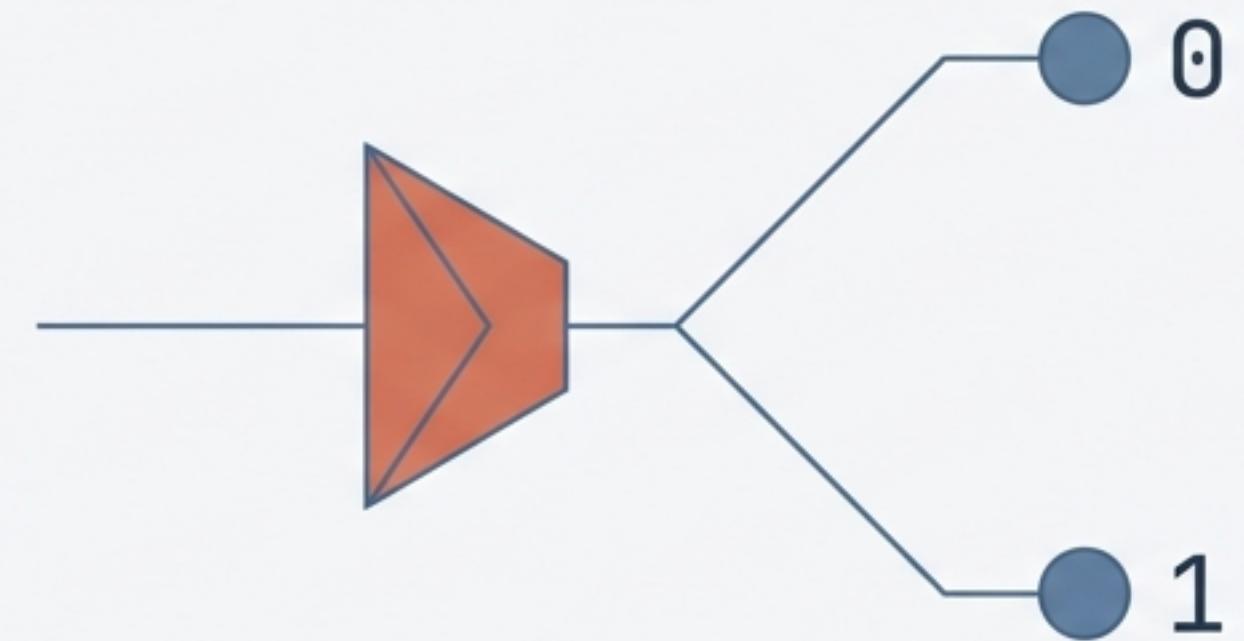


第6章：用于分类任务的微调

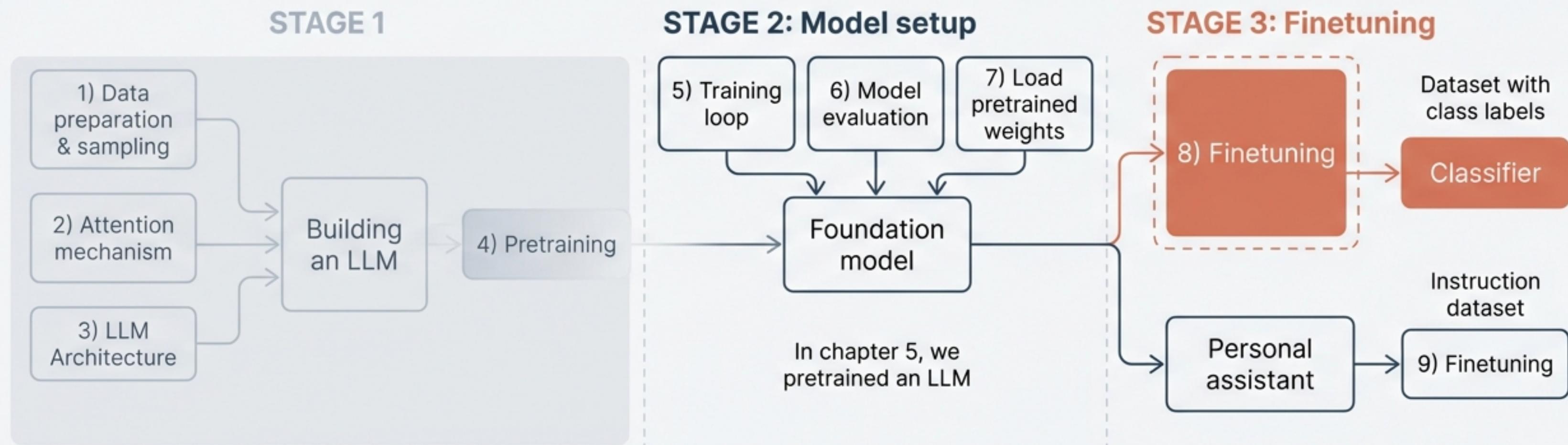
预训练模型准备与评估体系构建



Part 4：预训练模型准备

Part 5：分类损失与准确率计算

从通用生成到特定分类：LLM 开发阶段概览



当前状态

已拥有预训练的 GPT 基础模型
(Foundation Model)。

本节目标

