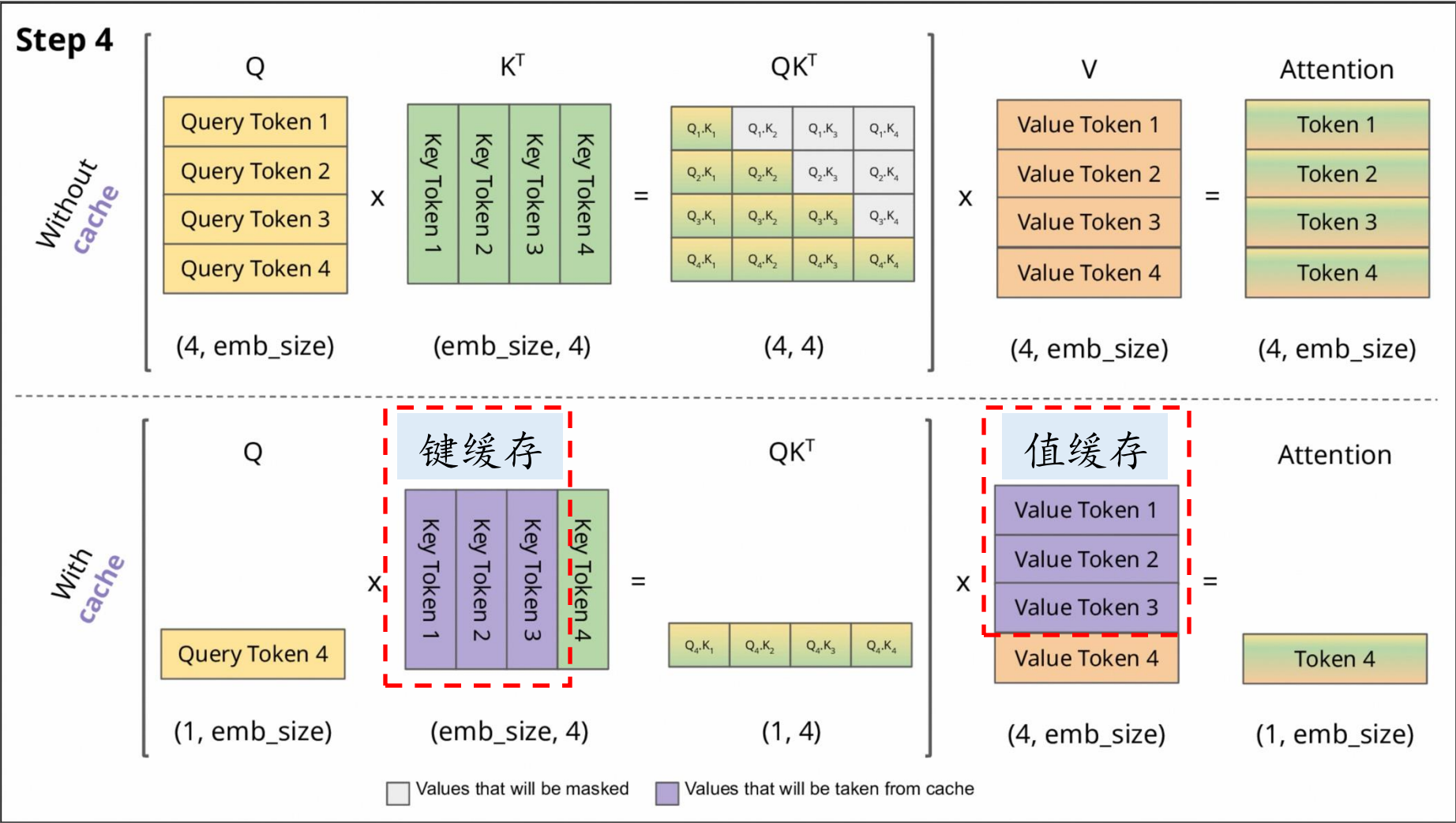


解码效率分析与加速算法

《大语言模型》编写团队：唐天一

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

➤ 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 u

输出：输出词序列 y

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

全量解码

$$\text{Attention}(q, K, V) = \text{softmax}\left(\frac{qK^T}{\sqrt{D}}\right)V$$

