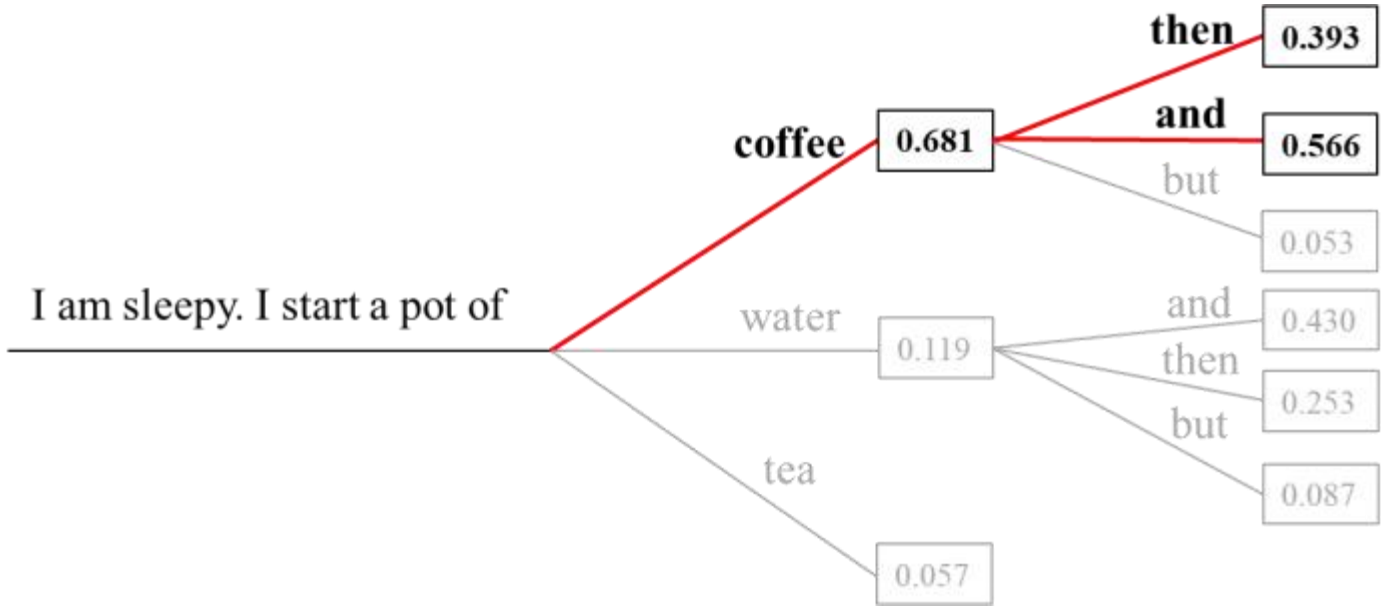


束搜索第一步（n=2）

在第一步解码过程中保留了概率最高的两个单词

贪心搜索的改进

- 束搜索：每步保留前n 个具有最高概率的句子
  - 缓解贪心搜索陷入“局部最优”的问题



束搜索第二步（n=2）

在第二步解码过程中进一步拓展了两个概率最高的单词

解码策略

- 随机采样：基于概率分布采样得到下一个词元
  - 增强结果的多样性

$$U_i \sim P(U \mid u_{<i})$$

I am sleepy. I start a pot of _____

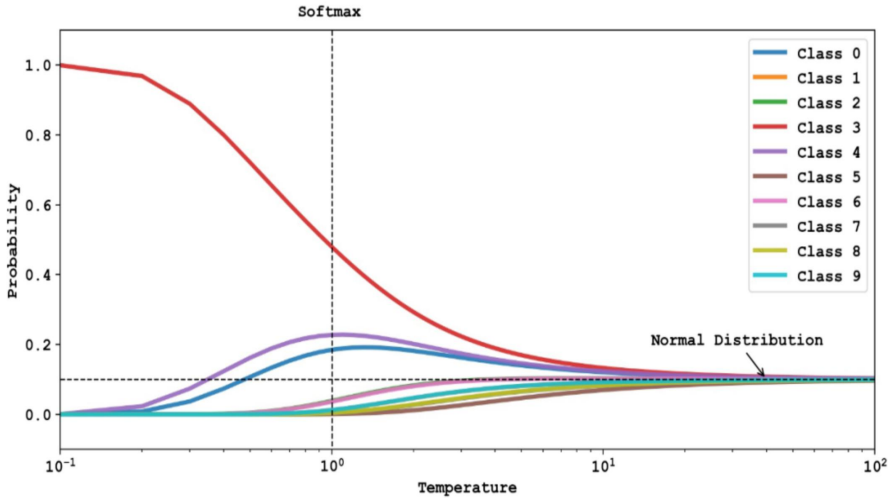
coffee	0.681	strong	0.008	soup	0.005
water	0.119	black	0.008	…	…
tea	0.057	hot	0.007	happy	4.3e-6
rice	0.017	oat	0.006	Boh	4.3e-6
chai	0.012	beans	0.006	…	…

基于文本前缀的下一个词概率分布

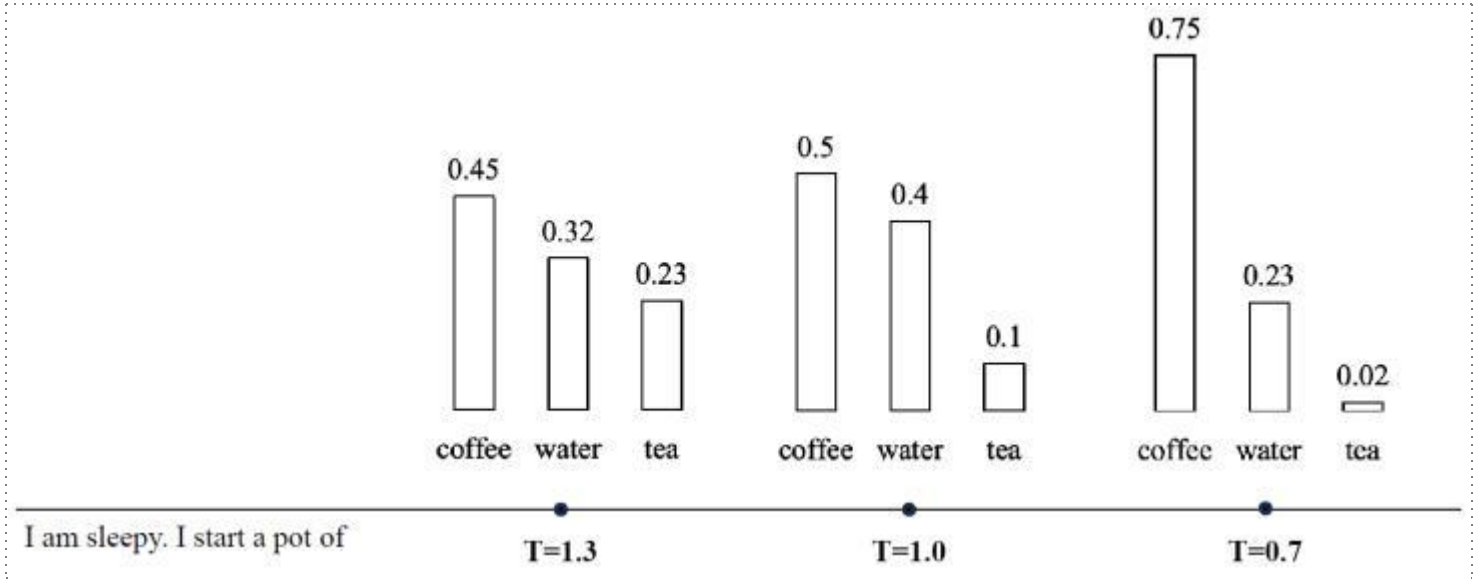
随机采样的改进

- 温度采样：调整softmax 函数的温度系数

$$P(u_j \mid \boldsymbol{u}_{<i}) = \frac{\exp(l_j/t)}{\sum_{j'} \exp(l_{j'}/t)}$$

