

Lesson Language Modeling

计算文字序列概率。

$\{N\text{-gram} \rightarrow n\text{-order N-gram} \text{文字序列概率}\}$

RNN

Transformers

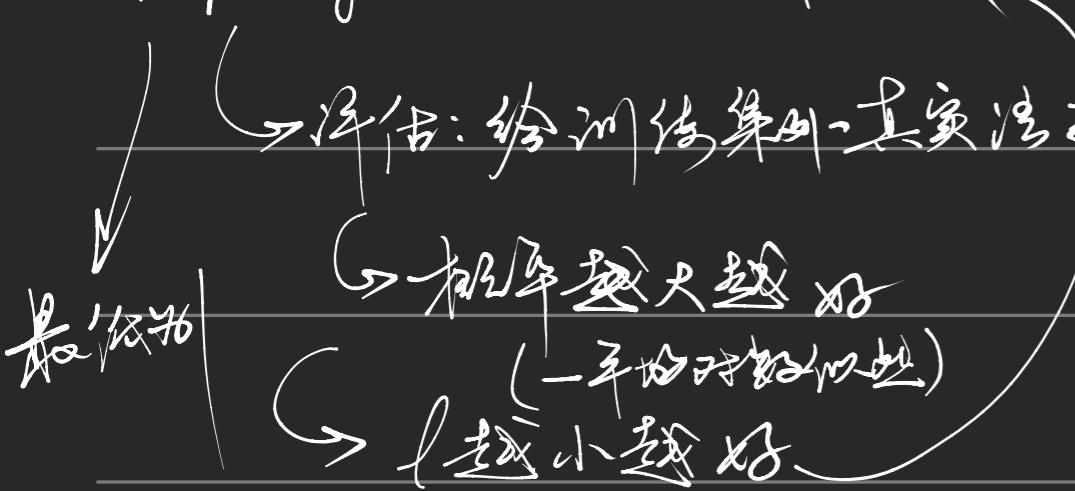
△ Tips: 用 [UNK] 表示所有未在训练语料中出现的词。

△ Tips: 计算方法:

$$l = -\frac{1}{m} \sum_{i=1}^m \log P(\vec{x}_i)$$

(从头到尾逐字计算)
逐个词逐字计算
(共 m 个)

$$\text{Perplexity} = 2^l \rightarrow \text{越小越好}$$



Corner Case:

- ① 词表大小不同，概率不同 \rightarrow 不同词表
- ② 模型不可表示 2^l
- ③ Extreme: 词表只有 [UNK] $\rightarrow 2^l = 1$

Sparseness (稀疏性) 表现: 通常 $N\text{-gram}$ 不足

\Rightarrow 例子 / 极端 = $(\frac{\text{第 } N \text{ 个}}{\text{总词汇量}})$ (因有词无匹配案例)

① Smoothing: (对于第 N 词)

对各种 $N\text{-grams}$ 算法 \propto 第 N 词:

所指词概率 + 其他词概率 P.

② Backoff (对于第 $N\text{-grams}$) \rightarrow 用 $N-2, N-3 \dots \text{-gram}$

③ Linear Interpolation

$$P(w_i | N\text{-gram}) = \lambda_1 P(w_i | N-1\text{-gram}) + \lambda_2 P(w_i | N-2\text{-gram}) + \dots$$

* RNN

$$h^{(t)} = \sigma(W_e e^{(t)} + W_h h^{(t-1)} + b_1)$$

逐字输出 \rightarrow 前状态 \rightarrow logit

$$\text{输出: } \hat{A}^{(t)} = \text{softmax} (\cup h^{(t)} + b_2) \in \mathbb{R}^{|\mathcal{V}|}$$

\rightarrow 无梯度问题

无梯度 ∇ 在 $n\text{-grams}$

[FIX: fixed window NN]:

词 $\vec{e}^{(t)}$ \rightarrow 词嵌入 $\vec{e}^{(t)}$

hidden-layer: $f(\vec{W} \vec{e}^{(t)} + \vec{b}_1)$

$$\hat{A}^{(t)} = \text{softmax} (\cup h + b_2) \in \mathbb{R}^{|\mathcal{V}|}$$

训练 RNN:

逐步计算 $\hat{Y}^{(t)}(\theta)$: 每一步进行预测输出 $\hat{A}^{(t)}$

与损失 $\gamma^{(t)}$ 的交叉熵 = $-\log \hat{Y}_{(t+1)}^{(t)} \rightarrow$ 下一步实际词

$$J(\theta) = \frac{1}{T} \sum \hat{Y}^{(t)}(\theta) \rightarrow$$

* $N\text{-gram}$

$$P(w_i | w_{i-1}, \dots, w_{i-n}) = P(w_i | w_{i-1}, \dots, w_{i-n})$$

$$= \frac{\text{Count}(w_{i-1}, w_i)}{\text{Count}(w_{i-1})} \rightarrow \text{如果 } w_i \text{ 因为 } N\text{-gram} \text{ 而存在}$$

