注意力和Transformer

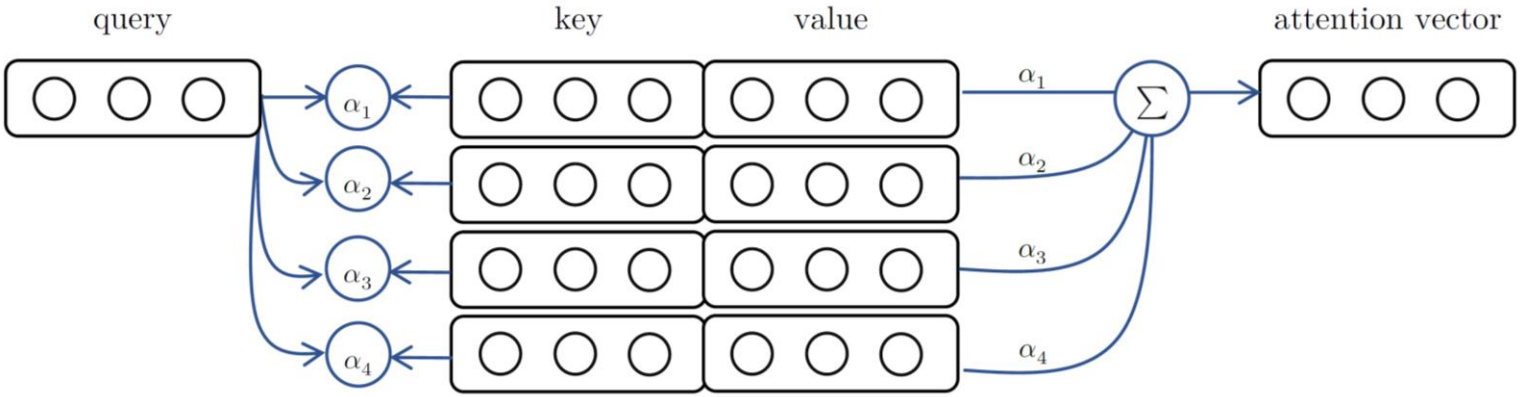
常宝宝
北京大学计算语言学研究所
chbb@pku.edu.cn

概要

- 注意力机制
- 残差连接
- Layer Normalization
- Transformer
- 针对Transformer的改进

注意力机制

- 依据不同处理状态，动态融合信息的神经网络机制
- 待融合的向量 $\langle \boldsymbol{k}_i, \boldsymbol{v}_i \rangle, i = 1, \cdots, n;\ \boldsymbol{k}_i \in \mathbb{R}^{d_k}, \boldsymbol{v}_i \in \mathbb{R}^{d_v}$
- 处理状态向量 $\boldsymbol{q},\ \boldsymbol{q} \in \mathbb{R}^{d_k}$
- 计算$\boldsymbol{q}$对$\langle \boldsymbol{k}_i, \boldsymbol{v}_i \rangle$的关注度权值 $\alpha_i = \mathrm{softmax}\big(f_a(\boldsymbol{q}, \boldsymbol{k}_i)\big)$
- 生成关注向量 $\boldsymbol{a} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i$



注意力机制

- 人类在完成不同任务时对不同信息元素关注程度不同，注意力机制可视作是对此的一种模拟
- 权值 α_i高，意味着融合后的信息对$\boldsymbol{v}_i$的关注度高，反之则关注度低
- 关注度权值计算

$$\alpha_i = \mathrm{softmax}\big(f_a(\boldsymbol{q}, \boldsymbol{k}_i)\big) = \frac{\exp\big(f_a(\boldsymbol{q}, \boldsymbol{k}_i)\big)}{\sum_{j=1}^{n} \exp\big(f_a(\boldsymbol{q}, \boldsymbol{k}_j)\big)}$$

- 如何选择$f_a(\boldsymbol{q}, \boldsymbol{k}_i)$？

注意力机制

- 点积式关注度(dot product)

$$f_a(\boldsymbol{q}, \boldsymbol{k}_i) = \boldsymbol{q} \cdot \boldsymbol{k}_i$$