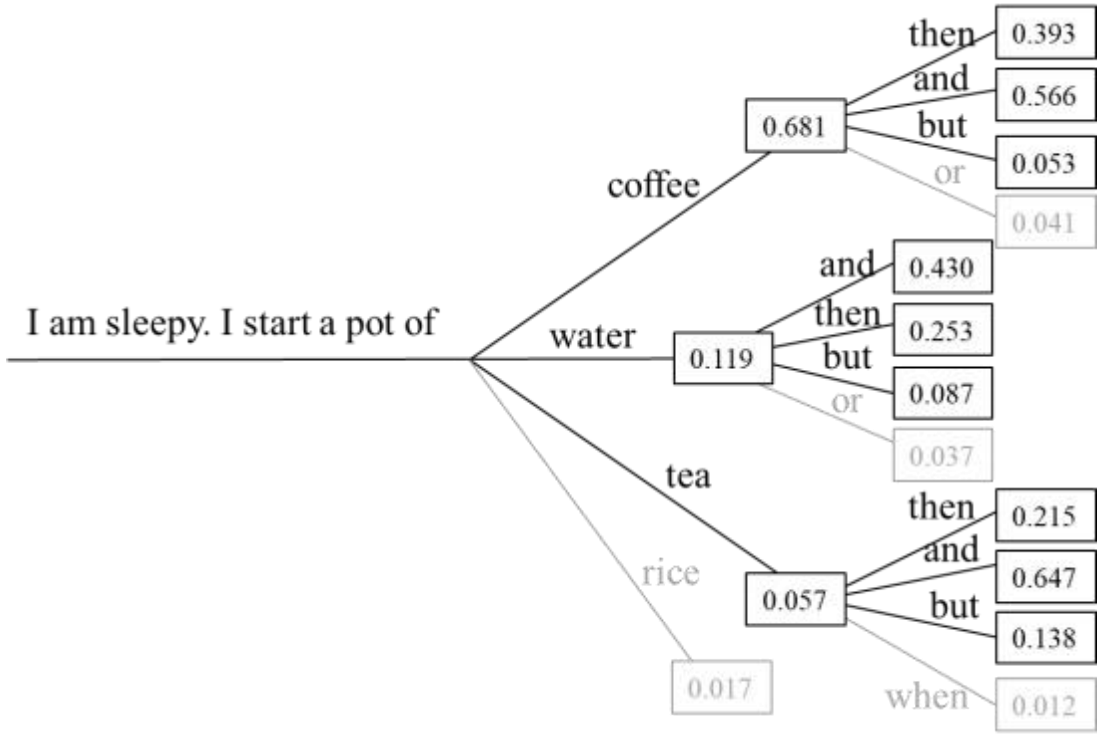


- 降低温度系数，增加高概率词元可能性，降低低概率词元可能性



随机采样的改进

- Top-t 采样：仅从概率前t 高的词元中采样
  - 直接剔除低概率的词，兼顾生成质量

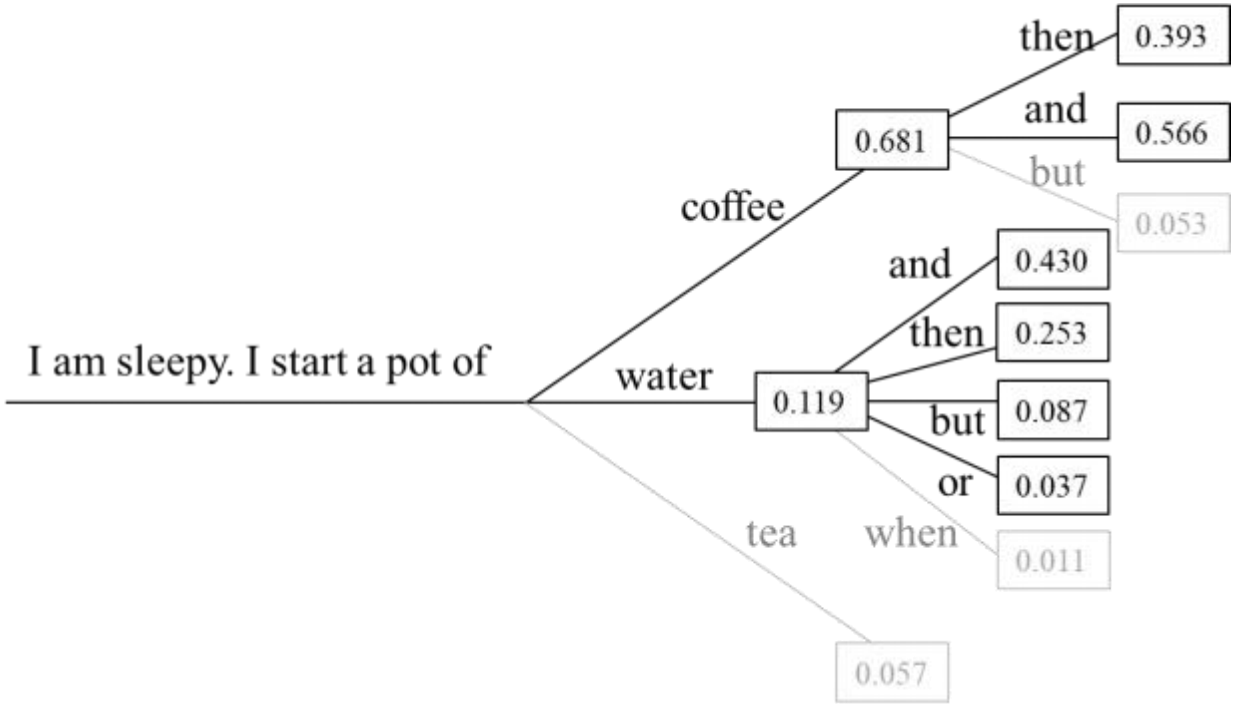


top-t 采样（k=3）

Hierarchical Neural Story Generation, **ACL 2018**

随机采样的改进

- Top-# 采样：仅从累积概率和为# 的高概率词元中采样
  - 考虑整体概率分布，适应不同场景



top-p 采样（p=0.8）

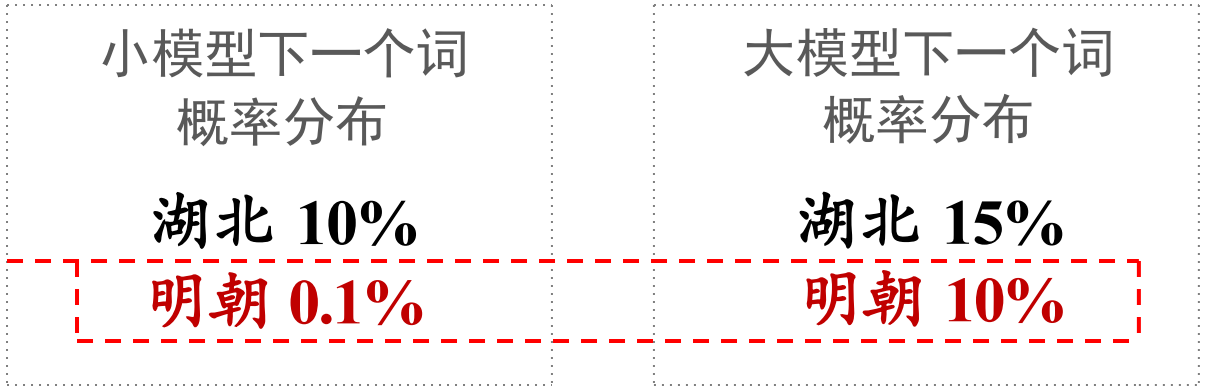
随机采样的改进

- 重复惩罚：缓解生成重复文本
  - n-元惩罚
    - 直接避免生成重复的连续 n 个词元（通常$n \in \{3,4,5\}$）
  - 出现惩罚
    - 词元 t 生成的概率 $P(t) = \text{logit}(t) - \text{是否出现}(t) \times \alpha$
  - 频率惩罚
    - 词元 t 生成的概率 $P(t) = \text{logit}(t) - \text{出现次数}(t) \times \alpha$

