NLP

Lecture 1

1. $P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A|B)P(B)}{P(A)}$ (likelihood → prior, → marginal)

$P(B|A \cap C) = \frac{P(A|B \cap C)P(B \cap C)}{P(A \cap C)}$

2. Bernoulli → 两分类 (一次)

Binomial: n堆 i.i.d 的 Bernoulli variable ⇒ 1个 binomial variable

categorical distribution → 多分类 n堆 → Multi-nomial

Lecture 2

1. n-gram language models

1.1) 句根概率 $P(w_1, \cdots w_n)$

1.2) 怎么拆 { Uni-gram: 词相互独立 $P(w_1) \cdots P(w_n)$

Bi-gram: 只和前一词有关 $P(w_1)P(w_2|w_1) \cdots P(w_n|w_{n-1})$

1.3) 模型 $\theta = \{P(w) : w \in V\}$ 在unigram中. 由MLE可得 $P(w_j) = \frac{C(w_j)}{\sum_i C(w_i)}$

Bi-gram $P(w|v) = \frac{\#(vw)}{\#\ \text{以}\ v}$ Tri-gram $P(w|uv) = \frac{\#(uvw)}{\#(uv)}$

1.4) Smoothing 解决 missing words $P_{Laplacian}(w_j) = \frac{\#w_j + 1}{\sum_i \#w_i + V}$ ← 词表大小

bigram: $P_{laplacian}(w_2|w_1) = \frac{\#(w_1, w_2) + 1}{\#(w_1) + V}$

2. 词表示方式 (有 symbolic 与 semantic 之争)

2.1 one-hot 无法用内积 measure similarity

2.2 distributional word-vector 多维, 每维表示一定特征

2.3 Word2Vec 在一个滑动 window 中, 分 center word 和 context word

$P(\text{context}|\text{center}) > P(\text{在窗外}|\text{center})$ (且离 center 越远 P 越小)

一个词 w 对应两个 vector { $v_w$: w 为 central; $u_w$: w 为 context

context ↓ center ↓

$P(w_o|w_c;\theta) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$ $v_c$ 是一样的 context word

优化: 令 $L = -\log P \Rightarrow = -u_o^T v_c + \log \sum_{w=1}^{V} \exp(u_w^T v_c)$ $\frac{\partial L}{\partial v_c} = -u_o + \sum \left[\frac{\exp(u_x^T v_c)}{\sum \exp(u_w^T v_c)}\right] u_x$ (G函数)

$v_c^{new} = v_c^{old} - \eta \frac{\partial L}{\partial v_c}$

$\Rightarrow = -u_o + \sum_{w=1}^{V} [G(u_w^T v_c)] u_w$

当预测完全正确, $\frac{\partial L}{\partial v_c} \to 0$ (因为 $\sum_w P(w|w_c)u_w = u_o$)

↑ 正样本分量 ↑ 远离错误的平均方向

把 $v_c$ 往真实上下文词 $u_o$ 方向推

小巧思, 用 negative sampling: 不和所有词比, 随机采 K 个 negative pairs

目标: 正对↑ 负对↓ $O(V) \to O(K)$ $K \ll V$

2.4 Word2Vec 两个改进

2.4.1 glove 利用 word2vec 没利用到的全局信息 → 比值能过滤无效信息

用共现ratio 概率比值比概率更重要 $P(k|x_1)$ $P(k|x_2)$ 用log是考虑词频

用 $v_k^T(v_i - v_j)$ 模拟 context word k偏向i还是j $\log \frac{P(k|i)}{P(k|j)}$ $w_i^T \tilde{w}_j + b_i + b_j \approx \log X_{ij}$

2.4.2 fasttext 把变形词当独立不好 (原来) 词 = character (k和谁更近) n-grams组合 $S(w_c, w_o) = u_o^T (\sum_{g \in w_c} z_g)$

$= \sum_g u_o^T z_g$

Lecture 3

# NLP Lec 3 POS TAG

给定 $O=(O_1, O_2, \cdots, O_T)$ 词 $O_t \in V$

预测对应 $Q=(q_1, q_2, \cdots, q_T)$ $q_t \in S$ ↙ Tag集合

核心任务 $Q^* = \arg\max_Q Pr(Q|O) = \arg\max_Q Pr(O|Q)P(Q)$ 分母 $P(O)$ 对所有Q一样

1. 为什么重要：① 语法结构 ②语义信息（词性不同意思不同）

2. 为什么难：① Lexical ambiguity 一个词 → 多个Tag 常见词易模糊用

3. 一个简单 BS $q_t = \arg\max_i Pr(q_i|O_t)$ 只看当前词，不看上下文

A. 只看 emission prob $Pr(O|Q)$ 因为O知道，其实就是 which tag suits O most

B. 只看 $P(Q)$ 忽略O的信息，通用不相关序列

生tag $Pr(Q) = Pr(q_1 \cdots q_T) = Pr(q_1) Pr(q_2|q_1) Pr(q_3|q_1, q_2) \cdots$

emit $Pr(O|Q) = P(O_1|Q)P(O_2|O_1, Q) \cdots$

$q_2|q_1$ 都 $|S^2|$ $q_3|q_1, q_2$ 更是 $|S^3|$

↓

Markov assumption ① $Pr(Q) = Pr(q_1) \prod_{t=2}^{T} Pr(q_t|q_{t-1})$ prior $Pr(q_1) = \pi_{q_1}$

定义 Transitional matrix $A_{ij} = Pr(q_t = j | q_{t-1} = i)$

$\sum i \to j = 1$ 从 POS tag $i \to j$ 的 prob

likelihood $|S| \times |S|$ $\sum_{j \in S} A_{ij} = 1$ 一般不对称

↓

② $Pr(O|Q) = \prod_{t=1}^{T} Pr(O_t|q_t)$ 当前word只由当前tag生成

emission matrix $B_{io} = Pr(O_t = o | q_t = i)$

从 POS tag i 到生成 word o 概率

posterior

↓

$|S| \times |V|$ $\sum_{o \in V} B_{io} = 1$

$$Q^* = \arg\max_Q \left[ \pi_{q_1} \prod_{t=2}^{T} A_{q_{t-1}, q_t} \prod_{t=1}^{T} B_{q_t, O_t} \right]$$

出现 prob

NLP Lec 4 Inference 句子出现的 prob

任务: $O=(o_1, o_2, \cdots, o_T)$ inference

以及固定的 HMM 参数 $\lambda=(A, B, \pi)$

求 $Pr(O|\lambda)$ 这个 HMM 生成 O 的 prob

1. 难. $Pr(O)=\sum_Q Pr(O, Q)$ ← hidden tag sequence |S| 有 $|S|^T$ 种组合

2. 为什么可以 DP

重复利用部分最优

3. Forward Algo $\max(ax, ay, bx, by)=\max(a,b)\times\max(x,y)$

Define forward prob $\alpha_t(j)=P(o_1, \cdots o_t, q_t=j)$ 汇总了所有可能 path 求 $q_t=j$ 的 prob

$P(A, B|C)=P(A|C)P(B|A,C)$

$\alpha_1(i)=P(o_1, q_t=i)=\pi_i \times b_i(o_i)$

↑ $Pr(q_t=i)$ ↑ $Pr(o_i|q_t=i)$

$N=|S|$

$$\alpha_t(j)=\sum_{k=1}^{N}\alpha_{t-1}(k)\, a_{kj}\, b_j(o_t)$$

$q_{t-1}=k \to q_t=j$ $q_t=j \to o_t$

这里 $o_t$ 只和 $q_t$ 有关

$P(o_t|o_{1:t-1}, q_{t-1}=k, q_t=j)=P(o_t|q_t=j)$

Proof: $P(o_{1:t}, q_t=j)=\sum_{k=1}^{N}P(o_{1:t-1}, q_{t-1}=k, q_t=j)$

$=\sum_{k=1}^{N}P(o_{1:t-1}, q_{t-1}=k)\,P(q_t=j, o_t|o_{1:t-1}, q_{t-1}=k)$

$=\sum_{k=1}^{N}\alpha_{t-1}(k)\,P(q_t=j|q_{t-1}=k)\times P(o_t|q_t=j)$

$=\sum_{k=1}^{N}\alpha_{t-1}(k)\, a_{kj}\, b_j(o_t)$

总

T步 ↑ 算这个 $O(N^2)$ ↓ $TN^2$

$$Pr(O)=\sum_j P(O, q_T=j)=\sum_j \alpha_T(j)$$

从 $N^T \to TN^2$

↑ 拆成上一步

4. Backward Algo

$\beta_t(i)=Pr(o_{t+1}, \cdots, o_T|q_t=i)$ 当前 state 为 i, 未来剩余 observation 的 prob

规定 $\beta_T(i)=1$ 无法生成更多 seq

$\beta_{T-1}(i)=\sum_{j=1}^{N}a_{ij}\, b_j(o_T)\,\beta_T(j)$

$:=P(o_T|q_{T-1}=i)$

N 种可能转移到下一状态 j 生成预测

$$\beta_t(i)=\sum_{j=1}^{N}a_{ij}\, b_j(o_{t+1})\times\beta_{t+1}(j)$$

$P(o_{t+1:T}|q_t=i)=\sum_{j=1}^{N}P(o_{t+1:T}, q_{t+1}=j|q_t=i)=\sum_{j=1}^{N}P(q_{t+1}=j|q_t=i)\,P(o_{t+1:T}|q_{t+1}=j, q_t$

$=\sum_{j=1}^{N}a_{ij}\,P(o_{t+1}, o_{t+2:T}|q_{t+1}=j)$

$=\sum_{j=1}^{N}a_{ij}\,P(o_{t+1}|q_{t+1}=j)\,P(o_{t+2:T}|o_{t+1}, q_{t+1}=j)$

$=\sum_{j=1}^{N}a_{ij}\, b_j(o_{t+1})\,\beta_{t+1}(j)$ 与 $o_{t+1}$ 无关

初始化 $\alpha_1(i)$ $\forall i$ of $q_1$

for $t=2, \cdots, T$

for $j=1, \cdots, N$

$\alpha_t(j)=\sum_k \alpha_{t-1}(k)\, a_{kj}\, b_j(o_t)$

$Pr(O)=\sum_j \alpha_T(j)$

$Pr(O) = \sum_{j=1}^{N} \pi_j b_j(O_1) \beta_1(j)$

$\pi_j$ ↑ $Pr(q_1=j)$ ; $b_j(O_1)$ ↑ $Pr(O_1|q_1=j)$ ; $\beta_1(j)$ ↑ $P(O_{2:T}|q_1=j)$

初始化 $\beta_T(i)$ $\forall i$ for $q_T$

for $t=T-1,\dots,1$

for $j=1,\dots,N$

$\beta_t(j) = \sum_k \beta_{t+1}(k) a_{jk} b_j(O_t)$

$Pr(O) = \sum_j P(q_1=j) b_j(O_1) \beta_1(j)$

Lecture 5 Decoding 参数学习 $\pi, A, B$

prediction 找最佳 tag

1. Structured Prediction

1.1 独立逐词 fails. 当前依赖邻居. 邻居依赖当前

1.2 Joint prediction $Q^* = \arg\max Pr(Q|O;\theta)$ $Q=[q_1,\dots q_T]$, $q_t \in S$

但若穷举 $O(N^T)$ 与 Viterbi 压缩至 $O(TN^2)$

2. Viterbi Algo 用了 DP

$$V_t(j) = \max_{q_1,\dots,q_{t-1}} Pr(q_1,\dots q_{t-1}, q_t=j, O_1, \dots, O_t)$$

$$V_1(j) = \max_{\phi} Pr(q_1=j, O_1) = Pr(q_1=j) Pr(O_1|q_1=j)$$

$\pi(j)$ $b_1(O_1)$ $b_{1,1}$

$$V_2(j) = \max_{q_1} Pr(q_1, q_2=j, O_1, O_2) = \max_{q_1} Pr(q_1) Pr(O_1|q_1) Pr(q_2=j|q_1) Pr(O_2|q_2=j)$$

$$= \max_{q_1} V_1(q_1) Pr(q_2=j|q_1) Pr(O_2|q_2=j)$$

$$V_t(j) = \max_{q_1,\dots q_{t-1}} (q_{1:t-1}, q_t=j, O_{1:t}) = \max_{q_{t-1}} \max_{q_{1:t-2}} Pr(q_t=j|q_{t-1}) Pr(O_t|q_t=j) Pr(q_{1:t-1}, O_{1:t-1})$$