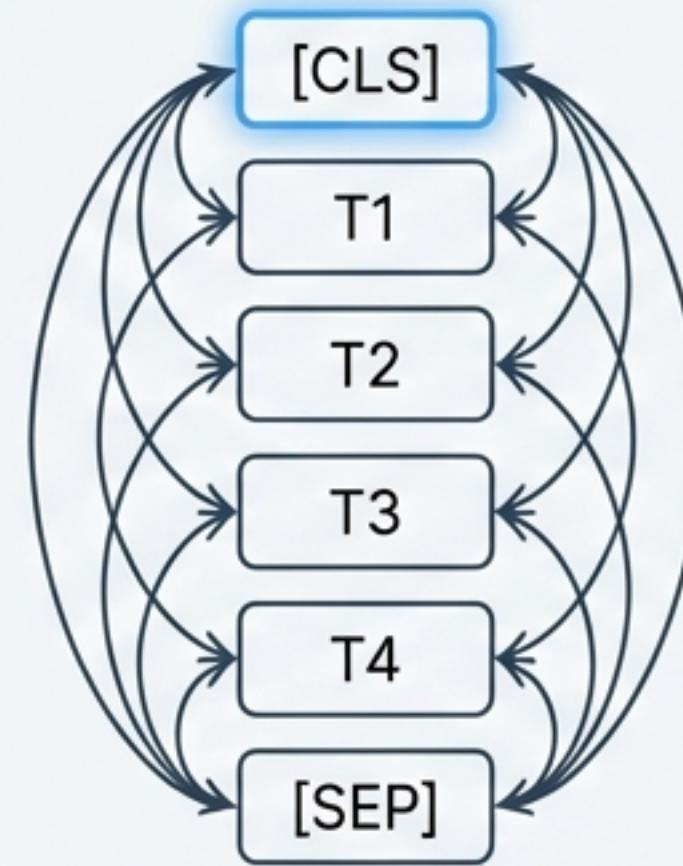
将“文本生成器”改造为“垃圾短信分类器”。

核心任务

1. 模型改造（修改输出层）
2. 评估构建（定义损失与准确率）

预训练模型选型机制：Encoder 与 Decoder 的差异

Encoder 模型 (如 BERT)

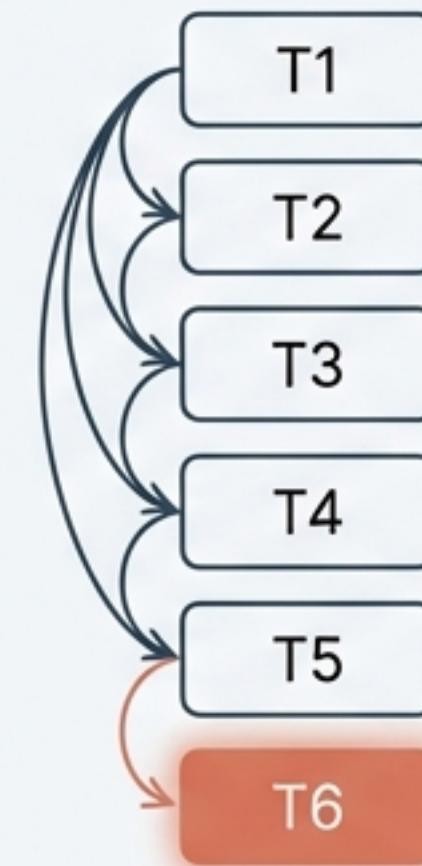


机制：双向注意力 (Bidirectional Attention)