随机采样的改进

- 对比解码
  - 在大模型和小模型的概率分布差值中采样
- 大模型会为更重要的单词分配更高的概率
  - 借助两者之间的差异发掘重要词汇

李时珍是湖北人，他出生于__



生成“明朝”概率大幅增加

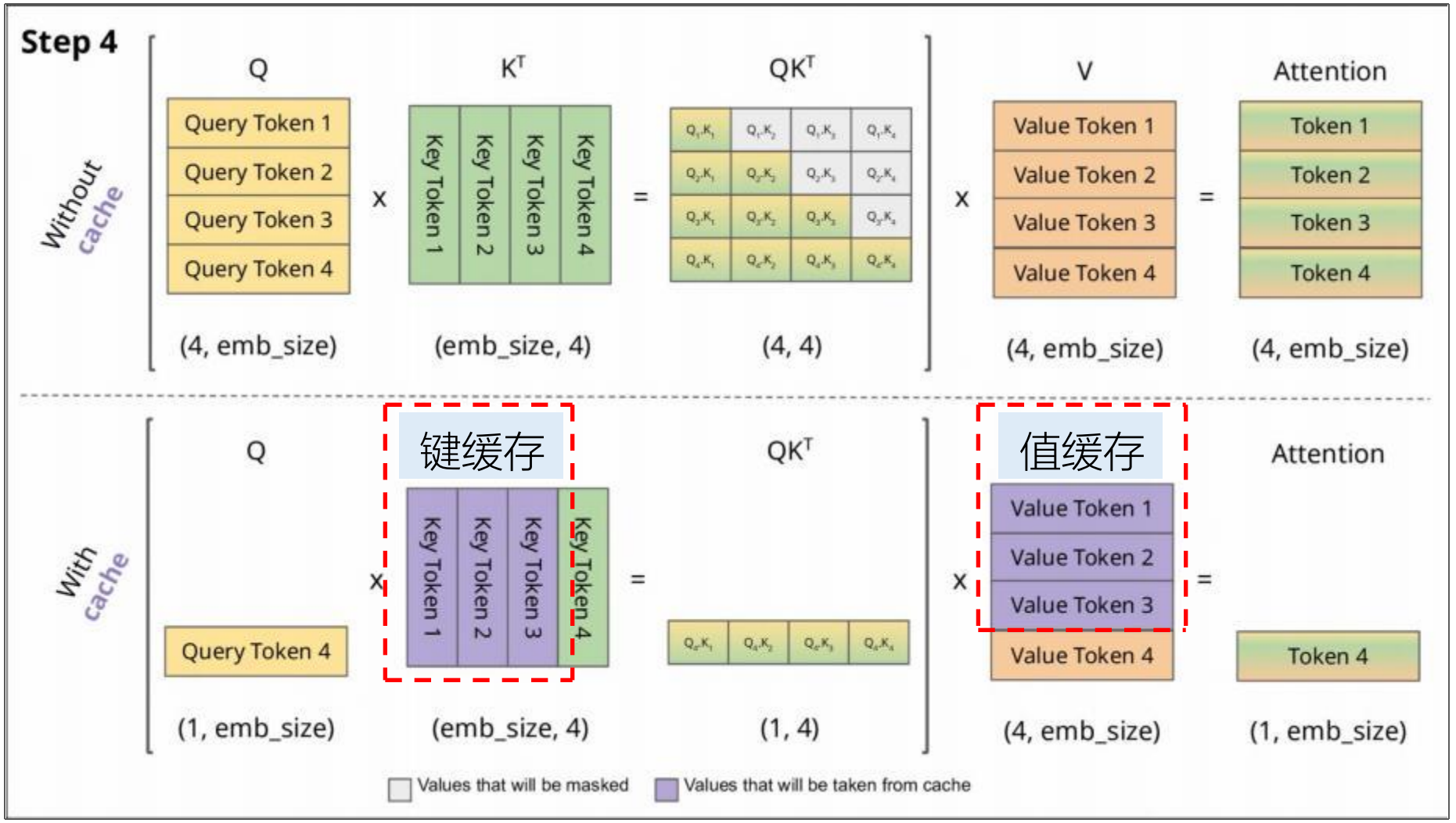
Contrastive Decoding: Open-ended Text Generation as Optimization, **ACL 2023**

02

解码效率分析与加速算法

2025

解码与键值缓存



每生成一个词，重新计算所有查询、键、值矩阵

新生成词的查询向量与之前词缓存的键值计算注意力

https://medium.com/@joaolages/kv-caching-explained-276520203249

解码与键值缓存

- 基于键值缓存优化的贪心解码算法示意图

输入：模型 $\mathcal{M}$，输入词序列 $\boldsymbol{u}$
输出：输出词序列 $\boldsymbol{y}$

1: $P, \boldsymbol{K}_{past}, \boldsymbol{V}_{past} = \mathcal{M}(\boldsymbol{u})$
2: $\hat{u} = \arg\max P$
3: $\boldsymbol{u} \leftarrow \boldsymbol{u} \oplus [\hat{u}]$

全量解码

$$\text{Attention}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{q}\boldsymbol{K}^\top}{\sqrt{D}})\boldsymbol{V}$$

4: **while** $\hat{u}$ 不是结束词且 $\boldsymbol{u}$ 的长度不超过预设长度 **do**
5: $P, \boldsymbol{k}, \boldsymbol{v} = \mathcal{M}(\hat{u}, \boldsymbol{K}_{past}, \boldsymbol{V}_{past})$ # 利用键值缓存计算新生成词状态
6: $\hat{u} = \arg\max P$
7: $\boldsymbol{u} \leftarrow \boldsymbol{u} \oplus [\hat{u}]$
8: $\boldsymbol{K}_{past}, \boldsymbol{V}_{past} \leftarrow \boldsymbol{K}_{past} \oplus \boldsymbol{k}, \boldsymbol{V}_{past} \oplus \boldsymbol{v}$ # 更新键值缓存
9: **end while**

解码加速的两个关键

增量解码

10: $\boldsymbol{y} \leftarrow \boldsymbol{u}$

解码效率的定量评估指标

GPU 评估指标

算力：每秒的浮点运算次数，FLOP/s

带宽：每秒的显存读写量，byte/s

计算强度上限：算力和带宽的比值

模型评估指标

运算量：所需总浮点运算数，FLOP

访存量：所需总显存读写量，byte

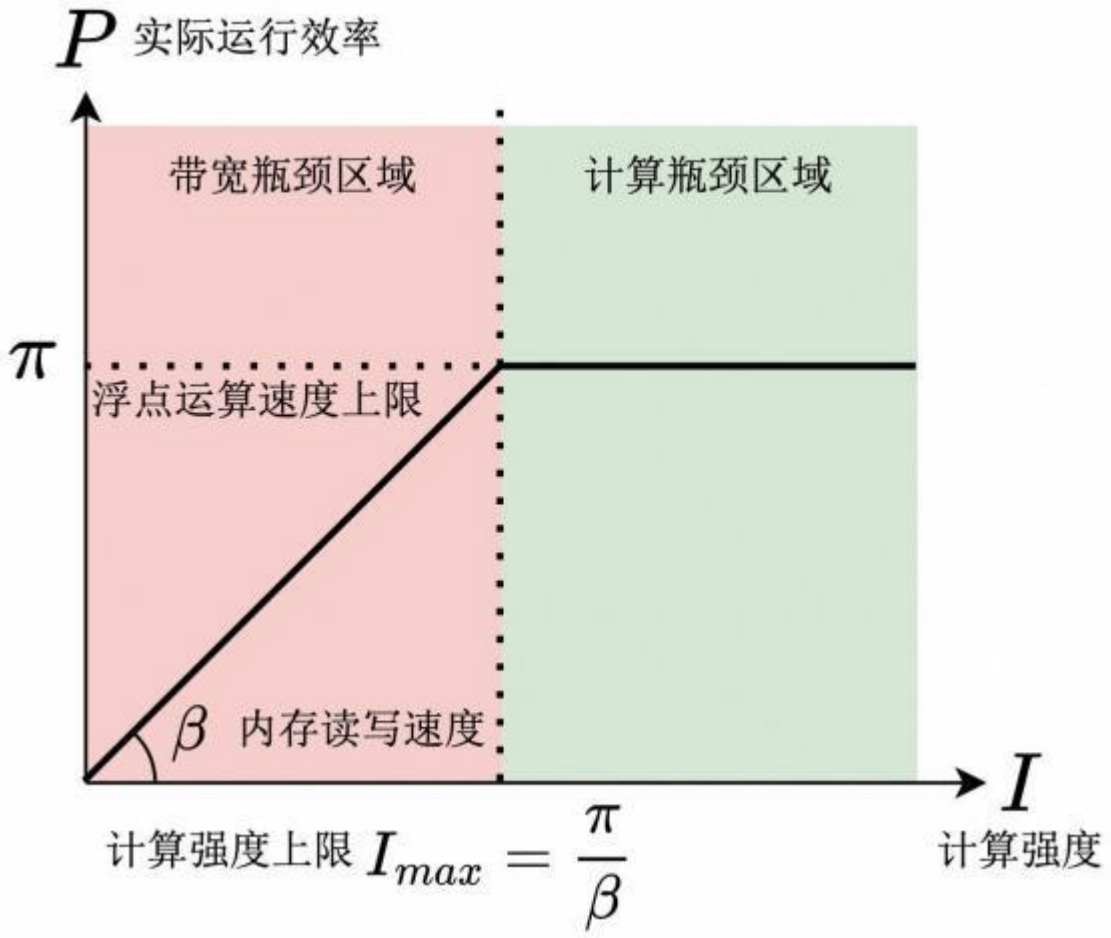
计算强度：运算量和访存量的比值

A100 (80G)：算力为312 TFLOP/s，带宽为2039 GB/s，计算强度上限约为142.51 FLOP/byte

https://www.nvidia.com/en-us/data-center/a100/

解码效率的定量评估指标

- 效率问题分析
  - 带宽瓶颈
    - 模型计算强度< GPU 计算强度上限
    - 运行效率主要受显存读写速度的影响
  - 计算瓶颈
    - 模型计算强度> GPU 计算强度上限
    - 运行效率主要受运算速度的影响