- 加法式关注度(additive)

$$f_a(\boldsymbol{q}, \boldsymbol{k}_i) = \boldsymbol{u}^\top \tanh(W[\boldsymbol{q}; \boldsymbol{k}_i])$$

- 乘法式关注度(multiplicative)

$$f_a(\boldsymbol{q}, \boldsymbol{k}_i) = \boldsymbol{k}_i^\top W \boldsymbol{q}$$

- 缩放点积式关注度(scaled dot production)

$$f_a(\boldsymbol{q}, \boldsymbol{k}_i) = \frac{\boldsymbol{q} \cdot \boldsymbol{k}_i}{\sqrt{d_k}}$$

注意力机制

- 自注意力机制(self attention)
  - 单独用于编码器端(encoder)或解码器端(decoder)
  - 用于融合语境信息生成语境敏感的语言表示(词向量)
  - 一般而言，建模单一序列元素之间依赖关系

- 一般(交叉)注意力机制(attention)
  - 同时用于编码器端和解码器端
  - 解码器端状态向量作为query向量，编码器端向量作为key向量和value向量
  - 用于动态确定编码器端信息对解码决策的不同影响
  - 建模编码序列与解码序列间的依赖关系

概要

- 注意力机制
- 残差连接
- Layer Normalization
- Transformer
- 针对Transformer的改进

残差连接

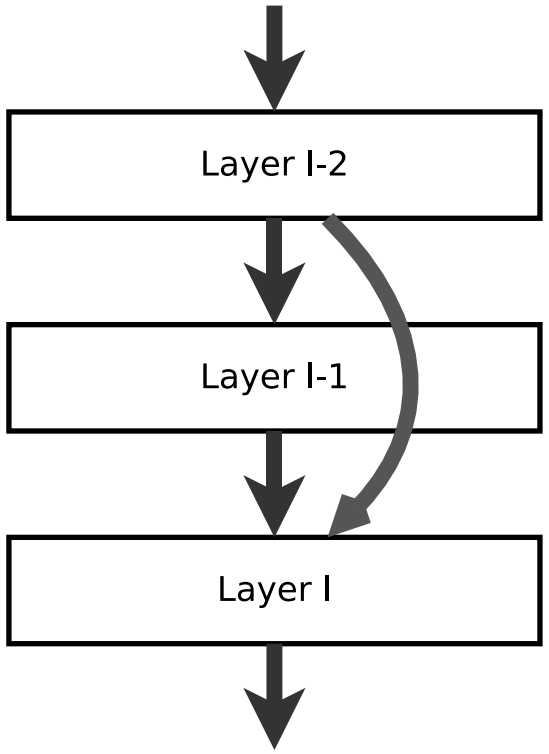
- 神经网络是在拟合函数$F(\boldsymbol{x})$
- 直接构建拟合函数的网络
  - FNN、CNN
- 通过对残差建模拟合函数
  - 残差(residual)

$$R(\boldsymbol{x}) = F(\boldsymbol{x}) - \boldsymbol{x}$$

  - 构建拟合如下映射的网络

$$F(\boldsymbol{x}) = R(\boldsymbol{x}) + \boldsymbol{x}$$

- 跨层连接



残差连接

- 网络深度和拟合精度
  - 增加深度并不总能改进拟合精度
- 缓解梯度消失问题
- 有助于建立深层网络
- 残差连接有很多成功应用。ResNet, Transformer

概要

- 注意力机制
- 残差连接
- Layer Normalization
- Transformer
- 针对Transformer的改进

Layer Normalization

- 输入：隐层向量 $\boldsymbol{x} = (x_1, \cdots, x_d)$

① 计算均值 $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$

② 计算方差 $\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2$

③ 向量归一 $\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$

④ 缩放平移 $y_i = \gamma_i \hat{x}_i + \beta_i$

- 输出：$\boldsymbol{y} = \mathrm{LayerNorm}_{\boldsymbol{\gamma},\boldsymbol{\beta}}(\boldsymbol{x})$

- 稳定隐藏状态分布
- 提高训练速度
- 改善模型推广能力

概要

- 注意力机制
- 残差连接
- Layer Normalization
- Transformer
- 针对Transformer的改进

Transformer

- 回顾 循环神经网络
  - 时刻t的隐状态$\boldsymbol{h}_t$是输入$\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$的函数，融合了左边所有的语境信息
  - 双向神经网络可以融合左右两边的语境信息