解码加速的两个关键

增量解码

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

解码效率的定量评估指标

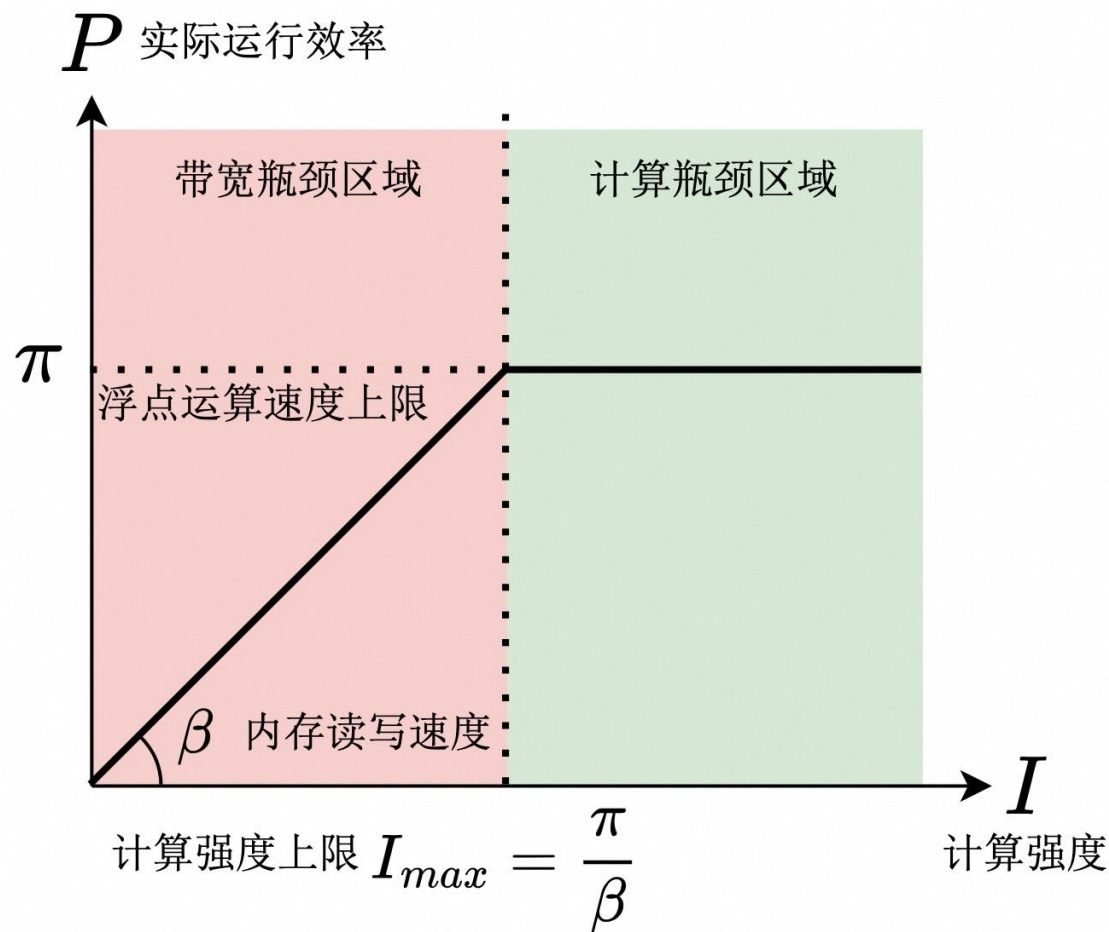
➤ 效率问题分析

➤ 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 $\mathbf{A} \in \mathbb{R}^{n \times m}$ 和矩阵 $\mathbf{B} \in \mathbb{R}^{m \times p}$ 相乘所需的运算量为 $2nmp$