$(q_{1:t-2}, q_{t-1}=i, O_{1:t-1})$ 交换 max $= \max_{q_{t-1}} Pr(q_t=j|q_{t-1}) Pr(O_t|q_t=j) \max_{q_{1:t-2}} Pr(\ )$

$q_t=j \mid q_{t-1}=i$

$O_t \mid q_t=j$

$$= \max_{q_{t-1}} Pr(q_t=j|q_{t-1}) Pr(O_t|q_t=j) V_{t-1}(q_{t-1})$$

$$= \max_i V_{t-1}(i) a_{ij} b_j(O_t)$$

必然来自使 $V_{t-1}(i) a_{ij}$ 最大的 path

$b_j(O_t)$ 与 i 无关

令 $P_t(j) = \arg\max_k V_{t-1}(k) a_{kj} b_j(O_t)$

保存最优路

若当前最优 path 与 j 结尾，则一定某上一状态 $P_t(j)$

从 $q_T^* = \max_k V_T(k)$ backtrack $q_{t-1}^* = P_t(q_t^*)$

算法 ① 初始化 $V_1(i)$ $\forall i \in q_1$ $TN^2$

② 从 $t=2,\dots,T$ T次

for $j=1,\dots,N$ N次

$V_t(j) = \max_k V_{t-1}(k) a_{kj} b_j(O_t)$ N次

$P_t(j) = \arg\max_k V_{t-1}(k) a_{kj} b_j(O_t)$

$O(TN^2)$

$O(T)$

③ Backtrack to find $Q^*$

④ 返回 $Pr(Q^*|O) = \max_j V_T(j)$

3. Expectation Maximization Algo 用了DP和MLE

E-step 在unlabeled data上算 estimated prob of labels ⇒ pseudo labels (prob)

labels: soft labels

M-step: estimate model paras using pseudo labels as in MLE

Estimating the soft labels

$\lambda = [A\ B\ \pi]$

E: $\xi_t(i,j) = Pr(q_t=i, q_{t+1}=j \mid O,\lambda) = Pr(q_t=i, q_{t+1}=j, O \mid \lambda) / Pr(O \mid \lambda)$

$\alpha_t(i) = Pr(O_{1:t}, q_t=i)$

$\beta_{t+1}(j) = Pr(O_{t+2:T} \mid q_{t+1}=j)$

$= Pr(O_{1:t}, O_{t+1}, O_{t+2:T}, q_t=i, q_{t+1}=j \mid \lambda) / \ldots$

$= Pr(O_{1:t}, q_t=i \mid \lambda)\, Pr(O_{t+2:T} \mid \ldots)$

$\alpha_t(i)\, Pr(q_{t+1}=j \mid q_t=i, \lambda)\, Pr(O_{t+1} \mid q_{t+1})\, \beta_{t+1}(j)$

对B $\gamma_t(i) = P(q_t=i \mid O,\lambda)$ i和O同时出现

$= Pr(q_t=i, O \mid \lambda) / Pr(O \mid \lambda)$

$= \alpha_t(i)\beta_t(i) / Pr(O \mid \lambda)$ 还要对i求和

$/ Pr(O \mid \lambda)$ 分母和这一项有关

Supervised MLE (隐状态序列已知)

$a_{ij} = \frac{C(i \to j)}{C(i)}$ $b_{i,o} = \frac{C(i \to o)}{C(i)}$ $\pi_i = \frac{C(q_1=i)}{m}$ ← # sentences in corpus

EM中 Given m sentences 一些有label

初始化 $\lambda = [A, B, \pi]$ random / POS-TAG if avail

loop until convergence:

E step: run forward, backward. 得到 $\alpha_t(j)$ $\beta_t(i)$ ↙DP.

M step: find $\xi_t(i,j)$ $\gamma_t(i)$ ↙MLE更新

$q: i \to j$

$$a_{ij} = \frac{\sum_{k=1}^{m}\sum_{t=1}^{T} \xi_t^{(m)}(i,j)}{\sum_{j}\sum_{k=1}^{m}\sum_{t=1}^{T} \xi_t^{(m)}(i,j)}$$

所有sentence 应该是j $\sum_j a_{ij} = 1$

$$b_{io} = \frac{\sum_{k=1}^{m}\sum_{t=1}^{T} [O_t=o]\, \gamma_t^{(m)}(i)}{\sum_{k=1}^{m}\sum_{t=1}^{T} \gamma_t^{(m)}(i)}$$

$i \to o$ 单个

$i \to$ o外所有

每一轮EM循环保证 $\log Pr(O \mid \lambda)$ 单调不减

但仅收敛至 Local Minimum 不良初始化易 fail

引入少量 supervised data和多次随机重启效果好

# Lec 6 Grammar and Syntax

1.1 n-grams & HMM

long-range dependency ×

只有 shallow syntax

1.2 constitute：一堆能组成unit的词. unit 可出现anywhere且保持词意. 保持时不能break down

context：句中除constitute的部分

⇓

context-free grammar CFG

2.1 四elements ① $N$: set of non-terminals 如 $\{S, NP, \cdots\}$

② $\Sigma$: set of terminals 实际词汇表

③ $R$: set of rules 形式为 $A \to \beta$, $A \in N$, $\beta \in (N \cup \Sigma)^*$

④ $S$: 起始符号且 $S \in N$

2.2 Derivations：① 单步 $\alpha A \gamma \Rightarrow \alpha \beta \gamma$ 若 $A \to \beta \in R$

② m步 $\alpha_1 \Rightarrow \alpha_2 \cdots \alpha_{m-1} \Rightarrow \alpha_m$ m次替换.

$A \to \beta$ 的应用仅取决于 non-terminal $A$ 本身

$L_G = \{w | w \text{ is in } \Sigma^* \text{ and } S \overset{*}{\Rightarrow} w\}$ 只包含terminal且从S开推

通过recursive完美建模 long-range dependency

支持多种句式

问题：① Ambiguity 一句对应多 parsing tree ② 搜索效率低下

3. Top down vs Bottom up

Top down：从S开始，用CFG展开non-terminal，匹配输入，无法匹配剪枝回溯用

Bottom up：从输入词(leaves)开始，用CFG生成non-terminal直到汇到S

Why it fails：① 重复子问题 ② late pruning

4. CYK Algo (DP) $O(n^3)$ $n^3$ $G$ rule size 每个子空间 $[i,j]$ 只解析一次 $n$个，需为term

4.1 CFG→CNF 严格限制一条rule右侧 $A \to BC$ 或 $A \to a$

转换算法

| CFG | CNF | |
|---|---|---|
| $A \to Bc$ | $A \to B$ $B \to c$ | 至多两个且两个得为non-term |
| $A \to B$ | 将所有 $B \to \gamma$ 规则赋给 $A$ | 变成 $A \to \gamma$ |
| $A \to BCD$ | $A \to XD$ $X \to BC$ | 多元产生形式 ⇒ 严格二叉树形式 |

4.2 CYK Parsing (n+1) x (n+1) matrix n words

~~w0~~ $^0$ $w_1$ $^1$ $w_2$ $^2$ $w_3$ $^3$ $w_4$ $^4$

(i,j) 包含所有可能的 non-terminals that can generate the constituent ranging i to j

∀(i,j)有 j > i+1 可以被 拆成 (i,j) = (i,k)+(k,j) (1,3) = (1,2) + (2,3)

对每个词 $w_j$ $T[j-1,j] = \{A \in N \mid A \to w_j \in R\}$

递推

$T[i,j] = \bigcup_{k=i+1}^{j-1} \{A \in N \mid \exists B \in T[i,k], C \in T[k,j] \text{ 且 } A \to BC \in R\}$

终止条件 $S \in T[0,n]$ 则句法

for j← 从1到句长： n

∀{A | A→words[j] grammar}

table[j-1,j] ← table[j-1,j] ∪ A init

for i 从 j-2 到 0 $n^2$ 所有子串长度

for k ← i+1 to j-1 do 从哪开始 有多长

for all {A | A→BC ∈ grammar and B ∈ [i,k] C ∈ [k,j] 对每个子串还要拆

table [i,j] ← table [i,j] ∪ A

R

A→BC O(R) 检查

# Lecture 7 Probabilistic CFG

CFG | CYK 判断一个句子有哪些合法 parse trees，但多个，哪个更有可能

⇒ 给 tree prob 并找出最优树

CFG | CYK 做不到

## 1. PCFG

1.1 给 production rule + prob $P(A \to \gamma)$ $\sum_{\gamma} P(A \to \gamma) = 1$

1.2 Parse Tree prob: $P(t|G) = \prod_{r \in t} P(r)$

- $t$：一颗 tree
- $r \in t$：$r$ 是 tree 里所有 rule
- PCFG 语法

1.3 Sentence prob：一个句子 $W_{1:T}$ 不是某 tree 的 prob，而是所有能生成这个 sentence 的 parsing tree 的总和

$$P(W_{1:m}|G) = \sum_{t \in T(W_1, \dots, W_m)} P(t|G)$$

- $W_{1:m}$：输入句子
- $t$：树
- $T(W_1, \dots, W_m)$

类似 $P(O) = \sum_{Q} P(O, Q)$，$Q$：hidden sequence

不能暴力枚举，可能指数级增 ⇒ DP!

⇓ tractable

1.4 三个关键假设

① Place invariant $P(N^j_{k,k+c} \Rightarrow \xi)$ 对不同 $k$ 一样

同一个 non-term 生成树的 prob 不依赖在句中位置.

② Context free $P(N^j_{kl} \Rightarrow \xi \mid \text{anything outside } [k,l]) = P(N^j_{kl} \Rightarrow \xi)$

若当前子树 cover span $[k,l]$，那么 span 外部上下文不影响内部展开

③ Ancestor-free 某个节点如何展开不依赖祖先

$$P(N^j_{kl} \Rightarrow \xi \mid \text{ancestor nodes outside } N^j_{kl}) = P(N^j_{kl} \Rightarrow \xi)$$

当前 NP 是从 S 1VP 下来，规则 prob 一样. 很强假设，也是局限

1.5 Inference and Estimation algos with PCFG

① 推句子的 prob $Pr(W_1, \dots W_m | G) = \sum_t Pr(t, W_{1:m}|G)$

② 找最优 parsing tree $t^* = \arg\max_t Pr(t | W_{1:m}, G)$

③ Learn a PCFG from a training corpus using MLE $G^* = \arg\max Pr(W_{1:m}|G)$

1.6 Inside | outside prob

outside：$\alpha_j(p,q) = P(W_{1(p-1)}, N^j_{pq}, W_{(q+1)m} | G)$

假设 pq 被 $N^j$ 代表，那么外部生成剩余词 prob

inside：$\beta_j(p,q) = P(W_{pq} | N^j_{pq}, G)$

$N^j_{pq}$ 表示 non-terminal $j$ derives word from $p$ to $q$

这棵子树内部生成这段 words 的 prob

1.7 Inside

1.7.1 Base case 若 span = 1 $p = q = k$ $W_k$

$\beta_j(k,k) = P(W_k | N^j, G)$ $N^j$ 展开 $W_k$ 的 prob

若有 rule $N^j \to W_k$ 则 $\beta_j(k,k) = P(N^j \to W_k)$

否则为 0

1.7.2 $\beta_j(p,q) = \sum_{r,s} \sum_{d=p}^{q-1} P(N^j \to N^r N^s) \beta_r(p,d) \beta_r(d,q)$

更大

枚举规则

t 切分

$$\beta_j(p,q) = \sum_{r,s}\sum_{d=q}^{q-1} P(w_{pd}, N^r_{pd}, w_{(d+1)q} N^s_{(d+1)q} \mid N^j_{pq}, G)$$

$$= \sum\sum P(N^r_{pd} N^s_{(d+1)q} \mid N^j_{pq}, G)\, P(w_{pd} \mid N^j_{pq}, N^r_{pd}, N^s_{(d+1)q}, G)\, P(w_{(d+1)q} \mid N^j_{pq}, N^r_{pd}, N^s_{(d+1)q}, G, w_{pd})$$

位置无关 | ancestor 和G无关. | ancestor | context 上下文 | 无关

$$= \sum\sum P(N^j \to N^r N^s)\, \beta_r(p,d)\, \beta_s(d+1,q)$$

1.8 Outside $\alpha_j(p,q) = P(w_{1:p-1}, N^j_{pq}, w_{q+1:m} \mid G)$ 和inside有关系

Base case $P(w_{1:m} \mid G) = \sum_j \alpha_j(k,k) P(N^j \to w_k)$

$P(w_{1:m}, N^j_{pq} \mid G) = \alpha_j(p,q)\beta_j(p,q)$

若 $N^1 = S$ 覆盖句子 $[1,m]$ 外面什么没有，那 $\alpha_1(1,m)=1$ 其余non-terminal $\alpha_j(1,m)=0, j\neq 1$.