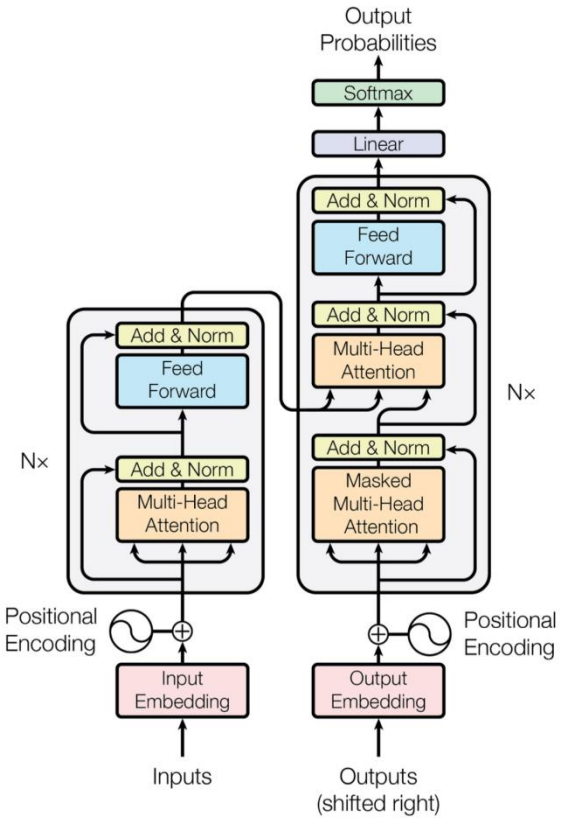
- 循环神经网络局限性
  - 长距离依赖关系建模存在困难

  建模token $\boldsymbol{x}_i$ 和 token $\boldsymbol{x}_{i+t}$ 间关系需要t步梯度传递
  - 无法并行计算

  token $\boldsymbol{x}_i$隐状态计算需要先于token $\boldsymbol{x}_{i+1}$隐状态进行

Transformer

- 编码器-解码器架构
- 编码器、解码器也可以独立使用
- 目前NLP处理中高频使用的模型
- GPT、Bert等 都基于Transformer
- 编码器和解码器结构类似但有差异，均为多层结构

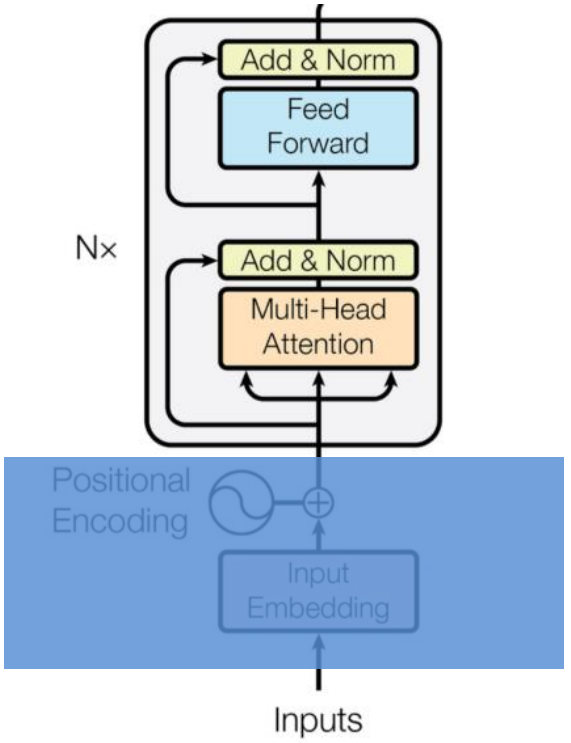


Transformer之编码器

- 多层结构，可由N层组成

- 单层组成
  (1) 多头自注意力子层
  (2) 前馈神经网子层

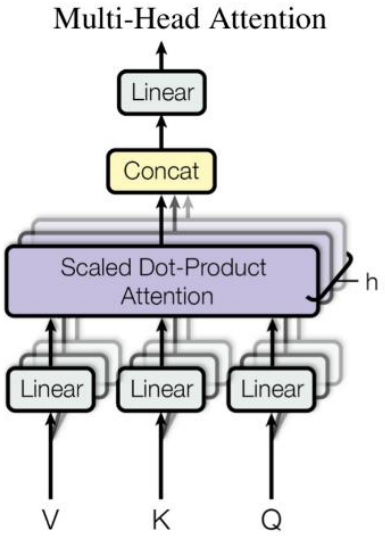
- 子层内部设有残差连接
- 对子层输出应用Layer Normalization