运算量、访存量和计算强度估计

- 矩阵乘法的运算量
  - 矩阵$A \in \mathbb{R}^{n\times m}$和矩阵$B \in \mathbb{R}^{m\times p}$相乘所需的运算量为$2nmp$
- 矩阵乘法的访存量
  - 矩阵$A \in \mathbb{R}^{n\times m}$和矩阵$B \in \mathbb{R}^{m\times p}$相乘所需的访存量为$O(nm + mp + np)$
- 矩阵乘法的计算强度
  - 运算量比访存量为$O\left(\frac{1}{\frac{1}{n}+\frac{1}{m}+\frac{1}{p}}\right)$

运算量、访存量和计算强度估计

- 张量 $X \in \mathbb{R}^{B\times T\times H}$ 与矩阵 $B \in \mathbb{R}^{H\times H}$ 相乘
  - 运算量：$2BTH^2$
  - 访存量：$O(BTH + H^2)$
  - 计算强度：$O(\frac{1}{\frac{1}{H}+\frac{1}{BT}})$
- 注意力计算 $\text{softmax}\left(\frac{QK^T}{D}\right)V$，$Q, K, V \in \mathbb{R}^{B\times T\times N\times D}$
  - 运算量：两次矩阵乘法各 $2BT^2ND$，softmax运算和 $\sqrt{D}$ 放缩共 $4BT^2N$
  - 访存量：两次矩阵乘法 $O(BTND + BT^2N)$，其他运算 $O(BT^2N)$
  - 计算强度：$O(\frac{1+\frac{1}{D}}{\frac{1}{D}+\frac{1}{T}})$

批次大小B，序列长度T，隐含层维度H，注意力头数N，每个头维度D

运算量、访存量和计算强度估计

- 全量解码阶段的计算强度（推导见教材）
  - 线性变换强度约为2730.7
    - 公式①④⑥⑧
  - 多头注意力强度约为114.7
    - 公式③
  - 其余操作强度约为1
    - 公式②⑤⑦⑨

A100 (80G) 的计算强度上限为142.5

全量解码是**计算瓶颈**的

计算公式	计算强度
① $\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}=\boldsymbol{X}\boldsymbol{W}^{Q,K,V}$	$O\left(\frac{1}{\frac{1}{H}+\frac{1}{BT}}\right)$
② $\boldsymbol{Q},\boldsymbol{K}=\mathrm{RoPE}(\boldsymbol{Q},\boldsymbol{K})$	$O(1)$
③ $\boldsymbol{O}=\mathrm{Attn}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})$	$O\left(\frac{1+\frac{1}{D}}{\frac{1}{D}+\frac{1}{T}}\right)$
④ $\boldsymbol{X}=\boldsymbol{O}\boldsymbol{W}^{O}$	$O\left(\frac{1}{\frac{1}{H}+\frac{1}{BT}}\right)$
⑤ $\boldsymbol{X}=\mathrm{Add\&Norm}(\boldsymbol{X})$	$O\left(\frac{1}{1+\frac{1}{BT}}\right)$
⑥ $\boldsymbol{G},\boldsymbol{U}=\boldsymbol{X}[\boldsymbol{W}^{G},\boldsymbol{W}^{U}]$	$O\left(\frac{1}{\frac{1}{H}+\frac{1}{H'}+\frac{1}{BT}}\right)$
⑦ $\boldsymbol{D}=\mathrm{Swish}(\boldsymbol{G})\cdot\boldsymbol{U}$	$O(1)$
⑧ $\boldsymbol{X}=\boldsymbol{D}\boldsymbol{W}^{D}$	$O\left(\frac{1}{\frac{1}{H}+\frac{1}{H'}+\frac{1}{BT}}\right)$
⑨ $\boldsymbol{X}=\mathrm{Add\&Norm}(\boldsymbol{X})$	$O\left(\frac{1}{1+\frac{1}{BT}}\right)$

$B=8$　$T=1024$　$H=4096$　$D=128$

运算量、访存量和计算强度估计

- 增量解码阶段的计算强度（推导见教材）
  - 将全量解码公式（注意力除外）中T 变为1
  - 线性变换的计算强度约为8.0
    - 公式①⑤⑦⑨
  - 多头注意力的计算强度约为1.0
    - 公式④

计算公式	计算强度
① $\boldsymbol{q},\boldsymbol{k},\boldsymbol{v}=\boldsymbol{x}\boldsymbol{W}^{QKV}$	$O\left(\frac{1}{\frac{1}{H}+\frac{1}{B}}\right)$
② $\boldsymbol{q},\boldsymbol{k}=\mathrm{RoPE}(\boldsymbol{q},\boldsymbol{k})$	$O(1)$
③ $\boldsymbol{K},\boldsymbol{V}=\mathrm{Cache}(\boldsymbol{k},\boldsymbol{v})$	-
④ $\boldsymbol{o}=\mathrm{Attn}(\boldsymbol{q},\boldsymbol{K},\boldsymbol{V})$	$O\left(\frac{1+\frac{1}{D}}{1+\frac{1}{D}+\frac{1}{T}}\right)$
⑤ $\boldsymbol{x}=\boldsymbol{o}\boldsymbol{W}^{O}$	$O\left(\frac{1}{\frac{1}{H}+\frac{1}{B}}\right)$
⑥ $\boldsymbol{x}=\mathrm{Add\&Norm}(\boldsymbol{x})$	$O\left(\frac{1}{1+\frac{1}{B}}\right)$
⑦ $\boldsymbol{g},\boldsymbol{u}=\boldsymbol{x}[\boldsymbol{W}^{G},\boldsymbol{W}^{U}]$	$O\left(\frac{1}{\frac{1}{H}+\frac{1}{H'}+\frac{1}{B}}\right)$
⑧ $\boldsymbol{d}=\mathrm{Swish}(\boldsymbol{g})\cdot\boldsymbol{u}$	$O(1)$
⑨ $\boldsymbol{x}=\boldsymbol{d}\boldsymbol{W}^{D}$	$O\left(\frac{1}{\frac{1}{H}+\frac{1}{H'}+\frac{1}{B}}\right)$
⑩ $\boldsymbol{x}=\mathrm{Add\&Norm}(\boldsymbol{x})$	$O\left(\frac{1}{1+\frac{1}{B}}\right)$

