

为什么 Decoder 使用最后一个位置预测下一个 Token

1. 问题

要讨论的观点是：

“只要某个 token 的隐藏状态集成了完整的上下文语义，它就可以用来预测下一个 token。标准 decoder-only Transformer 在 decode 阶段使用 causal attention，只有最后一个 token 具备完整前缀的语义。如果去掉这个限制，最后的打分阶段不一定必须使用最后一个 token。”

由此还可以提出一个更具体的设想：已经得到因果模型产生的 $h_1, \dots, h_t$ 后，再增加一次不使用 causal mask 的 attention，使所有位置都能访问完整前缀。这样得到的新表示是否都具备预测 x_{t+1} 的语义基础？

2. 核心理解

“最后一个 token” 本身没有特殊魔力。真正能够用于预测下一个 token 的，是一个满足以下两个条件的隐藏状态：

1. 它融合了当前完整前缀的信息；
2. 它在训练中与 next-token prediction 目标以及输出层完成了对齐。

标准 decoder-only Transformer 中，由于 causal mask，给定当前前缀

$$x_1, x_2, \dots, x_t, \quad (1)$$

只有最后一个位置的隐藏状态 h_t 能看到完整前缀。因此模型自然使用 h_t 预测 x_{t+1} 。

3. 标准语言模型中的训练对齐

causal mask 使第 i 个位置只能依赖它自己及之前的 token：

$$h_i = f_{\theta}(x_1, \dots, x_i). \quad (2)$$

共享的 LM Head 将每个位置的隐藏状态映射到词表概率：

$$p_{\theta}(x_{i+1} \mid x_1, \dots, x_i) = \text{softmax}(h_i W_{\text{out}} + b). \quad (3)$$

训练时并不是只训练序列的最后一个位置，而是在一次前向计算中并行训练：

$$h_1 \rightarrow x_2, \quad h_2 \rightarrow x_3, \quad \dots, \quad h_t \rightarrow x_{t+1}. \quad (4)$$

所以 decode 时使用 h_t ，不只是因为它位于张量的最后一行，而是因为：

1. 对当前长度为 t 的前缀，只有它看过 $x_1, \dots, x_t$ ；
2. 训练目标明确要求它预测该前缀之后的 x_{t+1} 。

换句话说，训练阶段的每个位置都是“它自己那个前缀的最后一个位置”。 h_i 被训练用于预测 x_{i+1} ，而不是当前完整前缀之后的 x_{t+1} 。

4. 再增加一次 Full Attention 的设想

假设在 causal Transformer 得到 $h_1, \dots, h_t$ 后，再计算：

$$(g_1, \dots, g_t) = \text{FullAttention}(h_1, \dots, h_t). \quad (5)$$

此时每个 g_i 在信息可见性上都可以访问完整前缀。因而从模型架构的可能性看，可以设计以下预测方式：

$$g_i \rightarrow x_{t+1}, \quad (6)$$

或者先汇聚所有位置：

$$\text{Pool}(g_1, \dots, g_t) \rightarrow x_{t+1}. \quad (7)$$

因此，“必须使用最后一个位置”并不是神经网络架构上的数学定理。只要某个表示真正汇总了完整前缀，并针对 next-token prediction 训练过，就可以用它预测下一个 token。

5. 为什么不能只在推理时临时添加 Full Attention

信息可见不等于表示已经与预测目标对齐。即使所有 g_i 都能访问完整前缀，也不能自动推出：

$$\text{softmax}(g_i W_{\text{out}} + b) \quad (8)$$

会给出正确的 x_{t+1} 分布。

原因包括：

1. 原模型的 h_i 被训练用于预测 x_{i+1} ，各位置承担的预测目标不同；
2. 不同位置仍然带有不同的位置编码、Query 和残差状态；
3. 新增的 Full Attention 如何汇总语义，以及输出应该落入 LM Head 的哪个表示空间，都需要训练学习；
4. 原有 LM Head 只对原模型产生的隐藏状态分布完成了训练对齐，未必能直接解释新模块产生的 g_i 。

所以，在训练好的 causal LM 后临时增加一层随机或未经联合训练的 no-mask attention，并不能保证任意位置都能预测同一个下一个 token。若要采用这种结构，Full Attention、位置选择或池化方式以及输出层都应当针对新目标联合训练。

6. 训练中的答案泄漏

如果训练时把完整句子交给 no-mask attention，同时仍让位置 i 预测 x_{i+1} ，那么位置 i 可以直接看到答案 x_{i+1} ，甚至看到更后面的 token。这会形成标签泄漏：训练损失看起来很低，但推理时这些未来信息并不存在。

要训练“完整前缀上的双向汇聚器”，至少需要满足以下条件之一：

1. 每个训练样本只输入当前真实可见的前缀 $x_1, \dots, x_t$ ，目标单独设为 x_{t+1} ；
2. 使用经过专门设计的 attention mask，保证用于预测的位置无法读取目标 token；
3. 增加独立的预测槽位，使该槽位读取完整前缀，但不能读取待预测答案。

这些方案在理论上可行，但会改变标准语言模型一次并行训练所有位置的方式，也可能增加训练或 decode 阶段的计算量。

7. 结论

用最后一个 token 预测下一个 token，不是理论上的唯一方案，而是以下因素共同决定的标准实现：

1. causal attention 决定了只有 h_t 看过当前完整前缀；
2. next-token prediction 损失把 h_t 与 x_{t+1} 对齐；
3. 只计算最新位置可以复用历史 KV Cache，实现高效自回归 decode。

因此， h_t 的特殊性来自“信息可见范围、训练对齐和推理效率”，而不是“最后一个位置天然更适合分类”。其他位置或汇聚表示也可以承担预测任务，但必须保证它确实融合完整前缀，并针对这一用途训练。

8. 后续分析基线

后续讨论以以下区分为基础：

1. “能够访问完整上下文”只描述信息可见性；
2. “具备适合预测的语义表示”还要求模型学会提取和组织这些信息；
3. “能够被现有 LM Head 正确解码”进一步要求隐藏状态与输出层完成训练对齐；
4. 比较不同方案时，还需要考虑标签泄漏、训练并行性和 KV Cache 复用成本。