多头自注意力子层

- 总体思想:
  $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$是token向量序列，以自注意力方式生成$\boldsymbol{x}_i$的表示
  $$\boldsymbol{h}_i = \sum_{i=1}^{n} \alpha_j \boldsymbol{x}_j$$
- 采用缩放点积式关注度, 令$\boldsymbol{q}_i = \boldsymbol{k}_i = \boldsymbol{v}_i = \boldsymbol{x}_i$
  $$\boldsymbol{\alpha} = \text{softmax}\left(\left[\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_1}{\sqrt{d_k}}, \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_2}{\sqrt{d_k}}, \cdots \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_n}{\sqrt{d_k}}\right]\right)$$
- 生成$\boldsymbol{x}_i$关注向量
  $$\boldsymbol{h}_i = \sum_{j=1}^{n} \alpha_j \boldsymbol{v}_j$$

多头自注意力子层



- 关注向量 矩阵运算

$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V$$

Q、K、V 由n行组成，第i行对应$\boldsymbol{q}_i$、$\boldsymbol{k}_i$和$\boldsymbol{v}_i$

- 多头机制：多角度并行生成多组(h组)注意力向量

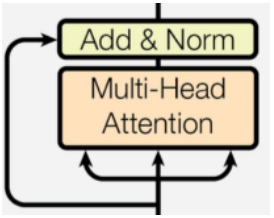
$$Q=K=V=X$$

$$\text{head}_i=\text{Attention}\left(QW_i^Q,KW_i^K,VW_i^V\right)$$

$$\text{MultiHead}(Q,K,V)=\text{Concat}(\text{head}_1,\text{head}_2,\cdots,\text{head}_h)W^O$$

- 残差连接及Layer Normalization

$$x^{(i)}\leftarrow\text{LayerNorm}\left(x^{(i-1)}+\text{Sublayer}\left(x^{(i-1)}\right)\right)$$



前馈神经网子层

- 关注向量是线性组合

- 通过前馈神经网引入非线性

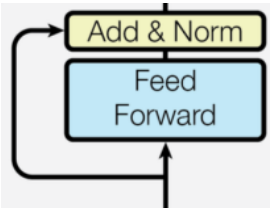
$$\mathrm{FFN}(\boldsymbol{x}) = W_2\,\mathrm{ReLU}(W_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2$$

  $\boldsymbol{x}$是多头自注意力层的输出

- 同一层的不同位置共享参数、跨层不共享参数

- 残差连接和Layer Normalization

$$x^{(i)} \leftarrow \mathrm{LayerNorm}\left(x^{(i-1)} + \mathrm{Sublayer}\left(x^{(i-1)}\right)\right)$$