优势：上下文全局可见

分类策略：使用 [CLS] Token

Decoder 模型 (如 GPT)



机制：单向因果注意力 (Causal Attention)

特性：仅见当前及历史 Token

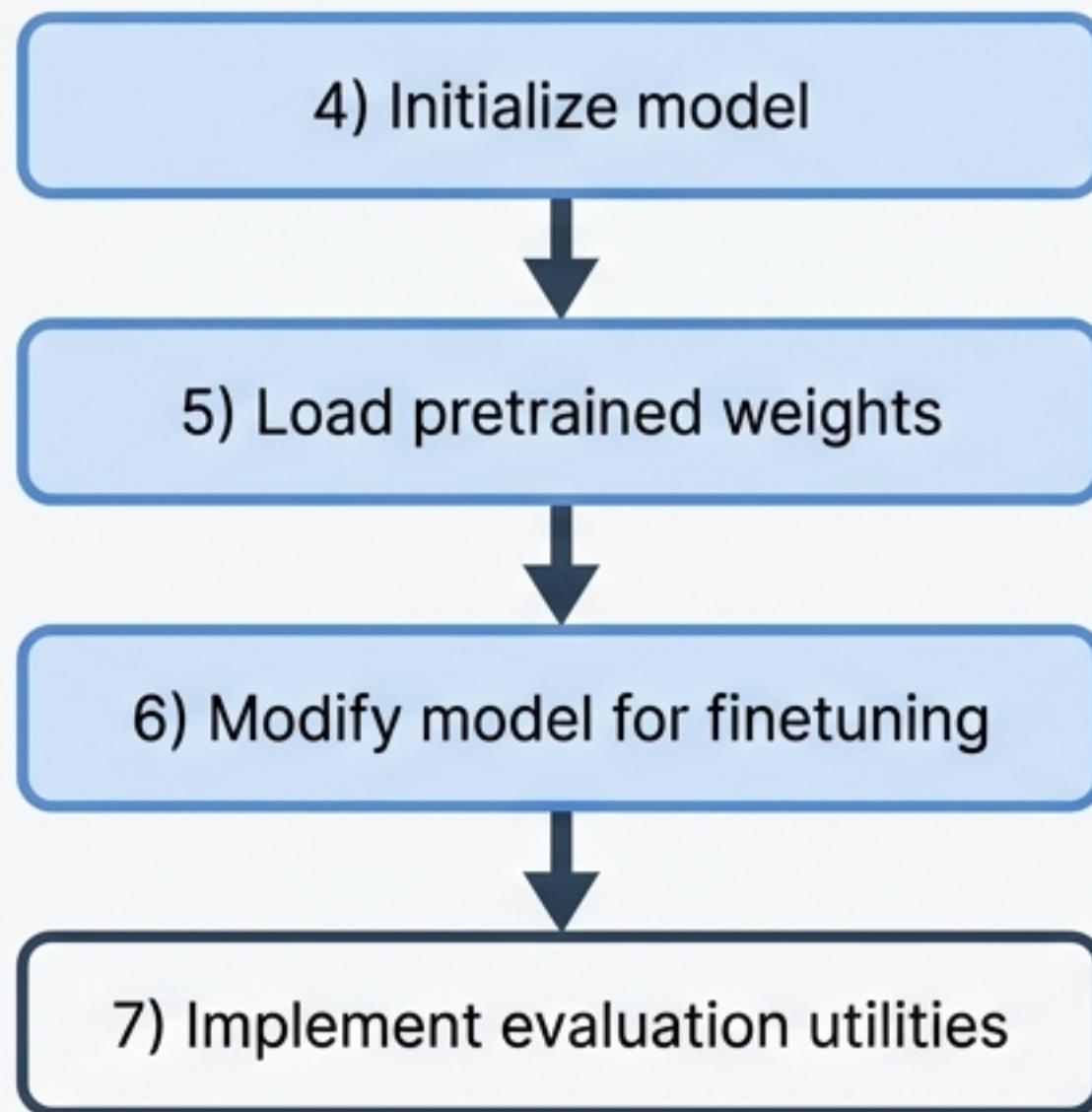
分类策略：使用最后一个 Token

结论：本课程采用 GPT (Decoder) 架构，利用最后一个 Token 的聚合信息进行分类。

预训练模型准备流程 (Model Preparation Workflow)