整句必须从S开始

$$\alpha_j(p,q) = \Big[\sum_{f,g} \sum_{e=q+1}^{m} \alpha_f(p,e) P(N^f \to N^j N^g)\, \beta_g(q+1,e)\Big]$$

自己在父左侧 | 更大的父亲f $f \to_{左} j\ g_{右}$ | 用右的里填充. | 只是单层上一级

$$+\Big[\sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q) P(N^f \to N^g N^j)\, \beta_g(e,p-1)\Big]$$

$f \to_{左} g\ j_{右}$ | 更大父亲f | 用左的inside prob填充

1.9 Optimal parsing tree 定义 $\delta_j(p,q)$ 表示生成 span $[p,q]$ 在j下 最高prob parse tree.

若 span长度为1 $\delta_j(k,k) = P(N^j \to w_k)$

prob ⇐ $\delta_j(p,q) = \max_{r,s,d} P(N^j \to N^r N^s)\, \delta_r(p,d)\, \delta_s(d+1,q)$ 和inside递推类似

一个值

记录 $\psi_j(p,q) = \operatorname{argmax}_{r,s,d} P(N^j \to N^r N^s)\, \delta_r(p,d)\, \delta_s(d+1,q)$ $\sum \to \max$

具体方式 ⇐

$= (r,s,d)$ 表示 左孩是 $N^r$ 右孩 $N^s$ split point为d. (左子树状态 / 右子树状态)

然后从 $\delta_1(1,m)$ 开始回溯，恢复整棵 parse tree

NLP Lec 8

1. n-gram 3问题 { ①数据稀疏 ②复杂度随数据n指数增长 ③用 fixed window ☆ rigid straightforward

↓ 神经网络. 参数更高效. 可用不同设计解决不同问题

2. Logistic Regression

$a = \sigma(\sum_i w_i x_i + b)$ ↑非线性压到(0,1) , $l(a,y) = -[y\log a + (1-y)\log(1-a)]$

$x \in R^m$ hidden size n.

$W \in R^{n\times n}$ $U \in R^{n\times m}$

3. RNN $h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$

$h^{(t)} \to h^{(t+1)}$ $h \to o$

biases

$\theta = \{U, W, V, b, c\}$

$x^t \to h^{(t+1)}$

$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$

$h^{(t)} = \tanh(a^{(t)})$

$o^{(t)} = c + Vh^{(t)}$

$\hat{y}^{(t)} = \text{softmax}(o^{(t)})$

$h^t$

$h^{(t)} = \tanh(b + Wh^{(t-1)} + Ux^{(t)})$

$= \tanh(b + W(\tanh(b + Wh^{(t-2)} + Ux^{(t-1)})) + Ux^{(t)})$

NLL Loss $-\sum_t \log P_{model}(y^{(t)} | \{x^{(1)}, \cdots, x^{(t)}\})$

↑

$\hat{y}^{(t)} = \text{softmax}(o^{(t)})$ $y$

BPTT

Back Propogation through time

gradient vanishing or explosion

若 W的 $|\lambda|<1$ $W^t \to 0$ 历史遗忘, $|\lambda|>1$ $W^t \to \infty$ 梯度爆炸.

4. Why tanh? 引入非线性 将值域压到 [-1,1] 对矩阵指数增长起截断. 缓解优化敏感度

$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

BPTT中梯度含有 $\prod \frac{\partial h^{(k)}}{\partial h^{(k-1)}}$ tanh导数 $\in (0,1]$ 使连乘不易发散, 但长序列仍面临梯度衰减问题

$y' = 1 - y^2$

令 $u = e^x - e^{-x}$ $v = e^x + e^{-x}$ $(\frac{u}{v})' = \frac{u'v - v'u}{v^2} = \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})^2}{(e^x + e^{-x})^2}$

$= \frac{v^2 - u^2}{v^2} = 1 - \frac{u^2}{v^2} = 1 - y^2$

Lecture 9

翻译难的不是找对应单词，而是词义消歧，语序重排，文化/语境适配

$\hat{T} = \arg\max_T P(T)P(S|T)$ T是target S=foreign.

fluency $P(E)$ faithfulness $P(F|E)$ $E = (e_1, \cdots, e_I)$ target 外语 $F = (f_1, \cdots, f_J)$

$E^* = \arg\max_E P(E|F) = \arg\max_E \frac{P(F|E)P(E)}{P(F)} = \arg\max_E P(F|E)P(E)$

翻译方向是F→E ↑固定

IBM Model 1 给定E，生成F

1. 生成F的长度J 2. 生成Alignment A； 3. 根据A，从E一对一逐个生成F.

$E \to A \to F$ $P(F|E) = \sum_A P(F, A|E)$

① bag of words. 所有alignment位置一样，不懂语序 所有alignment相同prob.

② independent word trans 只看词对，不看上下文 $P(f_j|e_{a_j})$独立

③ 只能处理简单 one to many 但多个词经常是联合生成

$P(F, A|E) = \frac{\varepsilon}{(I+1)^J} \prod_{j=1}^{J} P(f_j|e_{a_j})$ I和J能匹配的prob是 $\frac{1}{(I+1)^J}$ ↑NULL

Lecture 10

1. HMM对齐联合概率 ↗先决定长度I

$P(F, A|E) = P(J|I) \prod_{j=1}^{J} P(a_j|a_{j-1}, I)P(f_j|e_{a_j})$

用这个alignment生成$f_j$ 根据上个alignment生成新的

Markov Assumption: ① $P(a_j|f_1^{j-1}, a_1^{j-1}, e_1^I) = P(a_j|a_{j-1}, I)$ 当前alignment只与上一个alignment和目标句长度有关

② $P(f_j|f_1^{j-1}, a_1^j, e_1^I) = P(f_j|e_{a_j})$ 当前foreign word只由它对齐到的目标词生成，不依赖别的

$P(F|E) = P(J|I) \times \sum_A \prod_{j=1}^{J} P(a_j|a_{j-1}, I)P(f_j|e_{a_j})$

2. Transition prob 应鼓励 local alignment

对$P(a_j|a_{j-1}, I)$，希望 $a_j \approx a_{j-1}$ 看相对距离，不看绝对位置

trans model language model

3. Decoding $\hat{E} = \arg\max_E P(F|E)P(E)$ 希望最大化$P(E|F)$

若E已知，则可用Viterbi 但翻译时E不知 变成在所有E中搜

带bigram language model的general coding是NP-complete

必须用 heuristic search

3.1 Search-based decoding 把一个状态~~设成~~定义为partial translation

包含①已翻译的foreign span ②对应生成的target words ③当前accumulated score

searching tree节点为partial translation

3.2 Best-first Search：每次扩展score最高的 简单，但需维护大量候选节点，只看当前易陷于local optimum

3.3 A* search：当前质量+未来潜力 $f^*(p) = g(p) + h^*(p)$ 未来质量 但$h^*(p)$难估准，需要heuristic

综合优先级

3.4 Beam Search 最常见：每一步只保留 top-k 个当前最好状态

① 初始化 Beam ② 扩展 Beam 所有状态 ↑ beam size

③ 对所有扩展打分 ④ 保留 top-k ⑤ 重复直到生成结束

比 greedy 稳，比 exhausive search cheap

实践效果好；但不保证全局最优，k 太小易错过好番羽，k 太大计算成本高

4. Evaluations BLEU

n-gram precision $prec_n = \frac{\sum_{c \in \{candidate\}} \sum_{n\text{-gram} \in c} Count_{match}(n\text{-gram})}{\sum_{c \in \{candidate\}} \sum_{n\text{-gram} \in c} Count(n\text{-gram})}$

分子：n-gram 在 candidate 出现次数

分母：n-gram 在 reference 出现次数

$BLEU = \left(\prod_{n=1}^{4} prec_n\right)^{\frac{1}{4}}$ 几何平均 只看 unigram 过于宽松

clip：对每个 n-gram，限制在参考译文中最多被匹配的次数

一个5词句，分母 for unigram 是 5 不超过参考中该 n-gram 最多出现次数

5. Seq2Seq 直接学 $P(y_{1:T} | x_{1:T})$ 输入输出长度可以不同

5.1 encoder $h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$

5.2 decoder $d^{(t)} = f(d^{(t-1)}, y^{(t-1)}; \theta)$

5.3 MLE $P(y_{1:T'} | x_{1:T}) = \prod_{t=1}^{T'} P(y_t | v, y_{1:t-1})$ ← 前一个生成目标词

$P(y_t | v, y_1, \cdots, y_{t-1}) = softmax(o^{(t)})$

$o^{(t)} = Vd^{(t)}$

5.4 问题 ① fix-length hidden vector 会把 source sentence 压到固定长，无论 sequence 多长

② 仍然不能 encode long-range dependency：有 sequential recency bias

较近 token 过矩阵乘法少，更易保留，较远被递归，易被冲淡

③ Gradient vanishing / explosion

④ 无法并行计算，必须按时间顺序算

5.5 Attention mechanism：生成一个词可动态看 source sentence 所有 hidden state

$c_t = \sum_{j=1}^{T} \alpha_{t,j} h_j$ ↑ source 长

$\alpha_{t,j} = \frac{\exp(Score(t,j))}{\sum_{T} \exp(Score(t,k))}$

$Score(t,k) = \langle s_{t-1}, h_t \rangle = s_{t-1}^{T} W_a h_t$

$s_t = f(s_{t-1}, y_{t-1}, c_t)$ 多了 $c_t$

优：① 缓解 fixed-length vector bottleneck 不足：仍是 RNN，不能并行

② 可直接访问 source sentence 的 hidden state

③ long-range dependency

④ attention weight 有一定解释性

Lecture 11

1. Transformer { PE, MLP, Masked MHA } expressive ; { RC, LN } 更好训 — model should be large, data

1.1 PE $PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right)$ $PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right)$ $T = 2\pi\, 10000^{\frac{2i}{d}}$

pos是token在序列中位置; d - embedding dimension, 2i和2i+1是每个embedded维度

$h_t = E(w_t) + PE_t$ 差 i↑ → T↑ 拟合了不同距离的dependency

1.2 MLP

dim(x) → dim(x)*4 → dim(x)*4 → dim(x) 进出dim同, 可stack模块

因为 self-attention layer 只能找到 linear combination of value vector (GELU)

MLP可引入非线性

1.3 Self-Attention $X \in R^{B \times T \times C}$ (B: batch size, T: token length, C: 每个token dim)

$Q = XW_Q \quad K = XW_K \quad V = XW_V$

self.c_attn = $[W_k, W_q, W_v] \in (n_{embd}, 3n_{embd})$

$\Rightarrow Q, k, V \in R^{B \times n_h \times T \times h_s}$ $C = nh * hs = n\_embd$ (nh: 头数, hs: 头大小)

(n_embd, 3*n_embd)

q, k, v = self.c_attn(x).split(self.n_embd, dim=2) ⇒ B×T×C 每个

k = k.view(B, T, self.n_head, C // self.n_head).transpose(1, 2)

- B×T×3C

把C拆成 n_head * size_head.