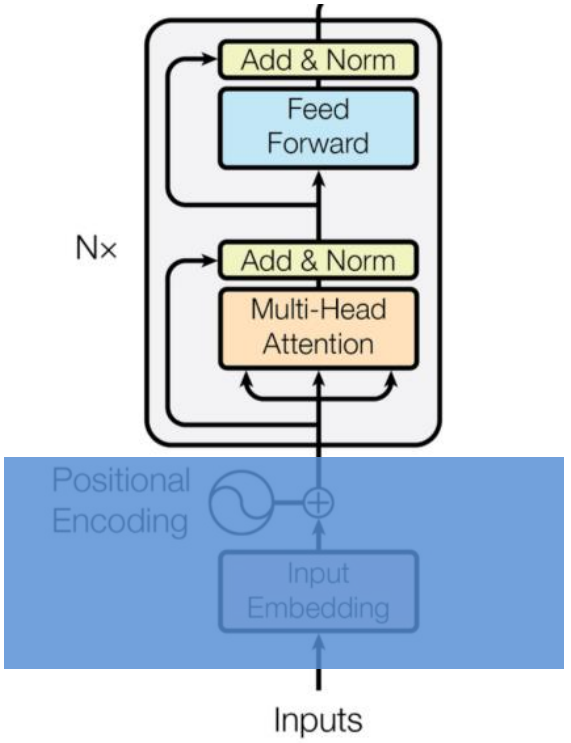


Transformer之编码器

- 多层结构，可由N层组成

- 单层组成
  (1) 多头自注意力子层
  (2) 前馈神经网子层

- 子层内部设有残差连接
- 对子层输出应用Layer Normalization



位置编码

- 自注意力机制丢失了序列顺序序信息，是词袋模型

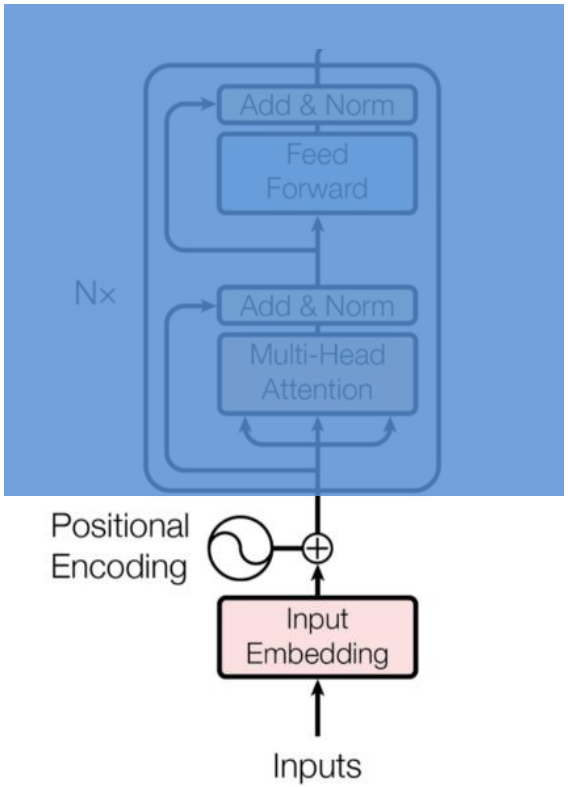
$$\boldsymbol{h}_i = \sum_{j=1}^{n} \alpha_j \boldsymbol{v}_j$$

- 引入位置向量 $\boldsymbol{p}_i$

$$\boldsymbol{x}_i = \boldsymbol{t}_i + \boldsymbol{p}_i$$

- $\boldsymbol{p}_i$作为模型参数，训练得到

- 不足：缺乏外推(extrapolation)能力
  测试集中可能出现较长的句子



位置编码

- 位置向量的每个维度的值通过三角函数计算得到
  - 位置向量同一维采用同一个周期函数计算
  - 维度索引越大函数周期越大$(2\pi, \cdots, 10000 \cdot 2\pi)$
  - 确定式(无需训练)位置向量

$$p_{pos,k} = \begin{cases} \sin(\omega_i \cdot pos), & \text{if } k = 2i \\ \cos(\omega_i \cdot pos), & \text{if } k = 2i+1 \end{cases}$$