STAGE 2: Model setup

In this section, we initialize the pretrained model from the previous chapter that we will finetune



1. 加载模型 (Load Weights)

导入预训练权重，利用迁移学习 (Transfer Learning) 能力。

2. 架构改造 (Replace Head)

替换输出层，适配二分类任务。

3. 冻结策略 (Freeze Layers)

冻结底层参数，仅训练顶层以提高效率。

步骤 1：加载预训练权重

Technical Specs

模型: GPT-2 Small (124M)

词表大小: 50,257

上下文长度: 1024

嵌入维度: 768

Terminal

```
model = GPTModel(BASE_CONFIG)  
load_weights_into_gpt(model, params)
```

Output Log

Input: 'Every effort moves you'

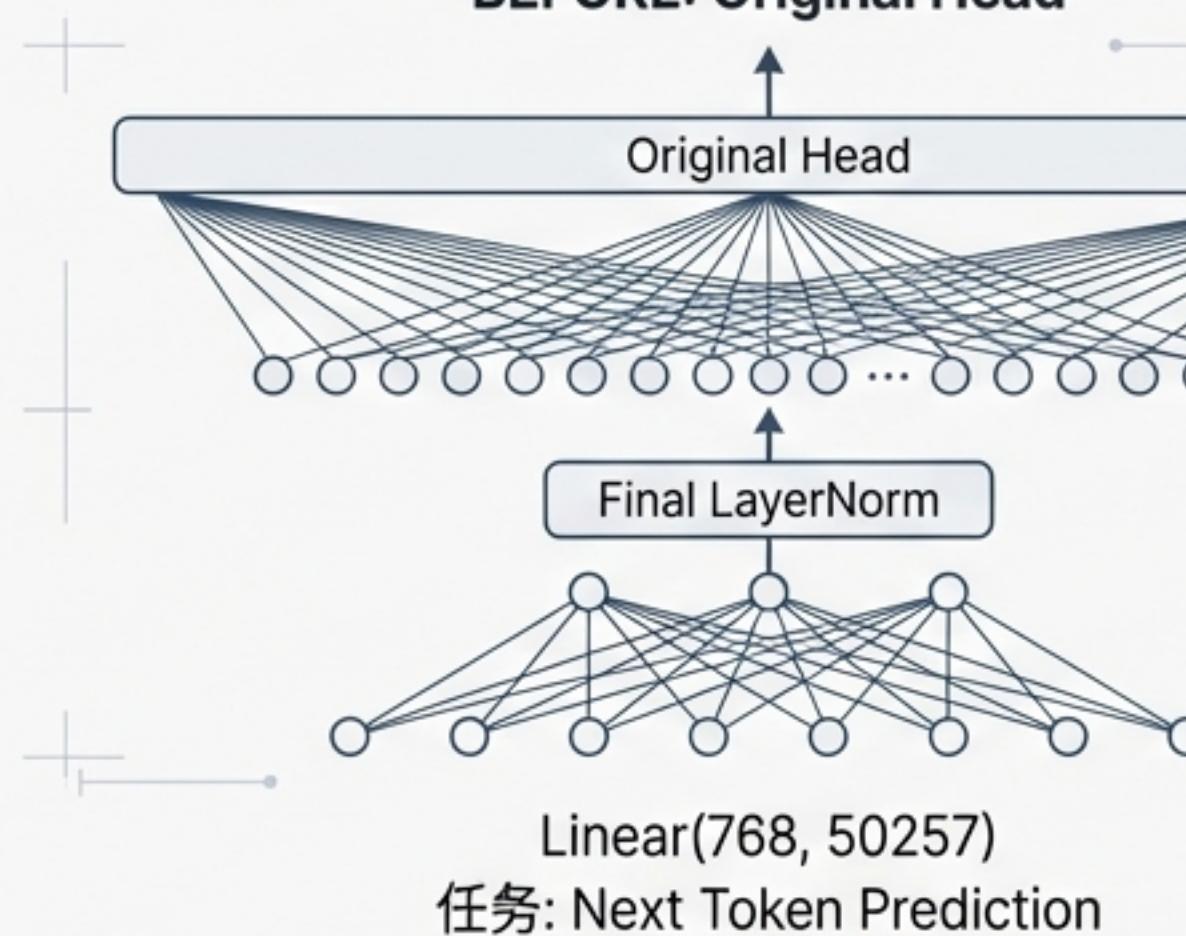
Output: '...forward. The first step is to understand...'