⇒ (B, T, #head, size(head))

转置 ⇒ (B, nh, T, hs)

$A = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$ $Output = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

att = (q @ k.transpose(-2, -1)) ⇒ [B, nh, T, T] / math.sqrt(k.size(

[B, nh, T, hs] [B, nh, hs, T]

(T×hs) * (hs×T] ⇒ [T×T] 除以 hs

att = att.masked_fill(self.bias[:,:,:T,:T]==0, float('-inf')) 上三角为0的矩阵

att = F.softmax(att, dim=-1) 不让看future

y = att @ v ⇒ [B, nh, T, hs] softmax(-∞) = 0

y = y.transpose(1,2).contiguous().view(B, T, C)

单个attention head只能学一种匹配, 多头让模型在不同子空间学不同关系

1.4 Layer norm　　layer-norm(x) = $\frac{x-\text{mean}(x)}{\sqrt{\text{variance}(x)+\varepsilon}}$ · $\gamma+\beta$　　learnable para

(过完MHA和feed-forward都有)

希望层内，the scale of activations remain the same about the input　　ε小值

↓

多层后仍如此　　如果不这样，若有层扩大/缩小 activation scales，会出现 saturation of softmax ⇒ vanishing gradient

Batch norm：对每个特征通道C单独算. 对每个c，收集 B×T 来做 norm

Layer norm：该样本内 C和T，用 CxT 来做 norm

| | BN | LN |
|---|---|---|
| Batch size | 对BC强依赖 | 不依赖 |
| 不同长度 sequence | 有问题，padding会干扰 | 独立处理每个位置，padding不污染整个batch统计量 |
| 训/测统一性 | 训练用batch统计量<br>测试用全局滑动平均<br>存在不一致 | 训测一样 |

不同activation之间尺度差异变小，softmax不易saturate

1.5 RC　　$x = MHA(x) + x$　　$x = MLP(x) + x$　只需学Δ

$MLP(x) = W_2\,\text{GeLU}(W_1 x + b_1) + b_2$

1.6 Training　　Raw Text ⟶ (tokenizer/encoder) Token indices ⟶ binary files ⟶ (采batch 输入x 输出y)

→ next token prediction loss

$x = [w_{1:T}]$　　$y = [w_{1:T+1}]$　　$P_\theta = (w_{t+1} \mid w_{\le t})$

$$L(\theta) = -\sum_{t=1}^{T} \log P_\theta(w_{t+1} \mid w_{\le t})$$

1.7 Inference：生成慢　　$idx_{cond} \in R^{B \times \text{block\_size}}$　　$\text{logits} \in R^{|B| \times |V|}$

logits → softmax → prob → sample next token

把新token拼回序列重复生成

低效：① 自回归得一个token一个生成

② 若无KV cache，则每生成一个新token都要重新生成所有历史token

Lec 12

1. 定义

Pretraing: 大量通用学

Mid traing: 特定高质量继续训练

Post traing/alignment: base model 转成遵循指令、符合人类偏好

2. Pretrain

2.1 Why? ① Transferability across task

② Improved task performance: 相比重头训，pretrain 把参数带到更平滑、有利的 loss landscape 尤其在数据少时

2.2 Impact factors ① Model Architecture 决定表达能力和 scailability 重要但固定

② Training Object

③ Data (数、质、范围) 很重要

④ Hyper parameters ($\eta$, BS) 大 data 下不那么重要

2.3 Data filtering deduplication $\Rightarrow$ Acc↑

避免反复看到重复，浪费容量，增加 memorization 风险

2.4 Objective

2.4.1 MLM: 把输入部分词替换为 Mask，model 完型填空。 没有 causal mask 只用 encoder 部分

$$h_{1:T} = \text{Encoder}(w_{1:T}) \qquad z_i = Ah_i + b$$

loss 只加在有 mask 位置

$$Z_{mask} = -\sum_{i \in Mask} \log p_\theta(w_i \mid \tilde{x})$$

$\tilde{x}$: mask version of $x$

随机选 15% 预测，80% 换成 [MASK]，10% random token，10% 原词不变

SFT 时没有 [MASK]，若 pre-train 老看到 [Mask] 会 mismatch

输入 $e_i = e_i^{token} + e_i^{segment} + e_i^{position}$

[CLS] [SEP] 分句

只用 decoder 部分 sentence A/B

2.4.2 Autoregressive 输入 $[w_{1:T}] \Rightarrow$ 得到每个位置 $[h_{1:T}]$ 拿 $h_t$ 来 linear+softmax

$$h_{1:T} = f(w_{1:T})$$

$$z_t = Ah_{t-1} + b$$

$$\min_\theta \sum_{x \in Dtrain} \sum_t -\log p_\theta(w_t \mid w_{<t})$$

pros: ① 目标统一-简单

② 训练与推理一致

③ Scailing behaviour 强

cons: ① 只能用 left-to-right context

② 自回归生成慢

③ error accumulation

下游任务：任务转文本，再输出加 linear layer

Lecture 13

Pretrained LM 能继续补全，但还不够多 不编

希望 LLM helpful harmness honest

只续写，不回答问题

1. SFT { single task：只训练特定任务，泛化弱
multi task / instruction tuning：多任务混合，让model学会服从指令

1.1 数据来源 { 早期：人工标注. pros：①高质②不依赖 teacher model ③避免 self-generated error amplification
cons：①慢 ②贵 ③多样性有限 ④规模有限
LIMA：Less is more for Alignment 不需那么多
后期：AI-generated SFT data：self instruct
少量 seed tasks → LLM生成新 instructions → 判断 task是否需 input → filtering
→ task pool 继续迭代
当 teacher model 足够强，人工标注 → 机器标注

数据格式 { Alpaca format single-run instruction / input / output
GPT format multi-run conversations：role + value
ChatML format role-based chat system / user / assistant

1.2 Loss $D=\{(x_i,y_i)\}$ $\hat{\theta} \leftarrow \arg\min_{\theta} \mathcal{L}(\theta) = \sum_{i \in D} \sum_{j=1}^{m} -\log(P_\theta[y_{i,j} \mid x_i, y_{i,1:j-1}])$

Do NOT Count X in the loss using a loss Mask

prompt loss weight { 0 → 只学怎么回答
0.1 → 保留少量语言建模信号，缓解 catastrophic forgetting

1.3 效果 SFT 减少 hallucination most effective 甚至比 PPO 更直接

更 obey 指令 helpful, harmless honest

1.4 Industrial SFT

SWE-lego { 输入：repo + bug fixing task + system prompt
output：bug-fixing trajectory

read files → edit files → run tests → observe execution feedback → refine edits

Synthetic data curation { LLM rewrite
AST reformulation 生成 bug

1.5 Multi-task SFT 不是越混越好 三类能力 general / math / coding

混太多 ⇒ 已有能力↓

小数量是好的

Dual stage { 1. 学习强化 specific ability
2. 混少量 general data 避免遗忘，保留泛化

Lec 14 LoRA QLoRA

Full FT —太贵→ PEFT → LoRA → QLoRA (低维子空间) 4bit-base LoRA

1. 参数量 1byte = 8bits

对一个 16-bit model Full FT

weight 16 bits fp16

gradient 16 bits

65B = $65\times10^9$ x 12 bytes = 780GB

Adam first momentum 32bits fp32

Adam second momentum 32bits fp32 ⇒ 为ensure stability of historical data

96bits = 12bytes

$m_t = \beta_1 m_{t-1} + (1-\beta_t) g_t$

$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$

$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

$\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon}$

2. PEFT：只更新一小部分参数 {① Parameter Overload：不需改变太多

② Intrinsic dim：特定任务在低维子空间：pretrain已把空间组织好

$\theta^D = \theta_0^D + \theta^d M$ ← 低维子空间 固定随机投影矩阵 $M = HG\Pi HB$

① 相同 base model，不同任务需不同 intrinsic dimension 能达到 Full FT 90% performa

② pretrain step ↑ ⇒ intrinsic dim ↓ 本来就厉害 最小 d

③ #params ↑ ⇒ intrinsic dim ↓

④ intrinsic dim ↑ ⇒ 多任务表现更差

3. Intrinsic Dim ⇒ Low Rank 参数更新 ΔW 形状大，但 rank 小

⇓ $rank(\Delta W) = r \ll \min(d,k)$

4. LoRA $W = W_0 + \Delta W = W_0 + BA$ $d\times k$

$h = W_0 x + BAx$ $B \in R^{d\times r}$ $A \in R^{r\times k}$

只用训练 $r(d+k)$ 参数 $\ll dk$

pros：① 一个backbone 可以根据 对应多个任务的 Adapter

② No inference latency $W' = W_0 + BA$ 无延迟，无新增参数

4.1 Initialization $A \sim N(0,\sigma^2)$ 打破对称，利rank $B = 0$ 不能全为0，会因为对称更新有问题

一开始 $\Delta W = BA = 0$ 完全等效 base model

不会因随机 Adapter 破坏原始能力

frozen weight 不算 GD了，但仍占显存

5. QLoRA：+ Quantization

5.1 量化 $X^{int8} = round\left(\frac{127}{absmax(X^{FP32})} X^{FP32}\right) = round(c^{FP32} \cdot X^{FP32})$

标准 INT4 有16个level，但均匀分布，LLM权重集中在0附近，中心高密度，但标准下误差大

$E[error] = \sum_v P(v) \cdot error(v, quan(v))$

NF4 更优，重新设计 16 level，每个 bucket 覆盖 1/16 的权重，减少 E[error]

5.2 Paged Optimizer 长序列训练，activation，gradient，optimizer state 会导致 VRAM↑

不够多把部分挪去 CPU RAM，用时再拿回来

Lec 15

1. SFT局限：Loss是最大化 reference answer的 token likelihood. 但开放性问题需要比较多个候选回答质量. 可能两词都非 answer. 但一个靠近，一个远离但 penalty 相同

2. Reward function: measure the quality of multiple open-ended answers

$R(s) \in \mathbb{R}$ $\max_\theta E_{\hat{s}\sim P_\theta(s)}[R(\hat{s})]$ 高reward的回答更应该被频繁生成

3. Policy Gradient：reward不可微

$\nabla f(x) = f(x)$

$\theta_{t+1} := \theta_t + \alpha \nabla_{\theta_t} E_{\hat{s}\sim P_{\theta_t}(s)}[R(\hat{s})]$ ↑与θ无关 $\nabla \log f(x) = \frac{\nabla f(x)}{f(x)} \Rightarrow f(x)\nabla \log f(x)$

$\nabla_\theta E_{\hat{s}\sim P_\theta(s)}[R(\hat{s})] = \nabla_\theta \sum_s R(s) P_\theta(s) = \sum_s R(s) \nabla_\theta P_\theta(s) = \sum_s R(s) P_\theta(s) \nabla_\theta \log P_\theta(s)$

$= \sum_s P_\theta(s) \boxed{R(s) \nabla_\theta \log P_\theta(s)} = E_{\hat{s}\sim P_\theta(s)}[R(\hat{s}) \nabla_\theta \log P_\theta(\hat{s})]$

用MC估计 $\approx \frac{1}{m} \sum_{i=1}^{m} R(s_i) \nabla_\theta \log P_\theta(s_i)$

↑采样数 ↑reward 增加该序列生成概率的方向

只需对 $\log P_\theta(s_i)$ 求导 鼓励|抑制

$\log P_\theta(s_i) = \sum_{t=1}^{T} \log P_\theta(w_i^t | W_i^{<t})$ 本来是Π，log后变累加

4. 为什么用 pairwise comparison？

4.1 R(s)来源：人类回答 { 贵、慢；absolute scoring模糊用：7和8边界不清；annotator calibration不一致，有人严格，有人宽松

pairwise更可靠：两回答，人判断哪个好，相对判断比绝对评分稳定，inter-annotator agreement更高

4.2 Bradly-Terry Model

若每个回答有一潜在强度 $r(s) \in \mathbb{R}$ $r(s) = RM_\phi(s)$

$P(i \succ j) = \frac{\exp(r_i)}{\exp(r_i)+\exp(r_j)} \Leftrightarrow P(i>j) = \sigma(r_i - r_j) = \frac{1}{1+\exp(r_j - r_i)} = \frac{\exp(r_i)}{\exp(r_i)+\exp(r_j)}$

偏好只依赖 $r_i - r_j$，不依赖绝对分数

4.3 Reward Model Loss Train Reward Model 和要训练的LLM无关

$J_{RM}(\phi) = -E_{(s^w, s^l)\sim D}[\log \sigma(RM_\phi(s^w) - RM_\phi(s^l))]$ ⇒ 训练目标：让 $RM_\phi(s^w)$ 尽可能大于 $RM_\phi(s^l)$

$s^w$: winner $s^l$: loser 模型认为winner胜过loser的prob

4.4 Reward Hacking：利用Reward Model的漏洞

4.5 RLHF Objective with KL Penalty 希望最大化的

$R(s) = RM_\phi(s) - \beta \log\left(\frac{P_\theta^{RL}(s)}{P_\theta^{PT}(s)}\right)$ ↙KL penalty

↑R ↑Original R(frozen) PT是pretrain(frozen) RL是正在训练的policy model

Reward Model鼓励模型符合人类偏好，KL penalty让输出接近 $P^{PT}(s)$

Lec 16

动机：朴素 policy gradient 可以优化 reward，但 policy network 对参数更新很敏感
小更新可能带来分布巨大变化，造成 policy collapse

$E \log X \le \log E(X)$

1. KL Divergence $\|\theta_{new} - \theta_{old}\|$ 不可靠 相同参数变化，不同区域可能导致完全不同的输出分布变化

$D_{KL}(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$ P是旧 Q是新 $D_{KL}(P\|Q) \ge 0$ 当且仅当 $P=Q$ 时为