注: <S>, <E> 也是 $N\text{-gram}$ 中

不断更新 $h^{(t)}$

更新：当前时刻 $Y^{(t)}(0)$ 对于历史时刻 $h^{(t')}$ 成等。
 $\frac{\partial Y^{(t)}}{\partial h^{(t')}}$ 为： $\frac{\partial Y^{(t)}}{\partial h^{(t')}} = \frac{\partial h^{(t)}}{\partial h^{(t')}} \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \frac{\partial h^{(t-1)}}{\partial h^{(t-2)}} \dots \frac{\partial h^{(2)}}{\partial h^{(1)}}$

△ Multi-head Attn

$Q \times V \rightarrow$ 最多至 m 个线性空间

for $i : 1 \sim m$:

$$Q_i = W_i^T Q$$

$$K_i = W_i^T K \Rightarrow m\text{-head Attn}$$

$$V_i = W_i^T V$$

* LSTM、GRU、多层、双向循环、

双向：正序 + 倒序（二者无直接关系），

捕捉隐藏状态。

\Rightarrow 不可用于语言模型、建模长序列

* k head: $\text{head}_i = \text{Attn}(Q_i, K_i, V_i)$

* $\text{MultiHead}(Q, K, V) = W^T [\text{head}_1 \dots \text{head}_m]$

多头捕捉局部和全局高维

* Range of Attn

Attn: $O(n)$

序列过长：窗口过大 \Rightarrow 循环 RNN -> 序列渐变

序渐变

* Attn

RNN: $O(\text{length})$

点积 Attn: $g \cdot k \cdot v$

$$\text{Attn}(g, K, V) = \sum_i \frac{\exp(g^T k_i)}{\sum_j \exp(g^T k_j)} v_i$$

* Transformer

* Position Embedding

方法一：对于每个位置 i ，设 $2d$ 个维度。
of Varying Periods

For LM: 将词向量进行 $-INF$ 处理。

每维情况： $p_{i,2j} = \sin(pos/10000^{2j/d})$

$$\text{Attn}(Q, K, V) = \underbrace{\text{softmax}(QK^T)}_{[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]} V$$

$$p_{i,2j+1} = \cos(pos/10000^{2j+1/d})$$

编码方式预先确定，与文本无关

$$\text{Softmax} \left(\begin{array}{c|cc|c} \overrightarrow{q_1} & k_1 & k_2 & \\ \hline \overrightarrow{q_2} & k_1 & k_2 & \\ \overrightarrow{q_3} & k_1 & k_2 & \\ \end{array} \right) \equiv$$

(每行进一) 行之间：时间对
同一注意力得分

行：时间注意力情况
(共 n 行 n 列)

Method 2: 可学习 \vec{p}_i

△ Absolute Embeddings

1. 不同位置同一词汇的嵌入向量不同

2. 避免序列长度 > 内存而使用分块处理

△ Relative Embeddings

通过相对位置量 → 简化、统一问题解决

Method 1: 原始实现

$r_{i,j} = \text{clip}(i-j, -k, k) \rightarrow \text{计算相对位置}$ Masked-Mul-Attn 过程:

每个 $r_{i,j}$ 对应编码向量 $\vec{w}_{r_{i,j}}$

每个 Attn-score 为: $\text{score}_{i,j} = \vec{q}_i^T (\vec{k}_j + \vec{w}_{r_{i,j}}^k)$

每个 Attn 输出量
 \uparrow
 $d_i = \sum_{j \neq i} \text{attn}_{i,j} (\vec{v}_j + \vec{w}_{r_{i,j}}^v)$
 \downarrow
 Softmax 为: 沸腾力分布