其中

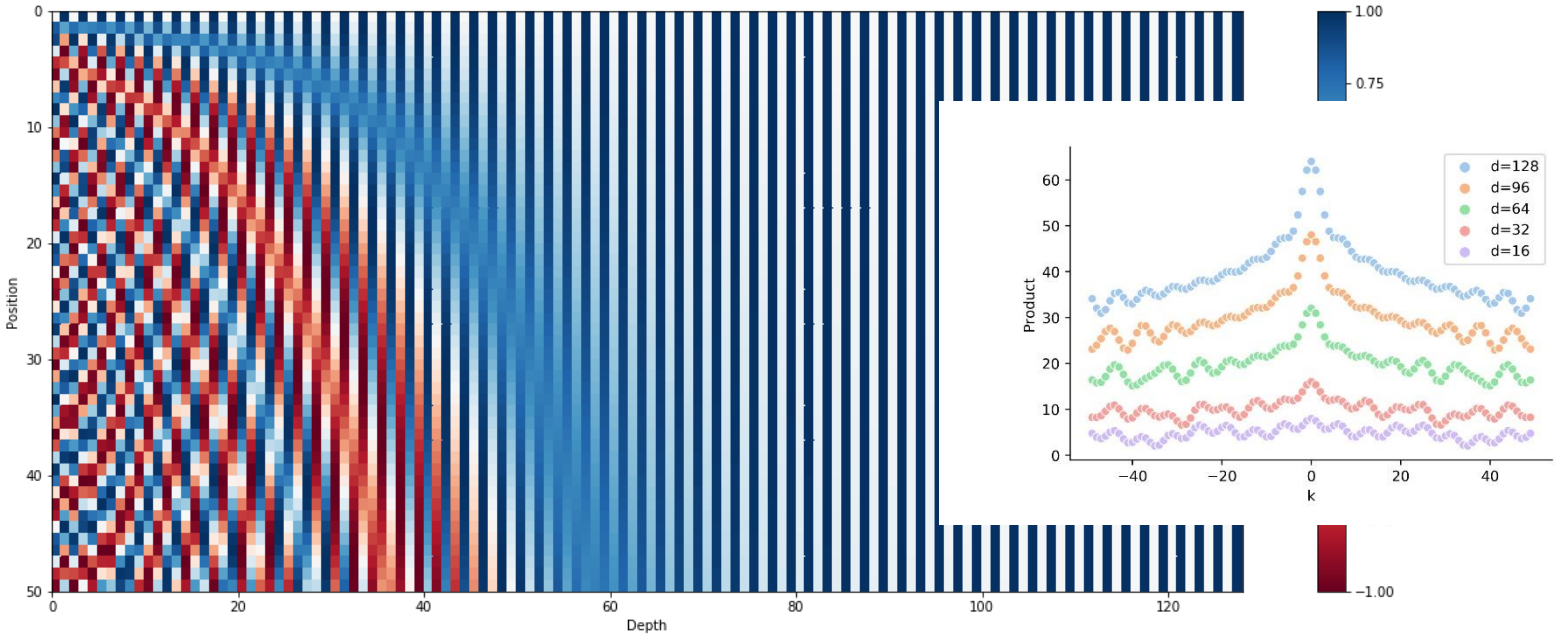
$$\omega_i = \frac{1}{10000^{\frac{2i}{d}}}$$

$$\boldsymbol{p}_{pos} = \begin{bmatrix} \sin(\omega_0 \cdot pos) \\ \cos(\omega_0 \cdot pos) \\ \sin(\omega_1 \cdot pos) \\ \cos(\omega_1 \cdot pos) \\ \vdots \\ \sin(\omega_{d/2-1} \cdot pos) \\ \cos(\omega_{d/2-1} \cdot pos) \end{bmatrix}$$

位置编码

$$\boldsymbol{p}_t \cdot \boldsymbol{p}_{t+k} = \sum_{i=0}^{d/2-1} \cos(\omega_i k)$$