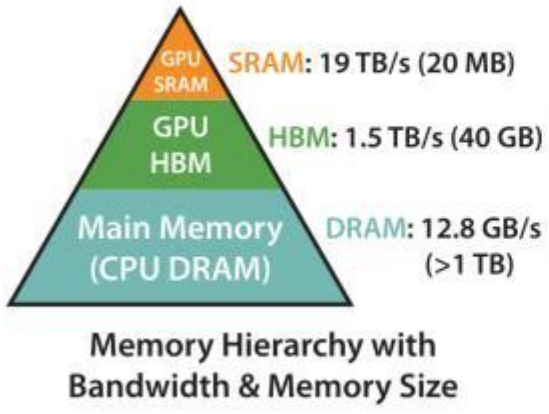
$B=8$　$T=1024$　$H=4096$　$D=128$

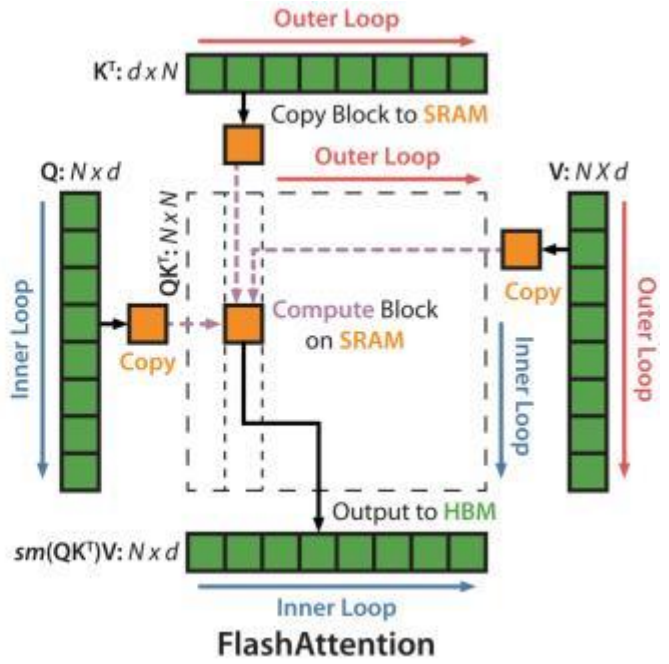
A100 (80G) 的计算强度上限为142.5

增量解码是带宽瓶颈（“内存墙”问题）

系统级优化

- FlashAttention
  - 改进注意力计算softmax $\left(\frac{QK^{\top}}{D}\right)$ V
  - 通过矩阵分块和算子融合，减少中间结果读写，减少访存量



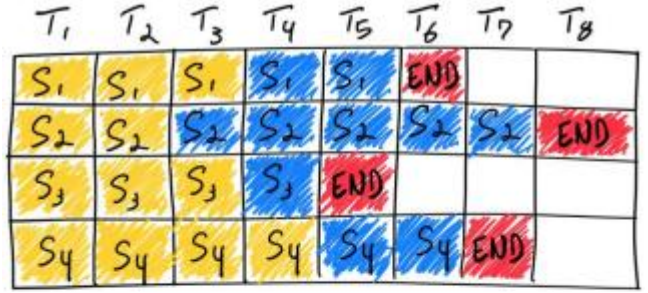


将QKV 分块计算，在SRAM 中直接计算得到最终结果

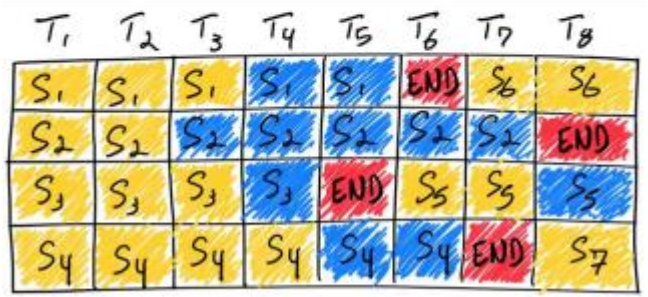
FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. **ICLR 2024**

系统级优化

- 传统批次推理：一个批次全部推理完成才进行下一个
- 批次管理优化：将每个请求进行分割，提升实际运行批次
  - 连续批处理
    - 分割为一个全量解码和若干个单步增量解码
    - 启发式选择部分请求全量解码或单步增量解码
  - 动态分割
    - 将全量解码进一步拆分
    - 同时进行全量解码和增量解码



传统批次推理

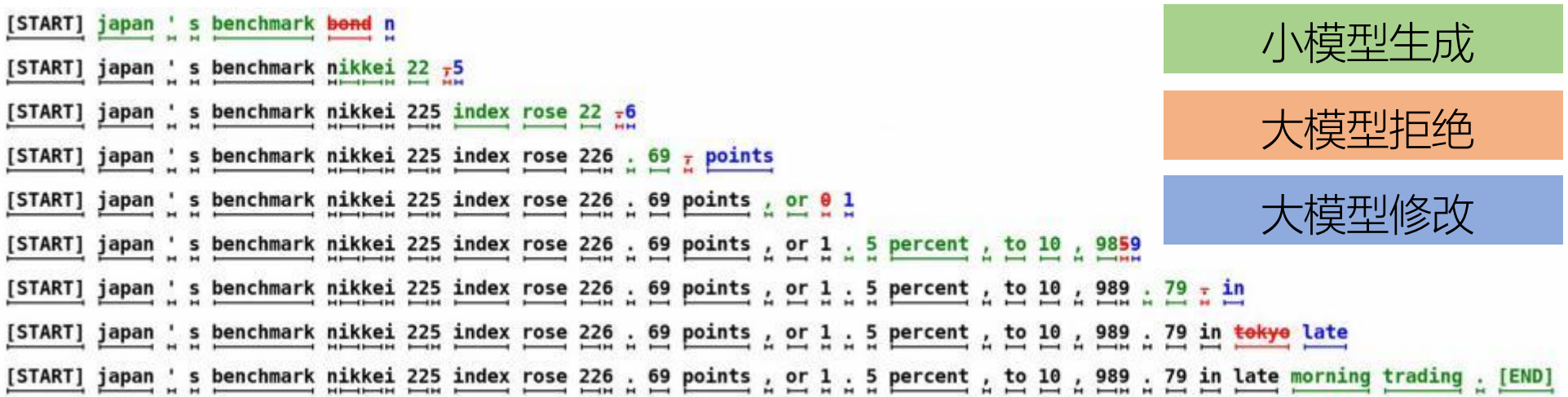


批次管理优化

https://www.anyscale.com/blog/continuous-batching-llm-inference

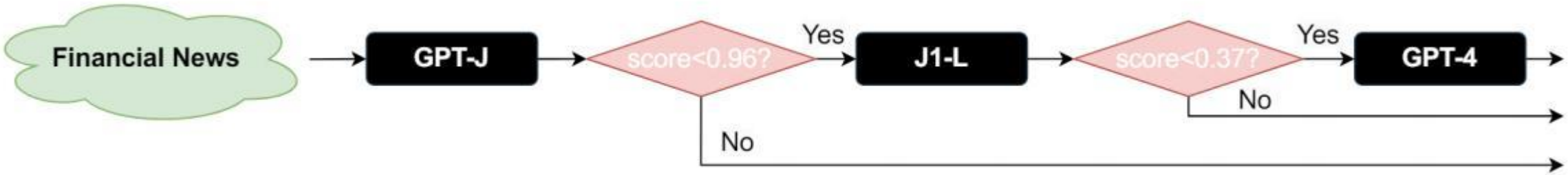
解码策略优化

- 推测解码
  - 先用小且高效的模型自回归地生成3 ~ 5 个词元
  - 再由大模型对这个片段进行一次验证，进行拒绝与修改
  - 不会降低大模型解码质量，一般带来两倍左右加速