Analysis: 模型具备语言生成能力，但尚无法理解分类指令（Zero-shot 失败）。

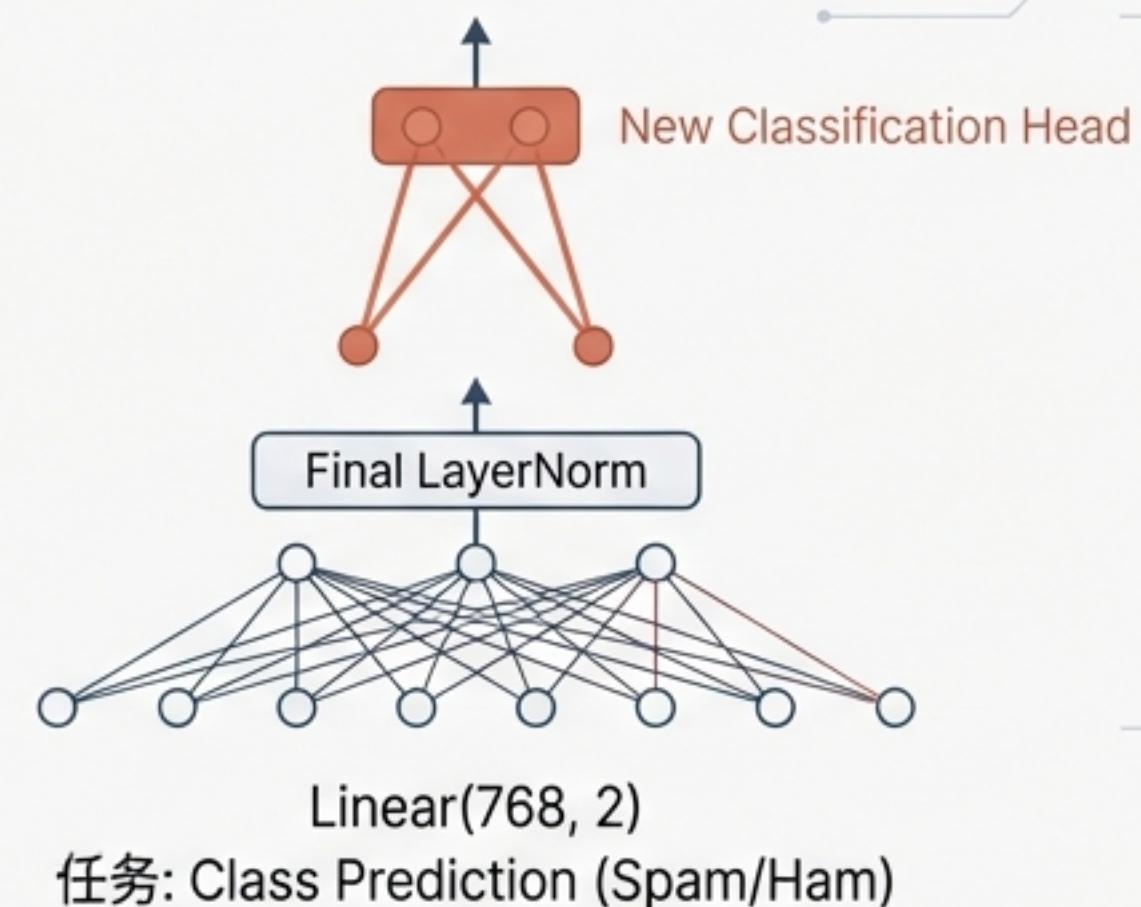
步骤 2: 替换输出层 (Modifying the Classification Head)