$\sum p \log \frac{p}{q}$ $D_{KL}(P\|Q) \ne D_{KL}(Q\|P)$

2. Natural Policy Gradient

目标是在 trust region 更新参数 $\log x \le x-1$ $\log(1+x)$ $\frac{1}{x+1}$ $-\sum p \log \frac{q}{p} \ge \log E\, q$

$$\max_d V(\theta+d) \quad \text{s.t. } D_{KL}(\pi_\theta \| \pi_{\theta+d}) \le \varepsilon$$

采精确 KL 约束难解 对 V 做一阶近似 $D_{KL}$ 做二阶近似

$$V(\theta+d) \approx V(\theta) + \nabla_\theta V(\theta)^T d$$

$$D_{KL}(\pi_\theta \| \pi_{\theta+d}) \approx \frac{1}{2} d^T F d$$

$$D_{KL} \approx \underbrace{D_{KL}(\pi_\theta \| \pi_\theta)}_{=0} + (\nabla_d D_{KL})^T d + \frac{1}{2} d^T (\nabla_d^2 D_{KL}) d = \frac{1}{2} d^T F d$$

正好取最值 导数为0

FIM Fisher Information Matrix

FIM 半正定且对称 $d^T F d \ge 0 \ \forall d$

⇓

变成 $\max_d \nabla_\theta V(\theta)^T d$ 尽可能大 $\text{s.t. } \frac{1}{2} d^T F d \le \varepsilon$

开拉 $L(d, \lambda) = \nabla_\theta V(\theta)^T d - \lambda(\frac{1}{2} d^T F d - \varepsilon)$ 令 $\lambda > 0$ 若满足，则会大。不满足，会小

$\nabla_d L(d,\lambda) = \nabla_\theta V(\theta) - \lambda F d = 0$

令 $d = \alpha F^{-1} g$ $\alpha = \frac{1}{\lambda}$

~~不妨令 $d = \alpha F^{-1} g$ 这样~~

Natural Gradient

$$d = \frac{1}{\lambda} \boxed{F^{-1} \nabla_\theta V(\theta)} = \theta_{new} - \theta_{old}$$

Step $F^T = F$

又有 $\frac{1}{2} d^T F d - \varepsilon = 0 \Rightarrow \frac{1}{2} \alpha^2 (F^{-1} g)^T F (F^{-1} g) = \frac{1}{2} \alpha^2 g^T F^{-1} g = \varepsilon$

所以 $\alpha = \sqrt{\frac{2\varepsilon}{g^T F^{-1} g}}$ $\Rightarrow d = \sqrt{\frac{2\varepsilon}{g^T F^{-1} g}} F^{-1} g$ 数

步长非普通 learning rate，而是由 trust region 和 Fisher metric 自动决定

3. TNPG 与 TRPO ($F^{-1}$ $O(d^3)$ d could be millions)

3.1 Truncated natural policy gradient (把 $v = F^{-1} g \Leftrightarrow Fv = g$ 用 conjugate gradient 迭代求解)

Trust Region

3.2 TRPO = TNPG + line search

① 先算 Natural Gradient 更新方向 ② 尝试一大步，是否改善，是否小于 KL threshold ③ 不安全缩小步长
避免显示求逆，仍属二阶优化。需多次 backward Hessian-product. 计算仍贵 重试
multiple iterative backward pass

4. PPO Proximal Policy Optimization

不用显式二阶优化，不用 conjugate gradient. 用 Adam, 但用 clipping 限制新旧策略比近似实现 safe update

$\pi(a|s)$ or $\pi_{old}$ : old policy $\pi'(a|s)$ or $\pi_{new}$ : new policy

Probability ratio $r_{sa}(\pi') = \frac{\pi'(a|s)}{\pi(a|s)}$ $r_{sA} > 1$ 在新 policy 下 a is more likely

Advantage $A(s,a) = Q(s,A) - V(s)$ MC rollouts

$A > 0$ action A 比平均好 $A < 0$ 比平均差

Goal max $r_{sA} \times A$

$$L^{CLIP}(\theta) = \hat{E}\left[\min\left(r(\theta)\tilde{A}, clip(r(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}\right)\right]$$

$$clip(r(\theta), 1-\varepsilon, 1+\varepsilon) = \begin{cases} 1-\varepsilon, & r(\theta) < 1-\varepsilon \\ r(\theta), & 1-\varepsilon \le r(\theta) \le 1+\varepsilon \\ 1+\varepsilon, & r(\theta) \ge 1+\varepsilon \end{cases}$$

NLP Lec 17

1. Recap $R(s) = RM_\phi(s) - \beta \log \frac{P_\theta^{RL}(s)}{P^{PT}(s)}$

PPO advantage - actor - critic (AC2) 要rollout

↑ identify true value of a rollout ↑ Policy LM ↑ Value model 相同架构，但 delayed param 用来算 advantage

需维护 critic, ref model, reward model

GAE 估计 advantage

2. DPO motivation 希望 model 符合 preference data，能否跳过 reward model + RL 直接优化 model?

closed-form solution to the reward function for simple MLE optimization

把 reward function → policy 概率比

2.1 从 RLHF 推 DPO

Objective $E_{y\sim P_\theta^{RL}(y|x)}\left[RM_\phi(x,y) - \beta \log \frac{P_\theta^{RL}(y|x)}{P_\theta^{RT}(y|x)}\right]$

令 $\pi(y) = P_\theta^{RL}(y|x)$ $J(\pi) = \sum_y \pi(y)\left[RM(x,y) - \beta \log \frac{\pi(y)}{P^{PT}(y|x)}\right]$

构造拉 $\sum_y \pi(y) = 1$

$$L(\pi,\lambda) = \sum_y \pi(y)\left[RM(x,y) - \beta \log \frac{\pi(y)}{P^{PT}(y|x)}\right] + \lambda\left(1 - \sum_y \pi(y)\right)$$

对 $\pi(y)$ 求偏导 只对其中一个 $\frac{\partial L}{\partial \pi(y)} = RM(x,y) - \beta\left(\log \frac{\pi(y)}{P^{PT}(y|x)} + 1\right) - \lambda = 0$

$$\log \frac{\pi^*(y)}{P^{PT}(y)} = \frac{1}{\beta} RM(x,y) - \frac{\beta+\lambda}{\beta}$$

$$\pi^*(y) = P^{PT}(y)\exp\left(\frac{1}{\beta} RM(x,y)\right)\exp\left(-\frac{\beta+\lambda}{\beta}\right)$$ 每一项都有

theoretical optimal → $p^*(y|x) = \frac{1}{Z(x)} P^{PT}(y)\exp\left(\frac{1}{\beta} RM(x,y)\right)$

令 $Z(x) = \sum_y P^{PT}(y|x)\exp\left(\frac{1}{\beta} RM(x,y)\right)$ ↑ 按 reward 指数求和

$Z(x)$ 要对所有求和，无法计算

2.2 用 Policy 反推 Reward Model

$$p^*(y|x) = \frac{1}{Z(x)} P^{PT}(y|x)\exp\left(\frac{1}{\beta} RM(x,y)\right)$$

取 log $\log p^*(y|x) = -\log(Z(x)) + \log P^{PT}(y|x) + \frac{1}{\beta} RM(x,y)$

$$\Rightarrow RM(x,y) = \beta \log p^*(y|x) + \beta \log(Z(x)) - \beta \log P^{PT}(y|x)$$

定义 $RM_\theta(x,y) = \beta \log \frac{P^*(y|x)}{P^{PT}(y|x)} + \beta \log(Z(x))$

DPO 定义一个由 policy 隐含表示的 reward model

但仍有 $Z(x)$，算不了。用 Bradley-Terry 消掉

Reward Model Loss $P(y_w \succ y_l|x) = \sigma(r(x,y_w) - r(x,y_l))$

代入 $\text{Loss} = -\log \sigma(r(x,y_w) - r(x,y_l))$

$$RM_\theta(x,y_w) - RM_\theta(x,y_l) = \beta \log \frac{P_\theta(y_w|x)}{P^{PT}(y_w|x)} + \beta \log(Z(x)) - \beta \log \frac{P_\theta(y_l|x)}{P^{PT}(y_l|x)} - \beta \log(Z(x))$$

$$= \beta \log \frac{P_\theta(y_w|x)}{P^{PT}(y_w|x)} - \beta \log \frac{P_\theta(y_l|x)}{P^{PT}(y_l|x)}$$

$$\mathcal{L}_{DPO}(\theta) = -E_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{P_\theta(y_w|x)}{P^{PT}(y_w|x)} - \beta \log \frac{P_\theta(y_l|x)}{P^{PT}(y_l|x)}\right)\right]$$

pairwise 只关心 reward difference

对于 $y_w$，让model相比原来更愿意生成

$y_l$ 不愿意生成

增大两者差值

不用显式训 Reward Model，不用PPO rollout. Reference model 还在，提供不要离原来太远的基准

开源友好. 轻量. 简洁 便宜，plug-and-play. 对一般 alignment 很有效

PPO仍有更强全局探索，性能上限更高. 适合复杂. 长文本

3. GRPO：去掉 Critic 的 RL，让同一个prompt生成的多个回答互相比，用 group baseline 算 advantage

对同一问题 $q$，生成 $G$ 个回答 $o_{1:G}$（policy）. 对应 reward：$r_{1:G}$

$$A_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$$

同一个query的相对分数

3.1 GRPO Loss token-level ratio（$i$ 是第 $i$ 个回答第 $t$ token）

$$r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}$$

$$L_{clip} = \min\left[r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_{i,t}\right]$$

$$J_{GRPO}(\theta) = E_{q,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\frac{1}{T_i}\sum_{t=1}^{T_i} \min(r_{i,t}(\theta)A_i, \text{clip}(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)A_i) - \beta D_{KL}\right]\right.$$

$T_i$：第 $i$ 个长度

4. Extensions

4.1 KTO：解决DPO对 paired preference 依赖. DPO 要一个 prompt 下有 winner, loser

但现实更多是好|不好，很少有配对 只要 desired / undesired

4.2 SimPO：解决DPO的长度偏置 $P^{PT}$呢？ 更依赖SFT初始化，算起来多 去掉了 reference model

DPO 问题：序列越长，token-level log prob 累积差异越大，即使每个 token 优势很小

长回答可能因累计效应扩大 winner-loser 的 reward gap

reward margin 保证两者间有严格差距

输出length

$$r_{SimPO}(x,y) = \frac{\beta}{|y|}\log \pi_\theta(y|x) \qquad \mathcal{L}_{SimPO} = -\log \sigma(r_{SimPO}(x,y_w) - r_{SimPO}(x,y_l) - \gamma)$$

4.3 DAPO解决 GRPO的 clipping 抑制探索问题

GRPO | PPO 的 clipping 对称 $(1-\varepsilon, 1+\varepsilon)$

对低概率高 reward 的探索 token 上界太限制 exploration

DAPO：给低概率但正advantage更高 upper bound

$\Rightarrow$ $(1-\varepsilon_{low}, 1+\varepsilon_{high})$

NLP Lec 18 Synthetic Data

1. 为什么需要
   - ①真实数据增长慢，但大模型消耗数据越来越快
   - ②重复同一批数据并不等价于新数据
   - ③合成数据可降 hallucination
   - ④真实 instruction data 可能质量低
   - ⑤合成数据可用于 evaluation

2. Synthetic Data 类型

self-instruct：服务 instruction following； code data：可通过 execution 自动验证

math data 关注 推理过程/答案　　preference data 服务 RLHF/DPO/RM/Alignment

3. Prompting a teacher model

$x \xrightarrow{\text{Teacher LLM}} (x, y)$

cons：teacher 错，student 错

4. Retrieve and Transform

teacher model 生成的缺少 grounding，易产生不真实样本

Corpus $\xrightarrow{\text{Retrieve}}$ $z$ $\xrightarrow{\text{Transform}}$ $(x, y)$

已有真实 data　　检索到　　再把 z 改写成目标格式

pros：grounded in real knowledge sources　hallucination ↓

cons：quality heavily depends on retrieval quality

5. Extract and Rewrite

Web 上有 good data，but polluted by 广告，导航，无关文本。

Raw Web → Extract useful Passage → Score/Filter → Rewrite into IFT Format

Pros：standaridize formatting

cons：依赖 rules/filters，且若用 LLM filter，很贵

6. Reparsing：不引入新知识，改表达形式

$d \xrightarrow{\text{Rephrase Model}} \{d_1', d_2', \ldots, d_k'\}$　一个原始多改写

pros：保留原知识，halluciation ↓

cons：LLM 改写引入自己风格，tone 更统一，降低多样性

7. knowledge graph extraction：data 少/私有/offline　从小规模文档构建知识图谱，再生成文本

subject 关系 客体实体 (S, P, O)

pros：小 corpus → 大 synthetic corpus

cons：relation extraction 错误/LLM 误解关系 → 数据不真实

8. AI Rating/Hybrid Preference：human 准贵，AI 便宜不准

$r(x) \in \{\text{Human}, \text{AI}\}$　把难给人干，简单 AI 干

9. Self-Instruct：Teacher LLM 贵 → 让模型从少量人工 seed 自动扩展任务

生成 instruction 和 instance

$S_0 \rightarrow G(S_0) \rightarrow S_1 \rightarrow G(S_1) \rightarrow \cdots$

↑ 人工 Seed Task

$S_t$ := t轮后任务池

pros: ① instruction data ↑ ② cons: ① seed 决定质量上限 ② 生成任务可能表面多样，实则单-

10. Self-guide：在 self-instruct 上加结构化示例/过滤

input generation → input filter → output generation → input-output pair filter

通过 task sample 控制数据生成，减少随机低质输出

cons: 示例本身 incomplete / bias

11. Evol-Instruct：从 instruct 出发，迭代 t 次

如 In-Breadth Evolving，Deepening，加限制，加推理 ...