➤ 矩阵乘法的访存量

➤ 矩阵 $\mathbf{A} \in \mathbb{R}^{n \times m}$ 和矩阵 $\mathbf{B} \in \mathbb{R}^{m \times p}$ 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 $\mathbf{X} \in \mathbb{R}^{B \times T \times H}$ 与矩阵 $\mathbf{B} \in \mathbb{R}^{H \times H}$ 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 $\text{softmax}(\frac{\mathbf{QK}^\top}{\sqrt{D}}) \mathbf{V}$, $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{B \times T \times N \times D}$
 - 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: $O(\frac{1 + \frac{1}{D}}{\frac{1}{D} + \frac{1}{T}})$

运算量、访存量 and 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 $Q, K, V = XW^{Q,K,V}$	$O\left(\frac{1}{\frac{1}{H} + \frac{1}{BT}}\right)$
② $Q, K = \text{RoPE}(Q, K)$	$O(1)$
③ $O = \text{Attn}(Q, K, V)$	$O\left(\frac{1 + \frac{1}{D}}{\frac{1}{D} + \frac{1}{T}}\right)$
④ $X = OW^O$	$O\left(\frac{1}{\frac{1}{H} + \frac{1}{BT}}\right)$
⑤ $X = \text{Add\&Norm}(X)$	$O\left(\frac{1}{1 + \frac{1}{BT}}\right)$
⑥ $G, U = X[W^G, W^U]$	$O\left(\frac{1}{\frac{1}{H} + \frac{1}{H'} + \frac{1}{BT}}\right)$
⑦ $D = \text{Swish}(G) \cdot U$	$O(1)$
⑧ $X = DW^D$	$O\left(\frac{1}{\frac{1}{H} + \frac{1}{H'} + \frac{1}{BT}}\right)$
⑨ $X = \text{Add\&Norm}(X)$	$O\left(\frac{1}{1 + \frac{1}{BT}}\right)$

$B = 8 \quad T = 1024 \quad H = 4096 \quad D = 128$

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

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）

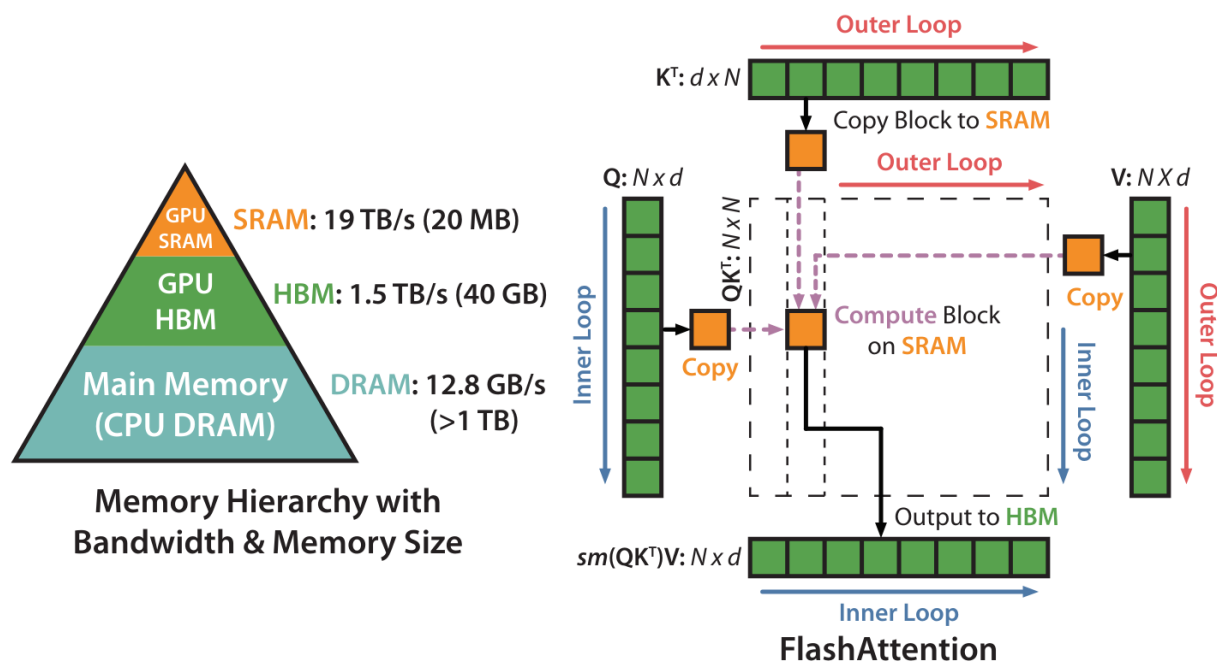
计算公式	计算强度
① $q, k, v = xW^{QKV}$	$O\left(\frac{1}{\frac{1}{H} + \frac{1}{B}}\right)$
② $q, k = \text{RoPE}(q, k)$	$O(1)$
③ $K, V = \text{Cache}(k, v)$	-
④ $o = \text{Attn}(q, K, V)$	$O\left(\frac{1 + \frac{1}{D}}{1 + \frac{1}{D} + \frac{1}{T}}\right)$
⑤ $x = oW^O$	$O\left(\frac{1}{\frac{1}{H} + \frac{1}{B}}\right)$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