Before & After

BEFORE: Original Head

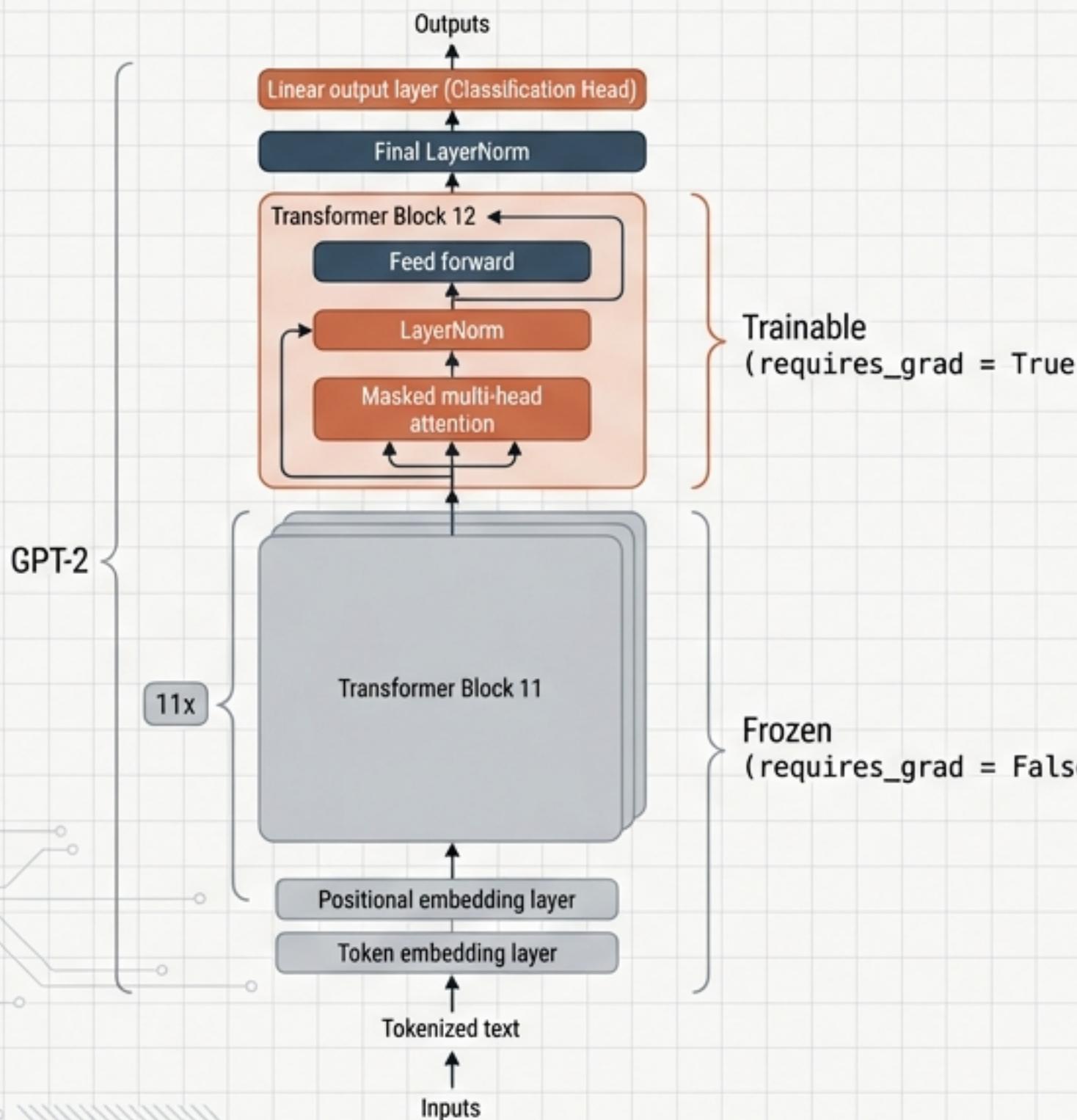


AFTER: New Classification Head



```
model.out_head = torch.nn.Linear(768, 2)
```