解码策略优化

- 级联解码
  - 引入一系列模型，按照效率从高到低排序
  - 依次让模型生成答案，由二分类器判断
  - 如果结果可靠则不需要后续生成

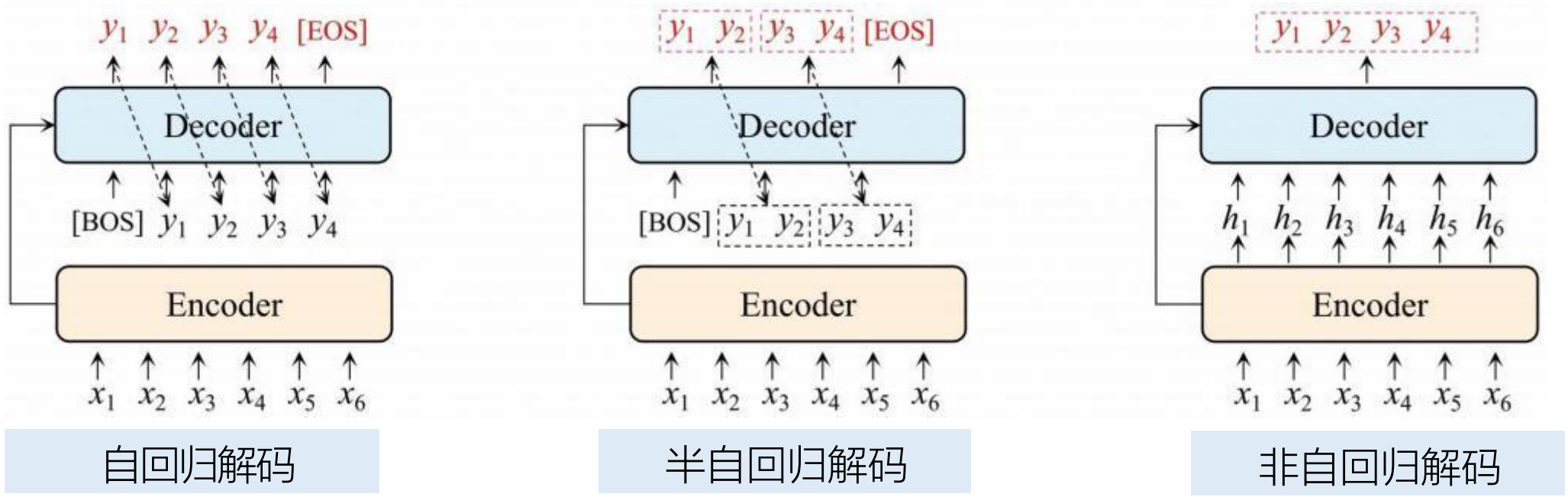


通过优先让相对较小模型进行解码，减少更大模型的调用开销

FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. **arXiv 2023**

解码策略优化

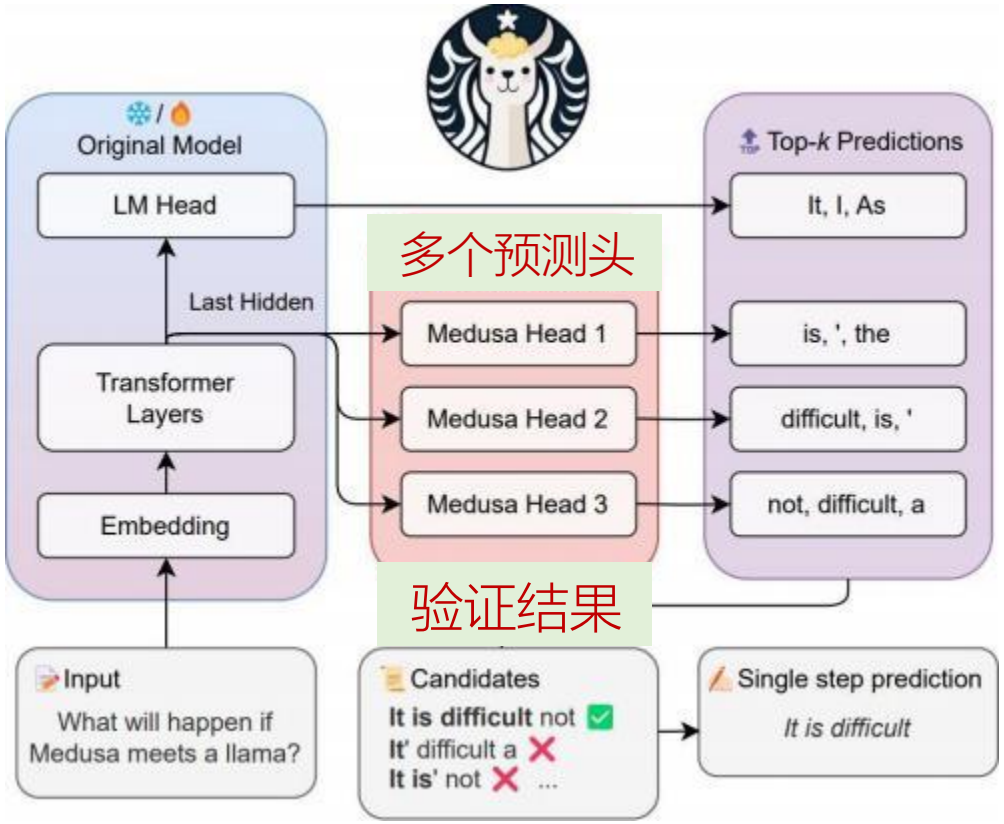
- 非（半）自回归解码
  - 非自回归解码：基于输入一次性生成所有词元
  - 半自回归解码：组内非自回归生成，组间自回归生成



A Survey on Non-Autoregressive Generation for Neural Machine Translation and Beyond. **PAMI 2023**

解码策略优化

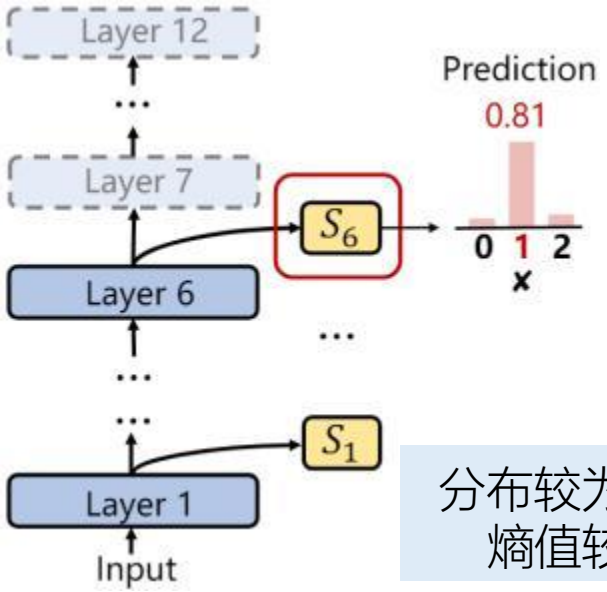
- 非（半）自回归解码
  - 非自回归解码：基于输入一次性生成
  - 半自回归解码：组内非自回归，组间自回归
- Medusa
  - 额外训练两个预测头分别预测第二个词和第三个词
  - 结合推测解码，加速原始大模型生成
  - 不影响生成质量，推理加速2.2 倍



Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. **ICML 2024**

解码策略优化

- 早退机制
  - 不需要所有层计算，满足条件跳过后续层计算
  - 每层得到输出概率分布，计算分布的熵值；如果熵值较低，则提前退出



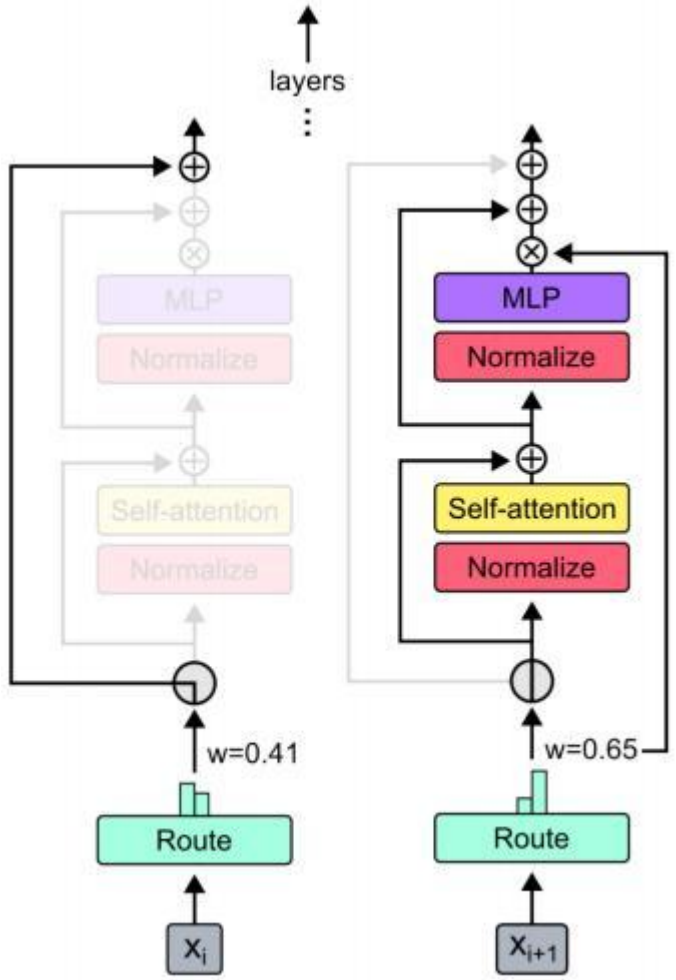
分布较为确定
熵值较低

Model	MNLI-m Acc	MNLI-m Spd-up	MNLI-mm Acc	MNLI-mm Spd-up	QQP F1/Acc	QQP Spd-up	QNLI Acc	QNLI Spd-up	SST-2 Acc	SST-2 Spd-up	MRPC F1/Acc	MRPC Spd-up	RTE Acc	RTE Spd-up	Macro
BERT															
BERT-base (Devlin et al., 2019)	84.6	1.00×	83.4	1.00×	71.2/ -	1.00×	90.5	1.00×	93.5	1.00×	88.9/ -	1.00×	66.4	1.00×	-
BERT-6L	80.8	2.00×	79.9	2.00×	69.7/88.3	2.00×	86.7	2.00×	91.0	2.00×	85.1/78.6	2.00×	63.9	2.00×	80.5
DeeBERT (Xin et al., 2020)	-		-	-	69.4/ -	1.96×	87.9	1.79×	91.5	1.89×	85.2/ -	1.79×	-	-	-
DeeBERT	74.4	1.87×	73.1	1.88×	70.4/88.8	2.13×	85.6	2.09×	90.2	2.00×	84.4/77.4	2.07×	64.3	1.95×	74.7
PABEE	79.8	2.07×	78.7	2.08×	70.4/88.6	2.09×	88.0	1.87×	89.3	1.95×	84.4/77.4	2.01×	64.0	1.81×	80.0
Ours	**83.3**	1.96×	**82.7**	1.96×	**71.2/89.4**	2.18×	**89.8**	1.97×	**92.8**	2.02×	**87.0/81.8**	1.98×	**64.5**	2.04×	**82.5**
RoBERTa															
RoBERTa-base (Xin et al., 2020)	87.0	1.00×	86.3	1.00×	71.8/ -	1.00×	92.4	1.00×	94.3	1.00×	90.4/ -	1.00×	67.5	1.00×	-
RoBERTa-6L	84.4	2.00×	83.4	2.00×	71.6/89.2	2.00×	90.4	2.00×	93.5	2.00×	**89.3/85.5**	2.00×	58.0	2.00×	82.5
DeeBERT	64.2	1.87×	64.7	1.87×	**72.0/89.3**	2.05×	83.8	2.01×	86.9	2.02×	88.7/84.3	1.86×	**60.8**	1.90×	75.4
Ours	**86.6**	1.92×	**86.2**	1.93×	**72.0/89.3**	2.54×	**91.7**	2.11×	**94.5**	1.98×	**89.3/85.5**	1.95×	58.0	2.11×	**83.6**
ALBERT															
ALBERT-base	85.2	1.00×	84.7	1.00×	70.5/88.7	1.00×	92.0	1.00×	93.3	1.00×	89.0/84.8	1.00×	72.0	1.00×	84.8
ALBERT-6L	82.4	2.00×	81.7	2.00×	69.8/88.3	2.00×	90.0	2.00×	91.8	2.00×	87.0/82.4	2.00×	65.8	2.00×	82.2
PABEE	84.2	1.90×	83.5	1.81×	**70.7/88.9**	2.11×	90.9	1.98×	92.4	1.80×	87.6/82.6	1.91×	66.8	2.06×	83.2
Ours	**84.8**	1.94×	**84.1**	1.95×	70.4/88.6	2.35×	**91.9**	1.97×	**92.8**	2.13×	**88.3/84.6**	1.95×	**72.0**	1.93×	**84.5**

AGlobal Past-Future Early Exit Method for Accelerating Inference of Pre-trained Language Models, **NAACL 2021**

解码策略优化

- 早退机制
  - 不需要所有层计算，满足条件跳过后续层计算
  - 每层得到输出概率分布，计算分布的熵值
  - 如果熵值较低，则提前退出
  - 混合深度方法（借鉴MoE）
    - 每层根据路由网络判定是否进行该层计算
    - 可以平衡时间开销，最多减少50% 计算开销



Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. **arXiv 2024**

03

模型压缩

2025

量化基础知识

- 量化：将映射浮点数到整数的过程

$$XQ = R(\boldsymbol{X}/S) - Z$$

R：取整函数
S：放缩因子
Z：零点因子

- 反量化：从量化值中恢复原始值

$$\tilde{X} = S \bullet (xq + Z)$$

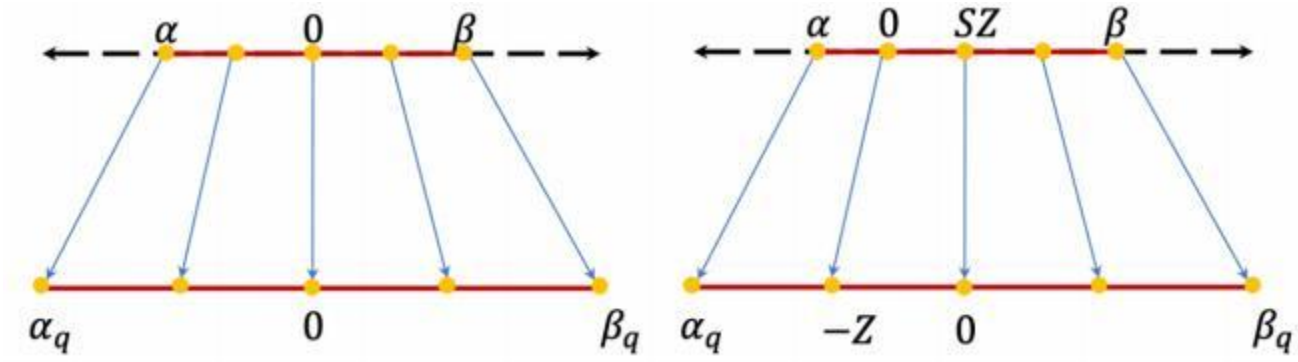
- 量化误差：原始值X 和恢复值 $\tilde{\mathbf{X}}$ 之间的数值差异