$B = 8 \quad T = 1024 \quad H = 4096 \quad D = 128$

➤ FlashAttention

➤ 改进注意力计算 $\text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right)V$

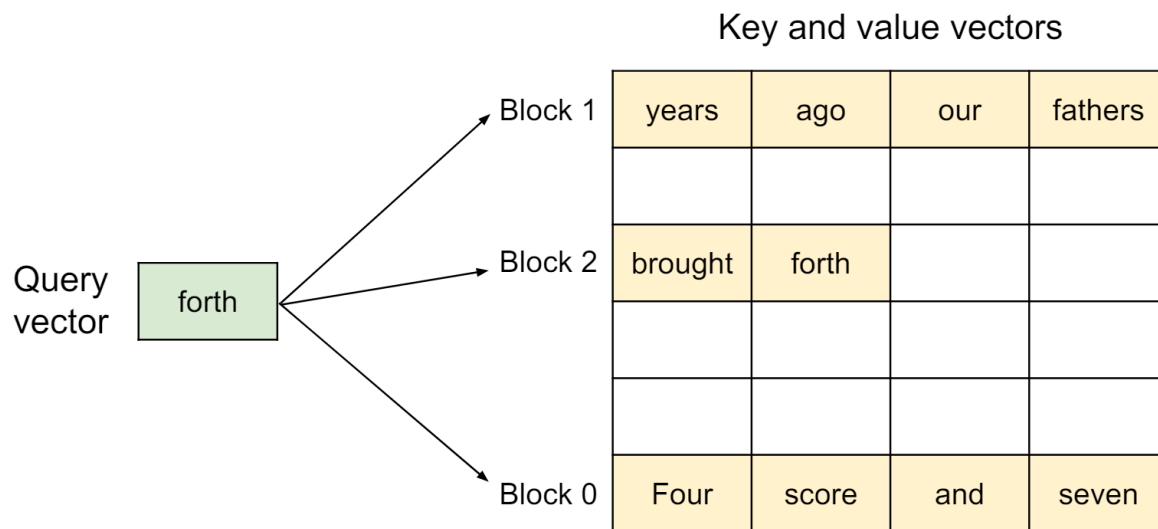
➤ 通过矩阵分块和算子融合，减少中间结果读写，减少访存量



将 QKV 分块计算，
在 SRAM 中直接计
算得到最终结果

➤ PagedAttention

- 优化键值缓存更新操作 $K' = K \oplus (x_{\hat{u}} W^K)$ 和 $V' = V \oplus (x_{\hat{u}} W^V)$
- 通过显存分页减少拼接时显存反复分配，同时提升注意力计算效率



查询与多个键值块
并行计算

显存分页

- 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

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
S_1	S_1	S_1	S_1	S_1	END		
S_2	S_2	S_2	S_2	S_2	S_2	S_2	END
S_3	S_3	S_3	S_3	END			
S_4	S_4	S_4	S_4	S_4	S_4	END	