步骤 3：层冻结与可训练参数选择



训练策略

- Embedding Layers: **冻结**
- Transformer Blocks 1-11: **冻结**（保留通用语言特征）
- Transformer Block 12: **微调**（适应特定任务语义）
- Final LayerNorm: **微调**
- Classification Head: **全新训练**

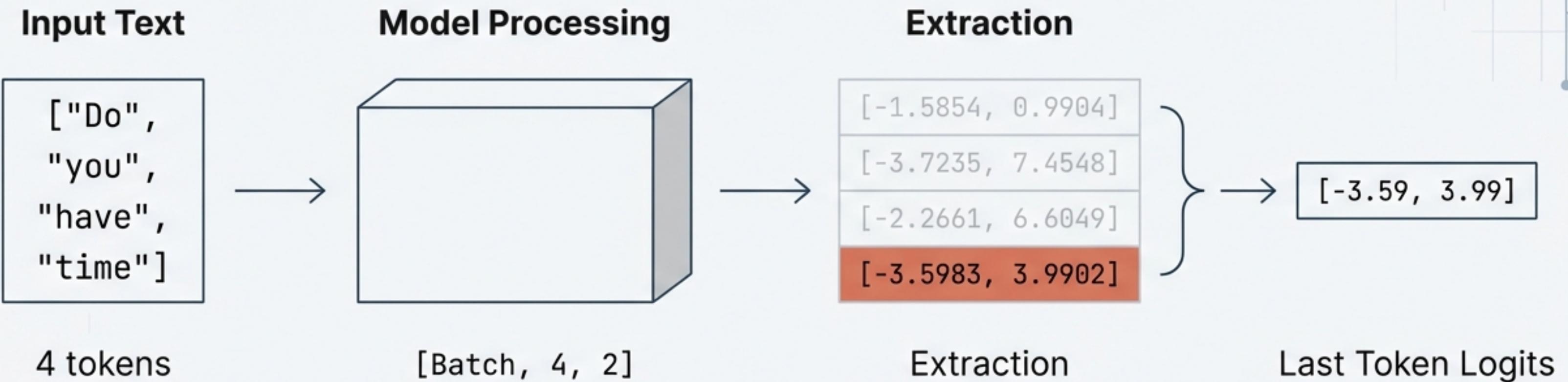
核心逻辑：为什么是“最后一个 Token”？



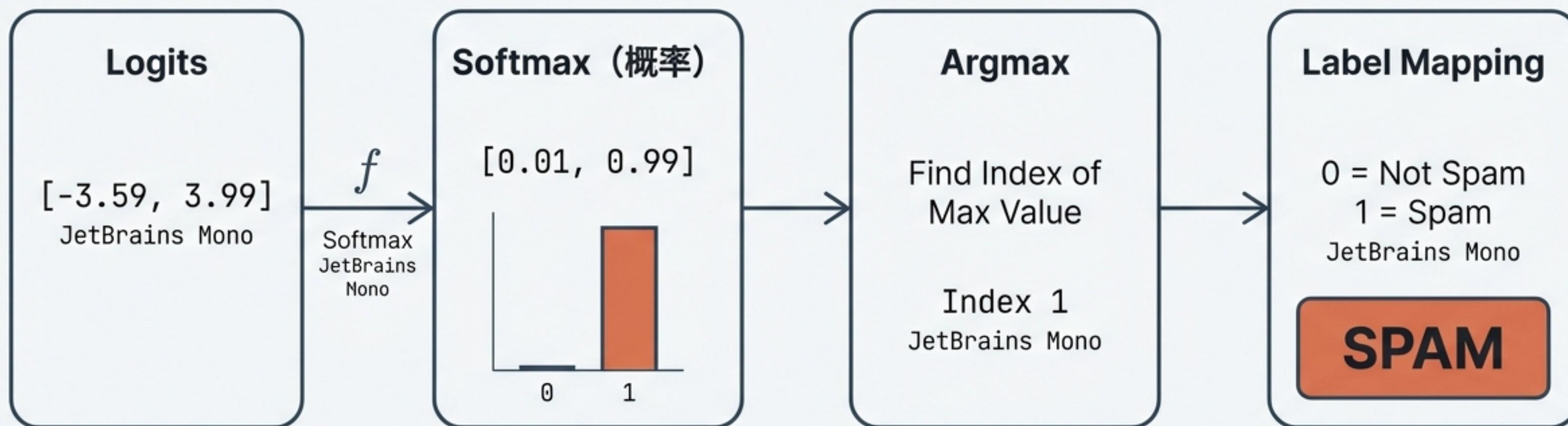
因果掩码 (Causal Mask) : Token 只能看见自己及之前的历史。