生成复杂、任务。 复杂 $\stackrel{?}{=}$ 好

12. Multi-agent Methods：多 agent 互评、修订、过滤。

pros: 减少单模型误差 cons: 互相强化错误，贵

13. Evaluation

13.1 Correctness：LLM 执行代码，比 GT ... ⇒ {0,1} 满意？

13.2 Complexity 例 $\alpha + \alpha s(x) + \beta q(x) + \gamma k(x)$

↑ 推理步数 ↑ 约束 ↑ 知识深度

简单 ⇒ 学不会难的

难 ⇒ 不现实

13.3 Diversity $D = \{x_i : n\}$ ① $\text{self-BLEU}(D) = \frac{1}{n}\sum_{i=1}^{n} \text{BLEU}(x_i, D \setminus \{x_i\})$

低 ⇒ 多样性↑

② Similarity Matrix $K \in R^{n \times n}$ $k_{ij} = \text{sim}(x_i, x_j)$ 对 K 做谱分解，用特征值分布的 entropy 衡量“有效独特样本”，谱集中 ⇒ diversity 低

③ Diversity Greedy Selection $\max_{x_j \in S} \text{sim}(x_i, x_j) < r$ 加入 $x_i$

13.4 Fidelity：是否忠于原始事实或 transformation 前句子

一致性 RoBERTa-MNLI $(x, x') < 0.75$ ⇒ 忽略

14. 局限
① Model Collapse 丢失细节，保留模板表达
② lack of data provence 合成数据吸收进参数后，后续犯错难以追踪来源
③ data leakage 复杂 rephrasing 难查

# NLP Lec 19 Scaling Law

## 1. Scale

N：参数量　D：训练token数　C：训练计算量　in 总FLOPs

希望平衡 N, D, C

## 2. FLOPs：FLOP floating-point-operation 加减乘除都算

transformer：以矩乘为主，忽略 bias, Layer Norm, residual, nonlinearity, softmax

$mxn\ R^n$　$R^m$　output

$Ax \Rightarrow 2mn$　1个dot要算 n乘 n加

m个　$\Rightarrow \underline{\underline{2mn}}$

$mxn\ nxp$

$AB \in R^{mxp}$

mxp output

一个output要 n乘 n加 $\Rightarrow 2mnp$

训练成本 $C \approx 6ND$

2n

每个token前传 每个参数乘加一次　D token　$\approx 2N \Rightarrow 2ND$

反传约为前传两倍

（一个算 dW，一个算 dX）　$\approx 4N \Rightarrow 4ND$

$\Downarrow$

$6ND$

$N$ { QKV+投影 $4 \times D \times D = 4D^2$

FFN（中间维度4D）$D \times 4D + 4D \times D = 8D^2$ } 一层 $\Rightarrow 12D^2$　$N = 12LD^2$

$N = 12 \times n_{layers} \times d_{model}^2 + (n_{vocab} + n_{pos}) \times d_{model}$　embedding的，若pos embed可学

$$Duration = \frac{C}{cluster\ throughout}$$

$10^3$

忽略 softmax, GELU, layernorm 等

GPU通信占 15%-30%时长

只能到 GPU性能 30%-60%

## 3. Kaplan Scaling Law：pretrained cross-entropy loss 随 scale 呈 power-law 下降

大模型通常 data-efficient

$L(C) = AC^{-\alpha} + L_\infty$　比例常数　无限扩展仍无法消除的loss

⇒ 更大模型有更陡 loss curve，同样data下更能有效提取模式，达到更低最终loss

局限：易被误解成 model size 越大越好，但未考虑固定compute下数据量不足的问题

## 4. Chinchilla Scaling Law：在固定 compute 下，很多旧大 model 是 under-trained：参数太多，数据太少

发现小model+大数据能在相同compute下更强

$C \approx 6ND \Rightarrow D \approx \frac{C}{6N}$　对每个C，找 loss 最低的 $N^*$ 和 $D^*$

$N_{opt} \propto C^a$ 和 $D_{opt} \propto C^b$　近似同步增大　a, b 接近 0.5　而 Kaplan $N_{opt} \propto C^{0.73}$　$D_{opt} \propto C^{0.27}$ 更偏参数

5. Variants
① batch size和lr也有scaling law. C↑⇒BS*↑, lr*↓. 可用小规模实验子预测大规模训练超参
② data 质量影响scaling. 少高质一样能打
③ 多模态也可能存在SL

6. Limitation
① pretrain loss ≠ 下游表现
② inverse scaling：模型越大某些任务越差：喜欢记而不跟从指令；模仿数据，挑简单任务做；被正面但不充分示例带偏
③ data filtering
- 小compute下 干净小数据更好
- 大compute下 加一些noisy有利

# NLP Lec 20 KV-cache

1. Naive Decoding 浪费

$$P(x_{1:T}) = \prod_{t=1}^{T} p(x_t \mid x_{<t})$$

每一步生成新 tokens 时都要对当前上下文做 self-attention

第 t 步生成时，历史 token KV 已算 $\sum_{t=1}^{n} O(t) = O(n^2)$

2. KV cache 用显存换时间：历史 token KV 不会因新 token 带来而改变，只存，不算

由于因果索引，无法看到未来的

At time step t $k_t = h_t W_K$, $V_t = h_t W_V$, $q_t = h_t W_Q$

然后把新的 $k_t, V_t$ append 到 cache

$$K_{1:t} = [K_{1:t-1}; k_t] \quad V_{1:t} = [V_{1:t-1}; V_t]$$

生成第 t 个 token 时

$$O_t = \mathrm{softmax}\left(\frac{q_t K_{1:t}^T}{\sqrt{d_k}}\right) V_{1:t}$$

$O(n^2) \to O(n)$

3. Prefill vs Decode

3.1 Prefill：处理输入 Prompt

$$X_{prompt} \longrightarrow K_{prompt}, V_{prompt}$$

可以并行，可以跨 token 运算 瓶颈是 compute-bound 指标 TTFT

但不能跨层并行，l 层依赖 l-1 层

Time to First Token

3.2 Decode 逐 token 生成 $x_t \sim p(x_t \mid x_{<t})$ 并行：低，GPU 利用低

指标 Time Per Output Token 瓶颈：memory-bandwith-bound

每步都从 HBM 读 KV Cache

4. KV Cache 大小于 FP16 下

K, V

$$\text{Size} = B \times L \times 2 \times N_{layers} \times N_{heads} \times d_{head} \times 2 \text{ bytes}$$

个 FP16

size $\propto B\, L\, N_{layers}\, N_{heads}\, d_{head}$ 长文档任务 OOM 元凶是 KV Cache

5. Flash Attention：减少 attention 中间矩阵写回

$$A = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

A 很大，读写慢。

不存完整 A，而是 block by block 算，在线维护 softmax 归一化项

对于 logits $m_i = q^T k_i$ softmax 权重为 $w_i = \frac{\exp(m_i - m^*)}{\sum_j \exp(m_j - m^*)}$ $m^* = \max_j m_j$

输出 $s = \sum_i w_i v_i$ 在线计算时维护 $d_i = \sum_{j=1}^{i} \exp(m_j - m^*)$

$$s_i = \frac{s_{i-1} d_{i-1} + \exp(m_i - m^*) v_i}{d_i}$$

6. MHA → MQA → GQA：减少 KV head

6.1 MHA　KV cache 随头数线性增长

6.2 MQA　所有 query head 共享同一组 K,V，极大压缩 KV Cache，但表达能力↓

6.3 GQA　把 query head 分成若干组，每组共享 K,V，（每个不同）

压缩比 $\frac{H_{kv}}{H_q}$（每组不同，每个不同）

7. Page Attention
① 每个请求预留最大长度，但有些没用完
② 不同请求物理显存不连续，没用的无法复用

logical block id → (block table) → physical block id

8. Continuous Batching 与 chunked prefill

Prefill　大矩阵，compute-bound
decode　小步读 KV-cache　memory-bound

一个在 prefill，其他 decode 要等

Prefill (L) → 分块 prefill　把 decode 插 prefill 中

9. Streaming LLM 与 Attention Sink

处理上下文问题：
全 KV-cache　VRAM OOM
最近 window：丢掉最初 token，性能崩溃
每次重算　$O(L^2)$

初始几个 token 虽语义信息不强，但稳定吸收大量 attention，形成 attention sink

训练中初始位置被最多看见，softmax 归一化也要某些位置接冗余注意力

Streaming LLM　KV Cache = Sink Tokens + Recent Window Tokens

$$K_{cache} = [K_{sink}; K_{t-w:t}]$$

$$V_{cache} = [V_{sink}; V_{t-w:t}]$$

# NLP Lec 21 Quantization, Pruning, Distillation

## 1. Format

若 cache 容量为 C bytes，每个参数用 $P_{bit}$ bits 能存 $N = \frac{C \times 8}{P_{bit}}$

FP16 (2Bytes) [-65504, 65504]

INT8 1Bytes [-128, 127]

INT4 0.5Bytes [-8, 7]

## 2. Quantization

$$q = \text{clamp}\left(\text{round}\left(\frac{x}{s} + Z\right), q_{min}, q_{max}\right)$$

(高精)

反量化 $\hat{x} = s(q - Z)$

Scale zero-point

### 2.1 Absmax systemmetric quan

$$s = \frac{\max(|x|)}{127}$$ INT8

$[-\max|x|, \max|x|] \rightarrow [-127, 127]$

### 2.2 Zero Point 量

$$s = \frac{\max(x) - \min(x)}{2 \times 127}$$ 或 $$s = \frac{x_{max} - x_{min}}{q_{max} - q_{min}}$$

$$Z = q_{min} - \frac{x_{min}}{s}$$

### 2.3 Precision Cliff : 4-bit好 3bit不行

Performance $\approx f$(# params) $- g$(quantization noise)