传统批次推理

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
S_1	S_1	S_1	S_1	S_1	END	S_6	S_6
S_2	S_2	S_2	S_2	S_2	S_2	S_2	END
S_3	S_3	S_3	S_3	END	S_5	S_5	S_5
S_4	S_4	S_4	S_4	S_4	S_4	END	S_7

批次管理优化

➤ 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

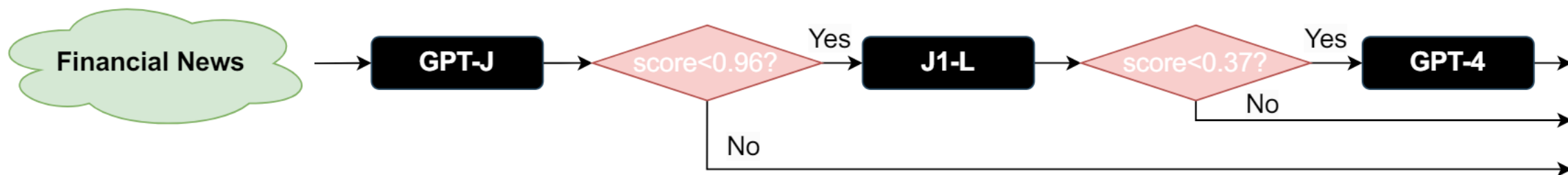
[START] japan ' s benchmark bend n	<div>小模型生成</div> <div>大模型拒绝</div> <div>大模型修改</div>
[START] japan ' s benchmark nikkei 22 5	
[START] japan ' s benchmark nikkei 225 index rose 22 6	
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points	
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 0 1	
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 9859	
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in	
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in tokyo late	
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in late morning trading . [END]	