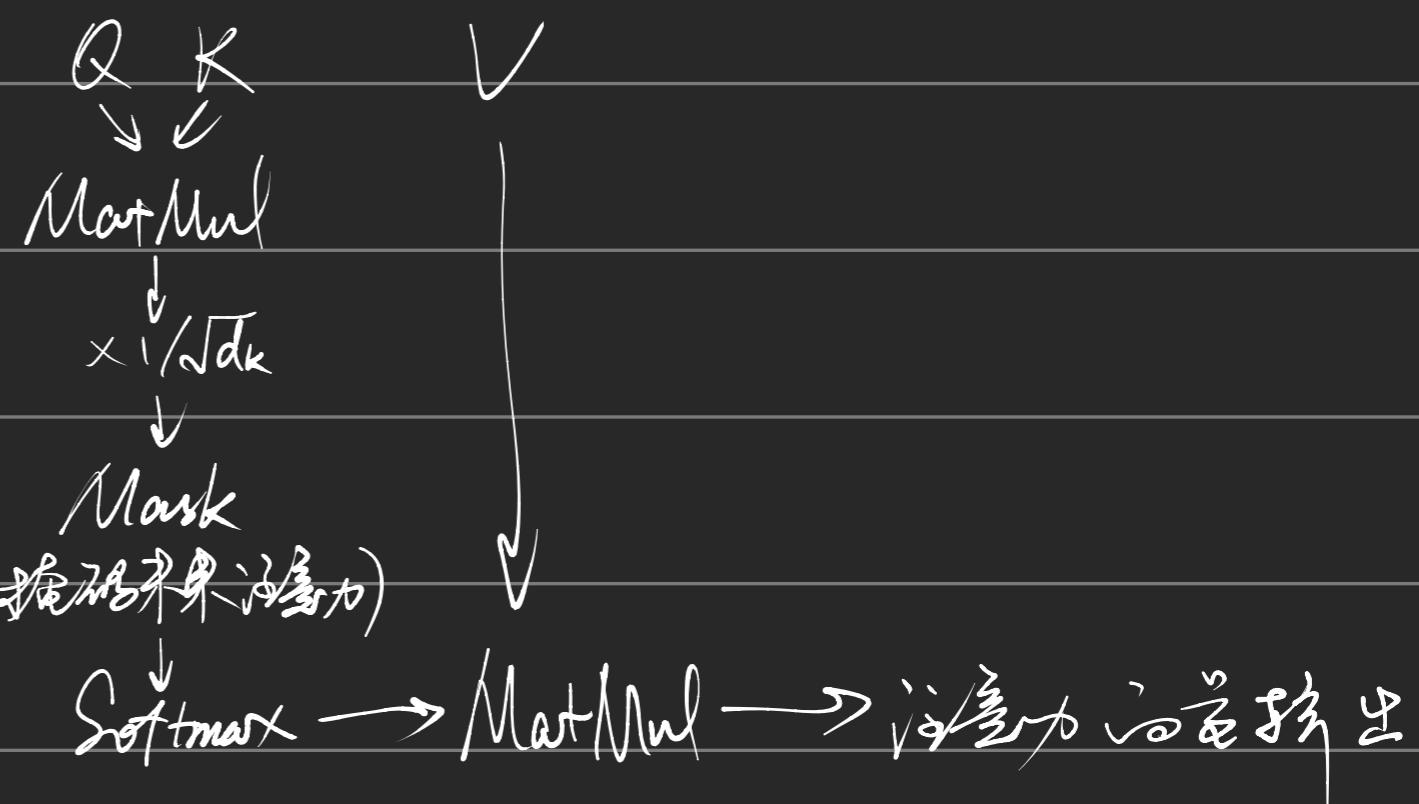
Trick 1: Residual connection

$$x^l = F(x^{l-1}) + x^{l-1}$$

Trick 2: Layer Normalization

对于每层 l 的输出向量 x^l : (包括 Mul-Attn & FFN)

$$\text{归一化量 } \mu^l = 0, \sigma^l = 1$$



Method 2: 沸腾力随距离衰减的通用 (P85 图示)

Method 3: RoPE

$f(g_n, n) = g_n \theta_n \rightarrow \text{旋转矩阵}$

新二范数乘积方式: $\langle f(g_n, n), f(k_m, m) \rangle = (g_n \theta_n)(k_m \theta_m)^T$
 $= g_n \theta_{n-m} k_m^T$ 基于相对位置
 旋转矩阵

θ_m 含义 (P87 图)

(NoPE)

Method 4: No Embedding \rightarrow 抽象

\rightarrow 在 Transformer 中实现

\times Transformer

修改方法: FFN 层: 计算操作

$$m_i = \text{MLP}(d_i) = W_2 * \text{ReLU}(W_1 \times d_i + b_1) + b_2$$

操作 \rightarrow 激活函数

\times 时间复杂度.

$A(Q, K, V)$:

$$\begin{aligned} Q: T \times d_k &\xrightarrow{\text{O}(T^2 d_k)} QK \xrightarrow{\text{softmax}} \text{O}(T^2) \\ K: T \times d_k &\xrightarrow{\text{(T} \times \text{T)}} \text{O}(T^2 d_k) \Rightarrow \text{O}(T^2 d_k) \\ V: T \times d_k &\xrightarrow{\text{Softmax}[(QK)V]} \text{O}(T^2 d_k) \end{aligned}$$

复杂度较高

Sol 1: Sparse Attn

Sol 2: Linear Attn