## 3. PTQ 与 QAT

PTQ：预训练完直接量化。快、便宜、无需额外训练。低bit易掉点 8bit

QAT：Quantization-Aware Training 训练时模拟量化误差 精度恢复好，贵需数据训练

### 3.1 QAT round(·) 不可导 ⇒ 梯度消失

STE Straight-Through Estimator

FP32

forward 用 round backward 假装 round 是 identity

## 4. Activation-outlier：某些 hidden vector 维度 ≥60，出现在至少 25% 层，至少 6% token中

问题来自 $s = \frac{\max(|x|)}{127}$ 若某 outlier 比普通值大 100 倍 ⇒ 大量非outlier信息被压成0

outlier 虽少，但对 model 重要

$L = \sum_i p_i l_i$ 最小化 L ⇒ 罕见用长的 $l_i$，常见用短

## 5. LLM.int8 (mixed precision quantization) outlier 单独高精度

$XW = X_{normal} W_{normal} + X_{outlier} W_{outliers}$ ⇒ I32 + FP16 ⇒ FP16

INT8 FP16

## 6. SmoothQuant Optimization 把 activation outlier 搬去平滑自己W

$Y = (X \text{diag}(s)^{-1})(\text{diag}(s) W)$ smoothing factor

7. Pruning 删参. unstructured 如删1

7.1 Activation Pruning 与 WANDA

$S_{ij} = |W_{ij}| \cdot \|X_j\|_2$ weight本身大不一定重要. 它连的activation 经常大才重要

7.2 Iterative Pruning 剪低分 blocks → finetune / retrain 恢复性能 → 重复

NXN $R_i = \|W_{block,i}\|_2$ 小的masked zero

8. Distillation { hard label: 正石角类别
soft : 概率分布

带 temperature $p_i^{(T)} = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$

T=1 low entropy

T>1 scales down logit 分布平滑 entropy↑

pros: ①完全不同架构

cons ② 需训 student 且 teacher 得 avaliable

比PTQ贵

9. 2:4

~~2件~~ 2bit

连续4个 变成0

horizontal elements

2 bit存 spatial index

2 X throughput 吞吐

# NLP Lec 22 MoE

一. 两核心组件（Experts 多个子网络，FFN；Gating network/Router 选择）

$$\hat{y} = \sum_{i \in E_k} G(x)_i E_i(x)$$

2. MoE in LSTM

$G = \text{softmax}(W_g X)$ ⇒ 选top-k个experts $I = \{i_0, i_1\} = \text{Topk}(G, k=2)$

⇒ 两专家prob未归一化

$$S_0 = \frac{G_{i_0}}{G_{i_0} + G_{i_1}} \quad S_1 = \frac{G_{i_1}}{G_{i_0} + G_{i_1}}$$

$Y = S_0 FFN_{i_0}(X) + S_1 FFN_{i_1}(X)$

3. MoE in Transformer

Self-Attention → MoE FNN（之前是FNN）

对于token hidden state $h$，router计算

$P = \text{softmax}(W_r h)$ ← router可学习参数

↑ 当前token分配给不同experts prob　← 当前token hidden representation

若选中 k个expert $\text{MoE}(h) = p_k E_k(h)$

↑ 对token representation的FFN变换

4. Routing Algo
   - Maximum：选1个. pros：低参数 cons：routing collapse 少数常用 多数闲置
   - Top-2 | Top-k：Top 2选 $I = \text{Topk}(P, k=2)$ $\text{MoE}(h) = s_i E_{i_1}(h) + s_2 E_{i_2}(h)$
     pros：表达更强，缓解 routing collapse 训练信号分配给多专家
     cons：① FLOPs↑ ② token发到不同device，communication overhead更大

5. GLaM：每token相似在FLOPs MoE优于Dense

6. Deep Seek-MoE：区分 shared experts（每个token必过，通用知识）；routed experts（由router选，专门知识）

每个MoE含 2 shared + 6 routed，每次选 2 shared + 6 routed

token-level

7. MoE Traing：选择Topk不可微

$$L = L_{task} + \lambda L_{aux}$$

主任务　辅助负载均衡损失，鼓励experts均衡用

$$L_{task} = -\frac{1}{T}\sum_{t=1}^{T} \log P_\theta(x_t | x_{<t})$$

$$h_t = u_t + \sum_{i=1}^{N} g_{t,i} FFN_i(u_t)$$ 残差

$$g_{t,i} = \begin{cases} s_{t,i}, & s_{t,i} \in \text{TopK}(\{s_{t,j} | 1 \le j \le N\}, K) \\ 0, & \text{otherwise} \end{cases} \quad s_{t,i} = G(u_t; \Theta)_i$$

不可导数

$$L_{aux} = \alpha N \sum_{i=1}^{N} f_i P_i$$ ← router给expert的prob

↑ token fraction

$$P_i = \frac{1}{T}\sum_{t=1}^{T} s_{t,i}$$

$$f_i = \frac{N}{KT}\sum_{t=1}^{T} \mathbb{I}$$ expert / top k / KT个选择

8. Build MoE LLMs

8.1 Sparse Upcycling copy FFNs to form experts
- 可复用 dense model 的 pretrained knowledg
- 比从0开始训练经济高效
- 参数增加N倍

8.2 Sparse Splitting split FFNs to form
- 激参↓
- expert容量 $\frac{1}{N}$ 单个 expert 表达有限

训

$$\max \sum_t h_t \cdot W_{a_t}$$

$$\text{s.t. } \forall e \sum_{t=0}^{T} 1_{a_t=e} = \frac{T}{E}$$

L23

# 1. Overview

## 1.1 GAN

优化 $\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log(D(x))] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$

对D来说都希望最大化　G希望小　D希望大　假图

z ~ latent noise prior

## 1.2 VAE

$E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)\|p(z))$

从latent z重构x

latent prior $N(0,I)$

希望 $q_\phi(z|x)$ 贴近 $N(0,I)$

encoder: $x \to z$ (latent distribution)

## 1.3 Diffusion: reverse

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$ 其中 $p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t))$

GAN 快. High quality

VAE 快. Mode Coverage/Diversity

Diffusion: High quality + Mode Coverage

# 2. Forward Process

$q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ 或 $q(x_t|x_{t-1}) = N(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I)$

其中 $\alpha_t = 1-\beta_t$

$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}\varepsilon_{t-1}$, $\varepsilon_{t-1} \sim N(0,I)$ 若不用 $\sqrt{\alpha_t}$ 缩放. 连续加噪会导致 variance 不可控

$x_{t-1} = \sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-1}}\varepsilon_{t-2}$ $\varepsilon_{t-2} \sim N(0,I)$

$x_t = \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\varepsilon_{t-2} + \sqrt{1-\alpha_t}\varepsilon_{t-1}$

$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\varepsilon$ $\varepsilon \sim N(0,I)$

两个独立 Gaussian 相加仍是 Gaussian

$N(0,\sigma_1^2I) + N(0,\sigma_2^2I) = N(0,(\sigma_1^2+\sigma_2^2)I)$

$\alpha_t(1-\alpha_{t-1}) + 1 - \alpha_t = 1 - \alpha_t\alpha_{t-1}$

推广到任意 t，定义 $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$

$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$

$q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$

$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$, $\varepsilon \sim N(0,I)$

定义 $\alpha_t = 1-\beta_t$ $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$ $\bar{\alpha}_t$ 控制 $x_0$ 在 $x_t$ 中还有多少信号.

$t\uparrow$ $\bar{\alpha}_t \to 0$ $\Rightarrow$ $x_t \approx \varepsilon \sim N(0,I)$

schedular 控制 $\bar{\alpha}_t$. linear 信号可能太早崩掉. cosine scheduler 会信号留存更久，避免 collapse

3. Reverse Process：从 noise 回 data

目标是学 $p_\theta(x_{t-1}|x_t)$　因为加噪是 infinitesimal ⇒ a reverse transition 建模成高斯

⇒ $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t))$

但真实 reverse posterior $q(x_{t-1}|x_t)$ 不可解

$$q(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})\,q(x_{t-1})}{q(x_t)}$$

对整个数据空间积分 intractable

$$q(x_t) = \int q(x_t|x_{t-1})\,q(x_{t-1})\,dx_{t-1}$$

但如果额外 condition on $x_0$

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)\,q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

高斯　高斯　tractable

$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$

其中 $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t} x_0$

4. 为什么预测 noise $\epsilon$ ⇔ 预测 reverse mean

在 inference 中，由于 $x_0$ 未知，但由于 $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$

可解出 $x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon)$

用 neural network 预测噪音：$\epsilon_\theta(x_t, t) \approx \epsilon$

代入 posterior mean　$\beta_t = 1-\alpha_t$

$$\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t,t)\right)$$

直接 model 不用预测 $x_{t-1}$ 或 $x_0$　只要 predict 加进去的 noise

5. ELBO

目标最大化 $\max_\theta \log p_\theta(x_0)$ ⇔ $\min_\theta -\log p_\theta(x_0)$

$p_\theta(x_0) = \int p_\theta(x_{0:T})\,dx_{1:T}$ 不可算，引入 $q(x_{1:T}|x_0)$

$$-\log p_\theta(x_0) = -\log \int q(x_{1:T}|x_0)\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\,dx_{1:T} \le E_{q(x_{1:T}|x_0)}\left[\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

$L_{VLB}$

$\log E[X] \ge E[\log[X]]$ → Jensen

已经假设给定真实数据 $x_0$

$$L_{VLB} = E_q\left[\log \frac{\prod_{t=1}^T q(x_t|x_{t-1})}{p(x_T)\prod_{t=1}^T p_\theta(x_{t-1}|x_t)}\right]$$

$$= E_q\left[\log \frac{q(x_T|x_0)}{p(x_T)} + \sum_{t=2}^T \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} - \log p_\theta(x_0|x_1)\right]$$

怎么推？ $q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)\,q(x_{t-1}|x_0)}{q(x_t|x_0)}$ ⇔ $q(x_t|x_{t-1}) = \frac{q(x_{t-1}|x_t, x_0)\,q(x_t|x_0)}{q(x_{t-1}|x_0)}$

由马尔可夫，$q(x_t|x_{t-1}) = q(x_t|x_{t-1}, x_0)$

$$L_{VLB} = E_q\left[\log \frac{q(x_T|x_0)\,q(x_{T-1}|x_T, x_0)}{p(x_T)\cdots q(x_{T-1}|x_0)} \cdots\right] = E_q\left[\log \frac{q(x_T|x_0)\prod_{t=1}^T q(x_{t-1}|x_t, x_0)}{p(x_T)\prod_{t=1}^T p_\theta(x_{t-1}|x_t)\times q(x_0|x_0)}\right]$$

上下相消　$q(x_0|x_1, x_0) = 1$

$$= E_q\left[\log \frac{q(x_T|x_0)}{p(x_T)} + \sum_{t=2}^T \log \frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)} - \log p_\theta(x_0|x_1)\right]$$

$$= E_q\left[D_{KL}(q(x_T|x_0)\|p(x_T)) + \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t)) - \log p_\theta(x_0|x_1)\right]$$

诸性质 $q(x_{1:T}|x_0) = q(x_{1:T\setminus\{t,t-1\}}, x_t|x_0)\, q(x_{t-1}|x_t, x_0)$

$E_{q\sim q(x_{1:T}|x_0)} = E_{q(x_{1:T\setminus\{t,t-1\}}, x_t|x_0)}\, E_{q(x_{t-1}|x_t,x_0)} \Rightarrow E_{x_t\sim q(x_t|x_0)}[-D_{KL}(\cdot)]$

两个 Gaussian KL $D_{KL}(N(\mu_1, \sigma_1^2 I)\|N(\mu_2, \sigma_2^2 I)) = \frac{1}{2\sigma^2}\|\mu_1-\mu_2\|_2^2$

$L_{simple} = E_{t,x_0,\varepsilon}[\|\varepsilon - \varepsilon_\theta(x_t, t)\|_2^2]$

$x_t = \sqrt{\bar\alpha_t}\, x_0 + \sqrt{1-\bar\alpha}\,\varepsilon$

6. Training Algo

1. Sample $x_0 \sim$ data
2. Sample $t \sim$ Uniform($\{1, \cdots, T\}$)
3. Sample $\varepsilon \sim N(0, I)$
4. Construct $x_t = \sqrt{\bar\alpha_t}\, x_0 + \sqrt{1-\bar\alpha_t}\,\varepsilon$
5. Predict $\varepsilon_\theta(x_t, t)$
6. Minimize $\|\varepsilon - \varepsilon_\theta(x_t, t)\|$

7. Sample Algo 要么 $\sigma_t^2 = \beta_t$

$\sigma_t^2 = \tilde\beta_t = \frac{1-\bar\alpha_{t-1}}{1-\bar\alpha_t}\beta_t$

1. Sample $x_T \sim N(0, I)$
2. 对 $t = T, T-1, \cdots, 1$:

$x_{t-1} = \mu_\theta(x_t, t) + \sigma_t z$

$z \sim N(0, I)$

其中 $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar\alpha_t}}\varepsilon_\theta(x_t, t)\right)$

↑ 其实是从 ~~$x_0$~~ $x_t$ 没有

$x_t = \sqrt{\bar\alpha_t}\, x_0 + \sqrt{1-\bar\alpha_t}\,\varepsilon$

解 $x_0 = \frac{1}{\sqrt{\bar\alpha_t}}(x_t - \sqrt{1-\bar\alpha_t}\,\varepsilon)$