➤ 推测解码流程示例（视频演示）



➤ 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

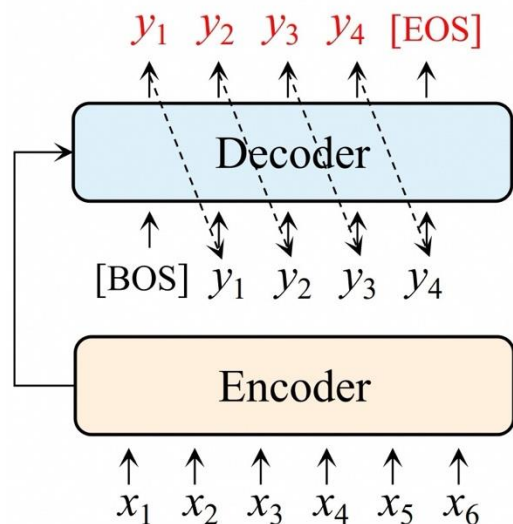


通过优先让相对较小模型进行解码，减少更大模型的调用开销

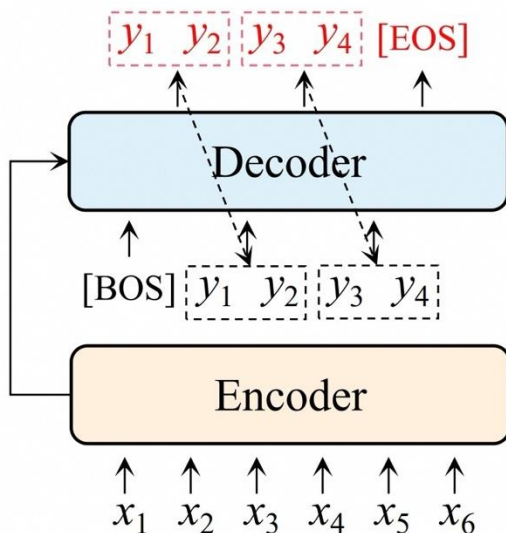
➤ 非（半）自回归解码

➤ 非自回归解码：基于输入一次性生成所有词元

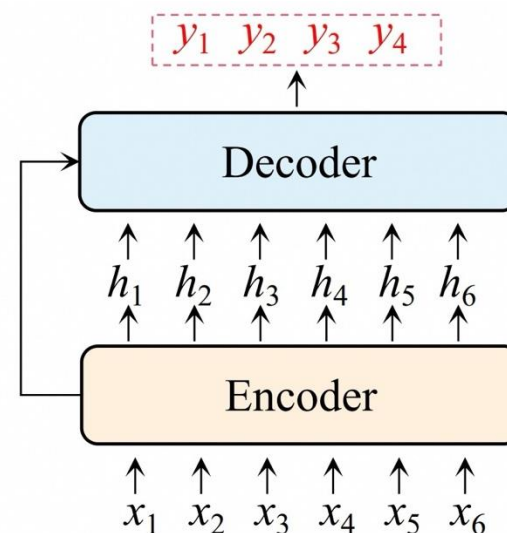
➤ 半自回归解码：组内非自回归生成，组间自回归生成



自回归解码



半自回归解码



非自回归解码

➤ 非（半）自回归解码

➤ 非自回归解码：基于输入一次性生成

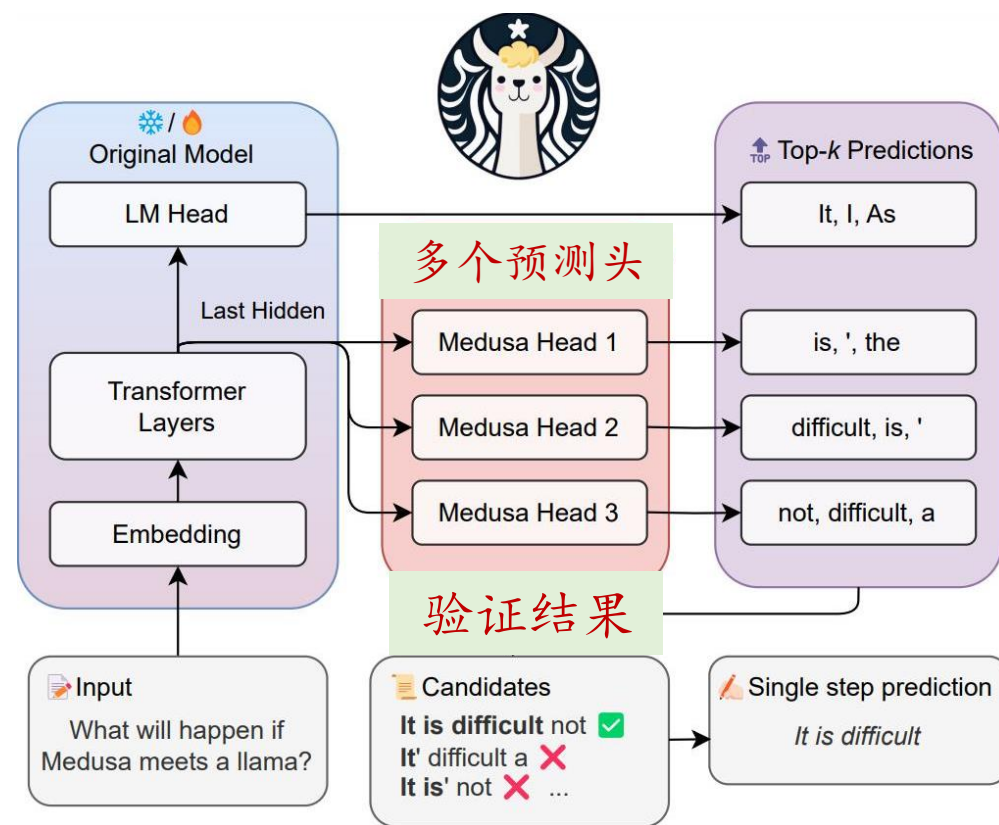
➤ 半自回归解码：组内非自回归，组间自回归

➤ Medusa

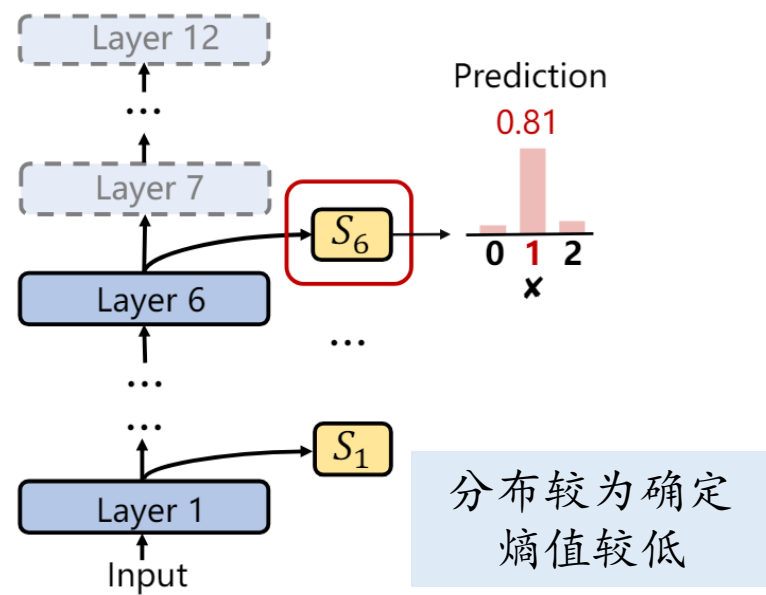
