

Lec 9 LLMs

* The good & bad

* LLM Training: pre-training & post-training

Pretraining.

Clean up data:

RE: 过滤数据

Deduplication (去重): min Hash 去重

用 pretrained model 过滤低概率数据

训练模型识别 illegal texts

△ Scaling Law (缩放定律) → (经验观察)

参数↑、训练数据↑、训练迭代↑ ⇒ 效果↑

C: 计算量 N: 模型大小 D: 数据量

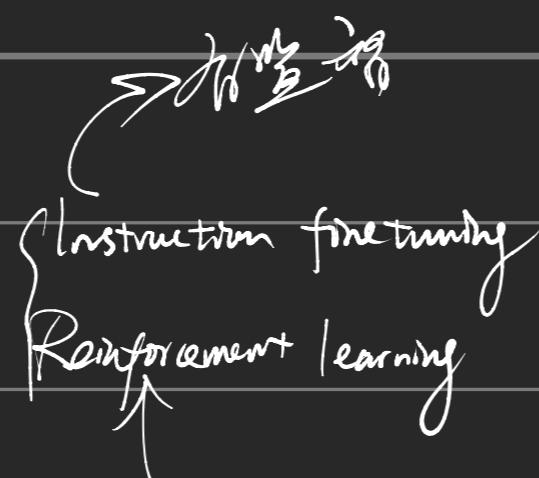
$$\Rightarrow \begin{cases} L(N) = \left(\frac{N_c}{N}\right)^{\alpha_n} \\ L(D) = \left(\frac{D_c}{D}\right)^{\alpha_d} \\ L(C) = \left(\frac{C_c}{C}\right)^{\alpha_c} \end{cases} \Rightarrow \begin{cases} N_{opt} \propto C^a \\ D_{opt} \propto C^b \\ C = 6DN \\ a+b=1 \end{cases}$$

a, b differ in different studies

△ Ability of Emergence → 演进能力

(小模型无但大模型有)

→ 某些任务中比大模型高



△ Instruction Fine-tuning (指令微调)

Background: 语言模型本身无法完成用户的需求

⇒ 微调语言模型 (从指令到语料监督学习),

使其可完成指令任务。

指令数据集训练:

手动创建

现有 NLP Tasks datasets

LLM生成 → 需要具备该能力

△ Parameter-Efficient Fine-tuning (PEFT) (参数高效微调)

调整一部分模型参数

prompt tuning (提示)

prefix tuning (前缀)

Adaptor (适配器模块)

LORA (低秩适应)

△ prompt: 将 Tunable Soft Prompt 加入到模型。

Mul-head Attn 中

△ prefix: 在调教 K, V, Q, P (将加 P_k, P_v 部分)

△ Adaptor: 将 Mul-head Attn 与 FNN 联接

Adaptor Network,

△ Low-Rank Adaptation

△ Reinforcement Learning with Human feedback

Background: 指令微调成本较高 ⇒ 人机交互

注入成功奖励 (已进行一部分指令微调)

⇒ RLHF: 学习与人类偏好保持一致 (奖励模型)

人类偏好

Reward

即: $E[R(x, \hat{y})]$ 最大. 其中 $\hat{y} \sim p_\theta(\hat{y}|x)$

模型参数

随机变量: 输出分布

人偏好吗?

⇒ Problem: 成步道 \Rightarrow m-步奖励模型 RMs

Parallel Decoding

⇒ Jacobi Decoding \Rightarrow 逐行解码

Guaranteed correct input

Guaranteed correct output \Rightarrow m-tokens 逐行
最多迭代 m-steps

Incorrect output

Speculative Decoding

Idea: { 快速 drafting }

{ 并行 LLM 多线程草稿 }

Drafting { 利用小模型 (Assistant model) }

{ Self-drafting (用原生模型生成初步) }

Use NAT (多线程并行解码)

Retrieval-based \rightarrow at named entities 使用

Verification: one token goes wrong \rightarrow 行为后验会修正

弃被拒绝

For drafting: 可以生成 token tree (逐层展开) .

→ 需要 Attention mask & position embedding.

同时并行验证

直接偏好优化

△ RL with rule-based reward over answers

(RL处理短语式奖励)

RV-cache

先前 tokens: K, V 被保存 (cache)

减少内存消耗: Head, Layer, Token

△ Multi-Query Attn \Rightarrow All heads use same KVs

△ Grouped-Query Attn \Rightarrow 将其归为 K, V 束

△ Multi-Head Content Attn (MCA)

$c = W_{DKV}h \Rightarrow$ (Transformer-层/层输出) \rightarrow Latent vec c

$k = W_k W_{kC} c, v = W_v W_{vC} c$ (但无需实际计算 $K \cdot V$)

$$z^T k = (W_Q h)^T (W_k W_{kC} c) = h^T (W_Q^T W_k W_{kC}) c \quad (\text{same for } V)$$

即：通过 c 反向 $K V$ 值并缓存 c .

穿透时投影嵌入 $K V$ 值

△ Share KV s across Layers

o Layer-Condensed KV-Cache (LCKV)

△ 保留及共享顶层 KV .

Jacobi 算法黑-热优化共享