↑ 用 $\varepsilon_\theta(x_t, t)$ 来代替

7. Diffusion 通常用 UNET

① noisy $x_t$

- downsample: 提 global structure
- bottleneck: 全局语义
- upsampling: 恢复 spatial resolution
- skip-connection: 保留细节

time embedding Emb(t) sin

$= q(x_t|x_{t-1}) = N(x_t; \sqrt{\alpha_t}x_{t-1}, \beta_t I)$, $\beta_t = 1-\alpha_t$

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)\, q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

$\log q(x_{t-1}|x_t, x_0) = \log q(x_t|x_{t-1}) + \log q(x_{t-1}|x_0) - \log q(x_t|x_0) + \text{const}$

代入高斯密度

$\log q(x_t|x_{t-1}) = -\frac{1}{2\beta_t}\|x_t - \sqrt{\alpha_t}\,x_{t-1}\|^2 + \text{const}$

$\log q(x_{t-1}|x_0) = -\frac{1}{2(1-\bar\alpha_{t-1})}\|x_{t-1} - \sqrt{\bar\alpha_{t-1}}\,x_0\|^2 + \text{const}$

$\log q(x_t|x_0) = -\frac{}{2(1-\bar\alpha_{t-1})}\|x_t - \sqrt{\alpha_t\bar\alpha_t}\,x_0\|^2$

$\mu: \sqrt{\alpha_t} \quad \sigma^2 = 1-\bar{\alpha}_t$

$$\log q(x_{t-1}|x_t, x_0) = \log q(x_t|x_{t-1}) + \log q(x_{t-1}|x_0) - \log q(x_t|x_0)$$

只关注 $x_{t-1}$ 的二次项和一次项

$$= -\frac{1}{2\beta_t}\|x_t - \sqrt{\alpha_t}x_{t-1}\|^2 - \frac{1}{2(1-\bar{\alpha}_{t-1})}\|x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0\|^2 + \frac{1}{2(1-\bar{\alpha}_t)}\|x_t - \sqrt{\bar{\alpha}_t}x_0\|^2$$

$$= -\frac{\alpha_t}{2\beta_t}x_{t-1}^2 - \frac{1}{2(1-\bar{\alpha}_{t-1})}x_{t-1}^2 + \frac{\sqrt{\alpha_t}}{\beta_t}x_t x_{t-1} + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_{t-1}x_0$$

$-\frac{1}{2}Ax_{t-1}^2 + B\cdot x_{t-1}$ 型，条件方差为 $A^{-1}$，均值为 $A^{-1}B$

$$= -\frac{1}{2}\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)x_{t-1}^2 + \left(\frac{\sqrt{\alpha_t}}{\beta_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right)x_{t-1}$$

$$\text{Var} = \frac{1}{A} = \frac{1}{\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}} = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{\alpha_t - \alpha_t\bar{\alpha}_{t-1} + \beta_t} = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\alpha_t\bar{\alpha}_{t-1}} = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$

$$\text{均值 } \mu = \frac{\beta_t(1-\bar{\alpha}_{t-1})}{1-\alpha_t\bar{\alpha}_{t-1}}\cdot\left(\frac{\sqrt{\alpha_t}}{\beta_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\right) \quad (1-\alpha_t)$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\beta_t(1-\bar{\alpha}_{t-1})\sqrt{\bar{\alpha}_{t-1}}}{(1-\alpha_t\bar{\alpha}_{t-1})(1-\bar{\alpha}_{t-1})}x_0$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0$$

A　　B

$$-\frac{B}{\sqrt{\alpha_t}} = \frac{\sqrt{\bar{\alpha}_{t-1}}}{-\frac{\beta_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}}$$

将 $x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}}$ 代入

$$\mu_\theta(x_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\cdot\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}}$$

$$\tilde{\mu} = Ax_t + B\cdot\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}}$$

$$= \left(A + \frac{B}{\sqrt{\alpha_t}}\right)x_t - \frac{B\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\epsilon$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}$$

$$= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} + \frac{\beta_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} = \frac{1}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\left(\alpha_t(1-\bar{\alpha}_{t-1}) + \beta_t\right) \quad 1-\alpha_t$$

$$= \frac{1}{1-\bar{\alpha}_t}(1-\alpha_t\bar{\alpha}_{t-1}) = \frac{1-\bar{\alpha}_t}{1-\bar{\alpha}_t}\cdot\frac{1}{\sqrt{\alpha_t}} = \frac{1}{\sqrt{\alpha_t}}$$

$$\Rightarrow \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

# NLP Lec 24 Score-based View 与 Text Diffusion

DDPM: predict noise $\epsilon$ → Score-based $\nabla_x \log p_t(x)$

1. Score function $s(x) = \nabla_x \log p(x)$ 指向 proba density 上升方向最快的向量场

prob density

2. Langevin Dynamics

用 score 来采样

$$x_{k+1} = x_k + \frac{\eta}{2}\nabla_x \log p(x_k) + \sqrt{\eta}\, z_k, \quad z_k \sim N(0,1)$$

Step Size

Score 引导进高密度区

随机扰动力，防止陷入局部 mode

物理直学 含 potential energy $U(x) = -\log p(x)$

带热随机扰动的 GD

3. Score unknown

真实数据分布 $p(x)$ 未知 $\nabla_x \log p(x)$ 不能算

训练神经网络 $S_\theta(x) \approx \nabla_x \log p(x)$

理想 loss $L_{score} = E_{p(x)}[\| S_\theta(x) - \nabla_x \log p(x) \|^2]$

target score unknown

⇓

denoising score matching

4. Denoising Score Matching

对 clean data 加 noise $x_t = x_0 + \sigma\epsilon, \ \epsilon \sim N(0,1)$

$$\Rightarrow q(x_t|x_0) = \frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|x_t - x_0\|^2}{2\sigma^2}\right)$$

取 log

$$\log q(x_t|x_0) = -\frac{1}{2\sigma^2}\|x_t - x_0\|^2 - \log\left((2\pi\sigma^2)^d\right) + \text{const}$$

对 $x_t$ 求梯度 $\nabla_{x_t} \log q(x_t|x_0) = \frac{x_0 - x_t}{\sigma^2}$ unbiased estimate of the gradient of the marginal density

$$= \frac{-\sigma\epsilon}{\sigma^2} = -\frac{\epsilon}{\sigma}$$

$E_{q(x_0|x_t)}[\nabla_{x_t} \log q(x_t|x_0)] = \nabla_{x_t} \log p(x_t)$

Denoising Score Matching $E_{q(x_0,x_t)}[\| S_\theta(x_t) - \nabla_{x_t} \log q(x_t|x_0) \|^2]$

说明预测 score 与 noise 等价 $= E[\| S_\theta(x_t) + \frac{\epsilon}{\sigma} \|^2]$

$S_\theta(x,t) \propto -\frac{1}{\sigma_t}\epsilon_\theta(x_t, t)$

5. 为什么 NLP 需要 diffusion

自回归问题 { 左→右生成 / 前面错只能继续生成 / global constraint 难以提前规划

Diffusion 优势 { whole sequence refinement / 所有 token 同时被反复修改 / 早期确定 global semantics / 后期修复 local syntax

5.1 核心困难 image 是离散存储，但可视为连续变量 $x \in R^{H\times W\times C}$

token 不连续 $w_i \in V$ { token id 间无自然距离 / cat dog 的 id 差距无语义意义 / 对 token id 加 Gaussian Noise 会落到无效 token

不能直接 $w_t = w_0 + \varepsilon$

5.2 Discrete Diffusion：用 transition matrix 替代 Gaussian Token

$Q_t \in R^{|V|\times|V|}$

forward pass: $q(x_t|x_{t-1}) = Cat(x_t; Q_t x_{t-1})$

三种 corruption

Keep 保持原 token，加噪 $Q_t = (1-\beta_t) I + \beta_t U$

Mask / Absorbing diffusion [mask] token 的 one-hot ↑ $Q_t = (1-\beta_t) I + \beta_t e_{mask} 1^T$ token 有概率替换为 mask

Swap / Uniform diffusion $Q_t = (1-\beta_t) I + \frac{\beta_t}{|V|-1}(1 1^T - I)$ 被替换成 V 中其他 token

5.3 Discrete forward posterior

定义累积 transition $\bar{Q}_t = Q_1 Q_2 \cdots Q_t$ ✓ 前乘行向量 x ✓ $x\bar{Q}_t$

forward $q(x_t|x_{t-1}) = Q_t(x_{t-1}, x_t)$ $\bar{Q}_t = Q_t \cdots Q_1$ 若右接列向量 x.

$q(x_t|x_0) = \bar{Q}_t(x_0, x_t)$

$$q(x_{t-1}|x_t, x_0) = \frac{Q_t(x_{t-1}, x_t)\,\bar{Q}_{t-1}(x_0, x_{t-1})}{\bar{Q}_t(x_0, x_t)}$$

训 reverse model $p_\theta(x_{t-1}|x_t, t) \approx q(x_{t-1}|x_t, x_0)$

$L_{KL} = E_{q(x_t,x_0)}[D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)]$

对于 categorical distribution KL-minimization 等价于 cross entropy

$L_{CE} = -E_{q(x_{t-1}|x_t,x_0)}[\log p_\theta(\hat{x}_{t-1} = x_{t-1}|x_t)]$

但 $Q_t \in R^{|V|\times|V|}$ |V| 很大 ⇒ continuous latent diffusion (Diffusion-LM)

5.4 Diffusion LM

把 token → embedding space $x_0 = Emb(w) \in R^{n\times d}$ ↓ n 个

$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$

$L_{simple}(x_0, \theta) = \sum_t E_{q(x_t|x_0)}[\|\mu_\theta(x_t, t) - \tilde{\mu}(x_t, x_0)\|^2]$

rounding 到 token $p_\theta(w|x_0) = \prod_{i=1}^{n} p_\theta(w_i|x_i)$

$L(x_0, w, \theta) = L_{simple}(x_0, \theta) - \log p_\theta(w|x_0)$ ↙ clean embedding → 离散词

5.5 Controlled Generation

目标是生成满足控制条件 c 的文本 $p(x_{0:T}|c)$ c是

在每一步 reverse diffusion 中 $p(x_{t-1}|x_t,c) \propto p(x_{t-1}|x_t)\,p(c|x_{t-1},x_t)$

$= p(c|x_{t-1})$ 条件独立

取 log $\log p(x_{t-1}|x_t,c) = \log p(x_{t-1}|x_t) + \log p(c|x_{t-1}) + \text{const}$

对 $x_{t-1}$ 求梯度 $\nabla_{x_{t-1}} \log p(x_{t-1}|x_t,c) = \nabla_{x_{t-1}} \log p(x_{t-1}|x_t) + \nabla_{x_{t-1}} \log p(c|x_{t-1})$ ↓ classifier

5.6 Classifier Training

给定 labeled dataset $D=\{(w,c)\}$

① ~~训练~~ Sample $(w,c)$ ② $w \mapsto x_0$ ③ Sample $t$ ④ $x_t = \sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$

⑤ classifier predict c from $x_t$ and $t$ ⑥ 最小化 cross entropy

$L_{cls} = -\log p_\phi(c|x_t,t)$

5.7 Block diffusion

普通 discrete diffusion { ① fixed-length gen ② 无法复用 KV-Cache

在 AR 和 diffusion 之间折中 $p_\theta(x) = \prod_{b=1}^{B} p_\theta(x_b|x_{<b})$ 每个 block 内部用 diffusion 并行去噪

Lec 25 Agentic System + planning | memory | tools | feedback

多步执行，记忆历史，调用工具，反思错误，与环境交互

4部分 LLM(brain) Memory Tools Enviroment

1. Context Engineering 优化送入LLM的上下文
   - limited context window
   - lost in the middle
   - 不相关/冲突的 retrived info
   - 其他形式 图等 不能序列化成token

   ⇒ 检索，压缩，结构化
   工具调用，分阶段 reasoning

2. Memory
   - long term：事实，规则，用户偏好；过去事件，交互轨迹，技能，习惯，操作流程，形成的行为模式
   - short term：当前输入和对话，中间状态，计划，变量

3. RAG 解决 outdated knowledge，LLM information source not transparent
   hallucination parametric memory不可靠
   ① Indexing：build indexed database for documents
   ② Retrieval：retrieve top-k relevant chunks
   ③ Generation：insert chunks into LM context and generate answer

4. Reflection：CoT，LLM-as-a-judge
   Prompt → Intermediate Reasoning → Answer

   main LLM-output → judge LLM evaluates quality → score | critique | retry

   变成 agentic system 的 loop 的一个模块

5. Tools

6. Collaboration | MCP 统一模型和工具的接口 传统 N×M（model × service）

   MCP ⇒ N+M