➤ 额外训练两个预测头分别预测第二个词和第三个词

➤ 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



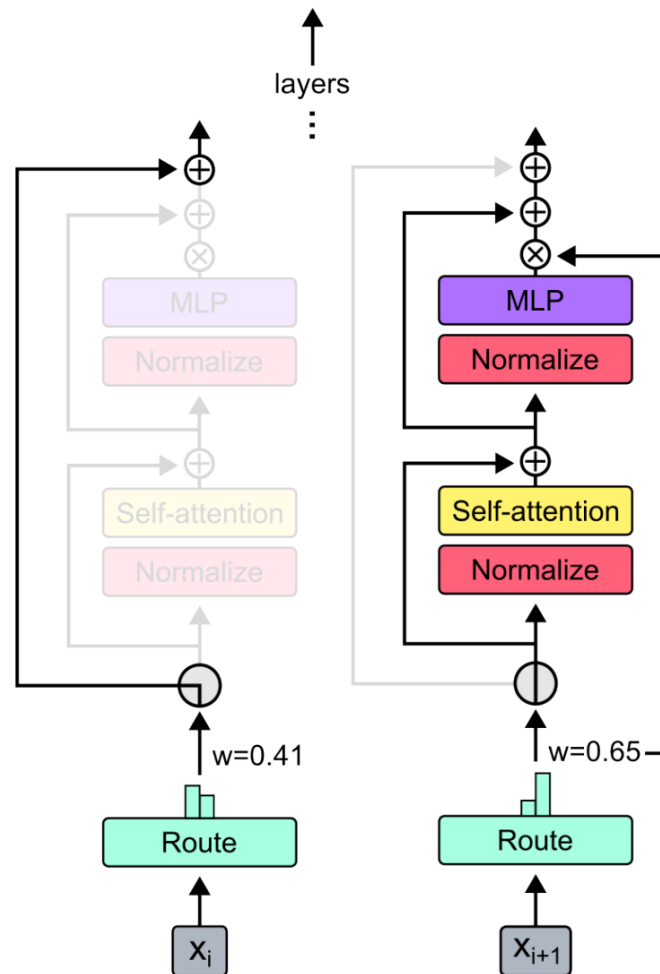
- 早退机制
 - 不需要所有层计算，满足条件跳过后续层计算
 - 每层得到输出概率分布，计算分布的熵值；如果熵值较低，则提前退出



Model	MNLI-m		MNLI-mm		QQP		QNLI		SST-2		MRPC		RTE		Macro
	Acc	Spd-up	Acc	Spd-up	F1/Acc	Spd-up	Acc	Spd-up	Acc	Spd-up	F1/Acc	Spd-up	Acc	Spd-up	
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

➤ 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





谢谢