$$\Delta = \|\boldsymbol{X} - \tilde{\boldsymbol{X}}\|_2^2$$

量化的目标是最小化量化误差

量化基础知识

- 对称量化和非对称量化
  - 根据零点因子Z 是否为零



对称量化（Z = 0 ）　　非对称量化（Z ≠ 0）

  - 量化粒度的选择

-1	0
0	-2
-1	2

W_{F16}

2

s_W

张量量化：一个矩阵
定义一组量化参数

1	2	s_W
-1	0	
0	-2	
-1	2	

W_{F16}

通道量化：一个矩阵对列维度
设置特定的量化参数

量化基础知识

- 非对称量化方法计算示例

$$X = \quad 1.2,2.4,3.6\ ,\ \ 11.2, 12.4, 13.6$$

确定输入范围 $\in\ [1.2, 13.6]$

$$S \cdot\ 127 + Z\ = 13.6$$

$$S \cdot\ -128 + Z\ = 1.2$$

量化到整数范围 $[-128, 127]$ 两个边界值映射

$$\Rightarrow S = 0.0486, \qquad Z = -152$$

计算量化参数 S 和 Z

$$X_q = \ [-127, -103, -78], [78,103,127]$$

计算量化后结果 X_q

结果：$[[1.22, 2.38, 3.60\ ], [11.18, 12.40, 13.58]]$

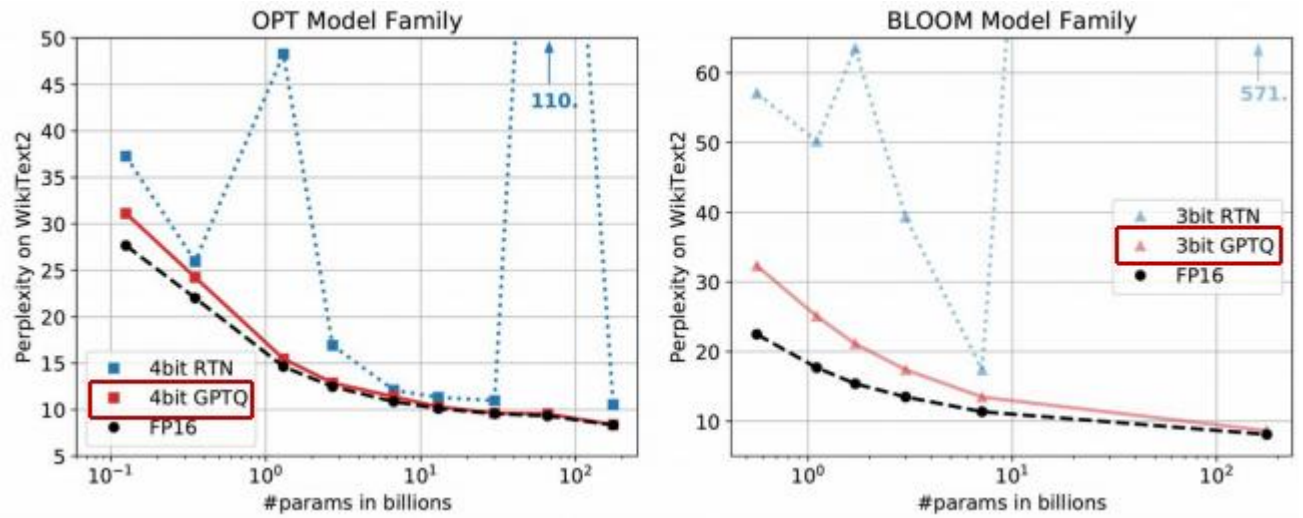
计算反量化后结果

训练后量化方法

- 权重量化（w 是原始权重）
  - 最小化重构损失（w_q 是量化后权重）

$$\arg\min_{w_q} \| Xw - Xw_q \|_2^2$$

  - GPTQ：将权重矩阵按照列维度分组，逐组量化



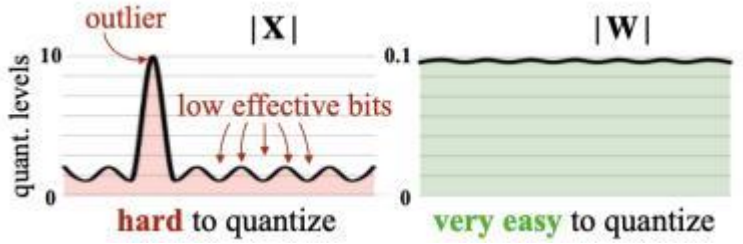
3/4 比特量化与16比特对比

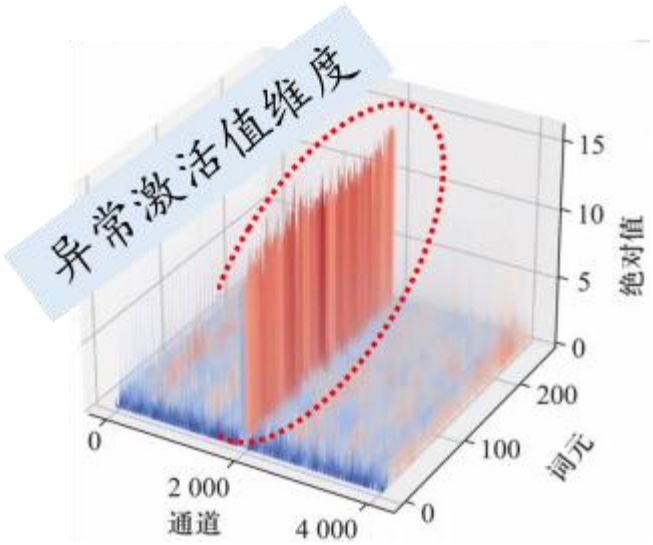
- 模型越大，表现越接近

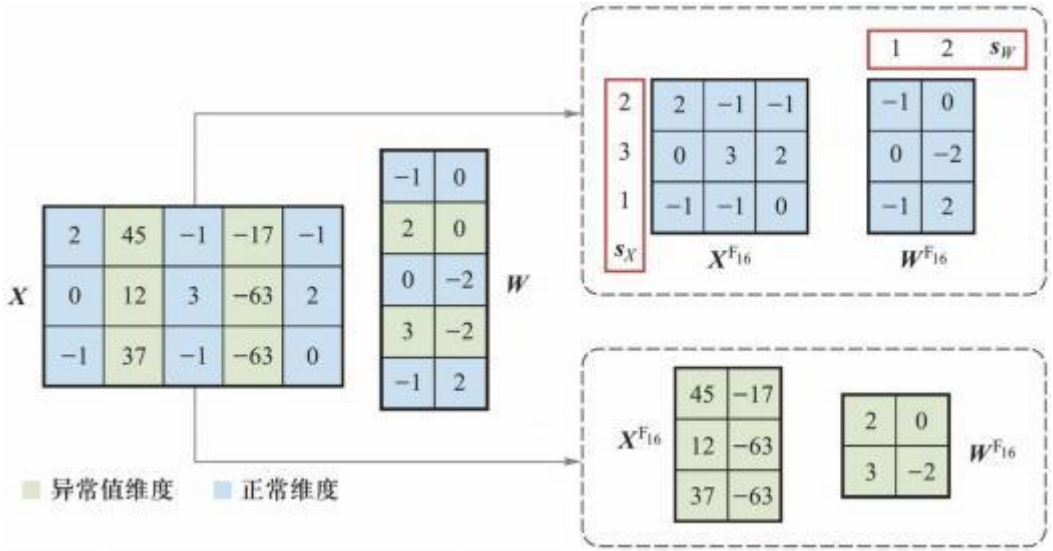
GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers, **ICLR 2023**

训练后量化方法

- 权重和激活值量化
  - 模型达到一定规模，某些维度会出现异常激活值
  - 混合精度分解：将异常值（16比特）和正常值（8比特）分开计算







正常值使用8比特量化

异常值使用16比特计算

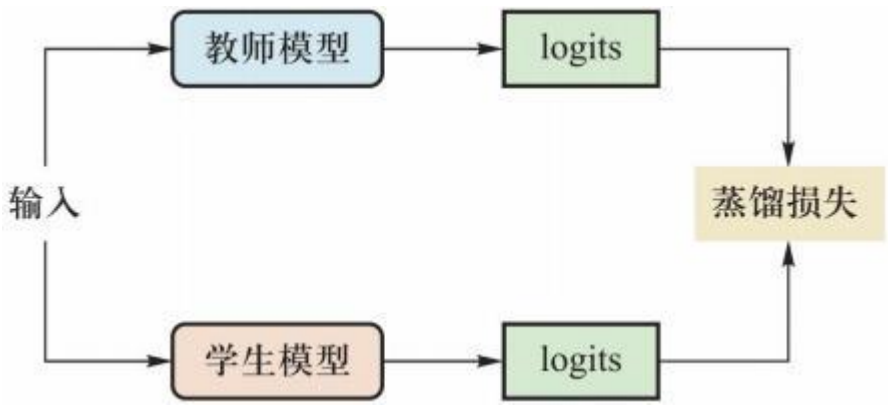
LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale, **NeurIPS 2022**

模型蒸馏

- 模型蒸馏
  - 将复杂模型的知识迁移到简单模型上
- 基于反馈的知识蒸馏
  - 使用教师模型的输出概率分布作为学生模型的“软标签”

$$\mathcal{L}(l_t, l_S) = \mathcal{L}_R(P_t(\cdot), P_S(\cdot))$$

$\mathcal{L}_R$是损失函数
常用KL散度



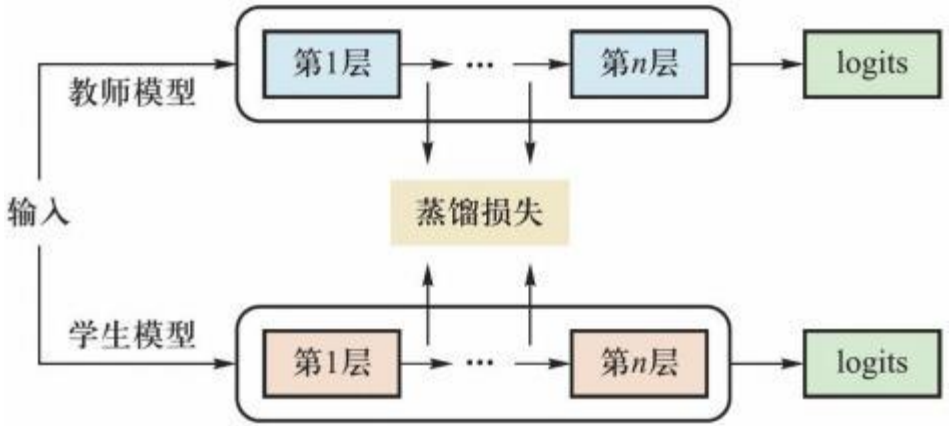
让学生模型的输出
与教师模型接近

模型蒸馏

- 模型蒸馏
  - 将复杂模型的知识迁移到简单模型上
- 基于特征的知识蒸馏
  - 使用教师模型中间层的输出特征作为监督信息训练学生模型

$$\mathcal{L}(f_t(x), f_s(x)) = \mathcal{L}_F(\Phi(f_t(x)), \Phi(f_s(x)))$$

$\Phi(.)$用于转换输出维度



相较于输出层，中间层能提供更丰富的信息

2025

谢谢大家

时间：202X.X