- Token 1 ('Do'): 仅包含 'Do' 的信息。
- Last Token ('time'): 聚合了 'Do you have time' 的完整语义。
- 结论: 最后一个 Token 是全句信息的唯一持有者。

数据流向：从输入序列到 Logits



预测逻辑：从 Logits 到类别标签



优化目标：交叉熵损失 (Cross Entropy Loss)

$\text{Loss} = \text{CrossEntropy}(\text{Logits_LastToken}, \text{Target_Label})$

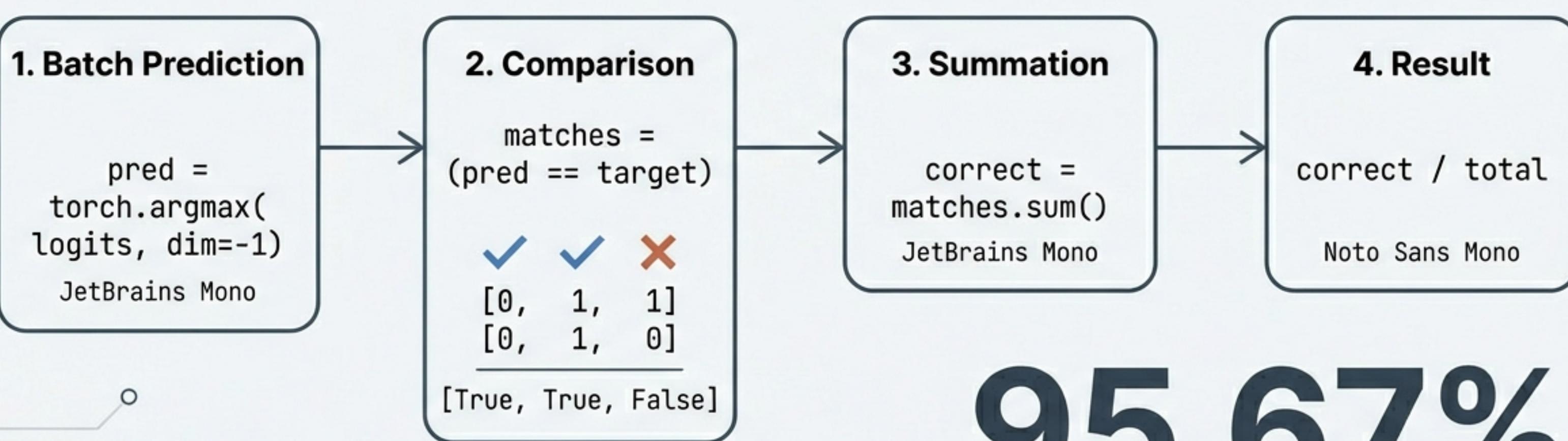


```
loss = torch.nn.functional.cross_entropy(logits, target_batch)
```

注意：计算损失时，忽略序列中除最后一个 Token 以外的所有位置。

性能评估：准确率 (Accuracy)

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Examples}}$$



95.67%

总结：从生成到分类的完整链路

模型适配 (Adaptation)	预测机制 (Prediction)	评估体系 (Evaluation)
 <ul style="list-style-type: none">替换 Head (768->2)冻结底层参数仅微调顶层 Block	 <ul style="list-style-type: none">利用 Decoder 因果特性提取 最后一个 TokenArgmax 确定类别	 <ul style="list-style-type: none">Cross Entropy 优化损失Accuracy 衡量效果关注最后一位输出