- 建模相对位置(距离差)
- 相对距离一样的位置，位置编码有一致的线性变换关系
- 无需训练、具有外推能力

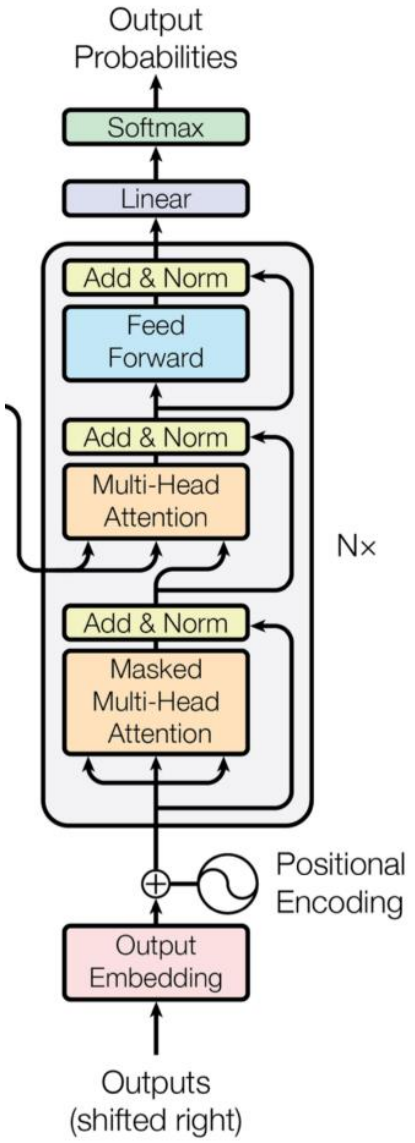


Transformer之解码器

- 多层结构，可由N层组成

- 单层组成
  (1) 多头遮蔽自注意力子层
  (2) 多头交叉注意力子层
  (3) 前馈神经网子层

- 子层内部设有残差连接
- 对子层输出应用Layer Normalization



解码器的工作方式

- 自回归生成模型

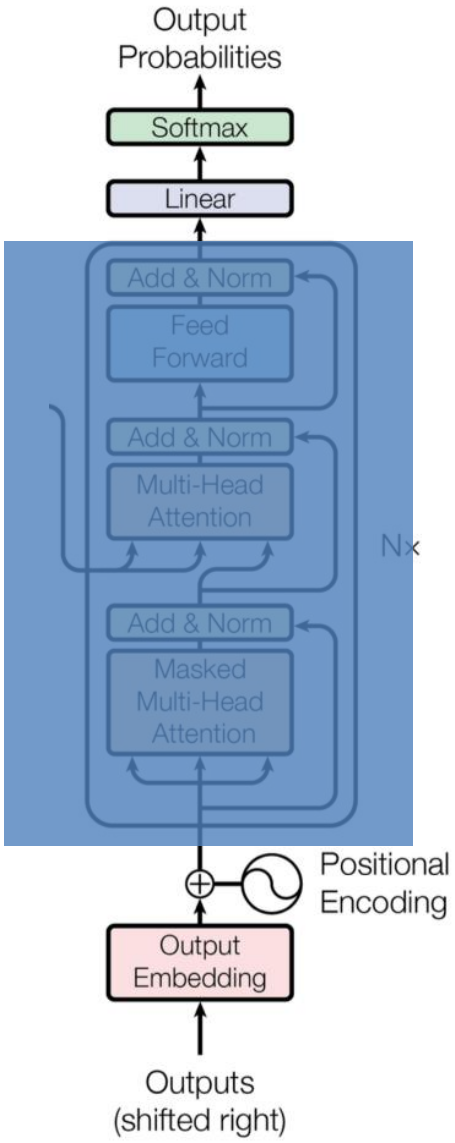
$$X = x_1 x_2 \cdots x_n \Longrightarrow Y = y_1 y_2 \cdots y_m$$

$$P(Y|X) = \prod_{i=1}^{m} P(y_i|Y_{<i}, X)$$

- 逐步生成，第i步生成y_i (贪婪解码)

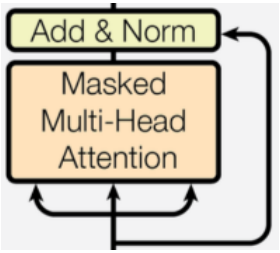
$$\hat{y} = \underset{y_i}{\operatorname{argmax}}\, P(y_i|Y_{<i}, X)$$

- transformer解码器
  - 输入：y_{i-1}的向量表示+位置向量
  - 以自注意力方式关注$Y_{<i}$
  - 以一般(交叉)注意力方式关注X



多头遮蔽自注意力子层

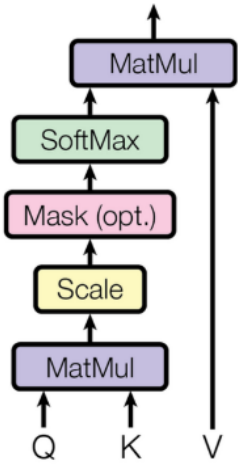
- 解码端以自回归方式生成输出序列
- 在生成y_i时，无法关注y_i之后的序列
- Query、key和value向量源自解码端
  $Q = K = V = Y$
- 为了便于并行计算，引入遮蔽(mask)策略



$$f_a(\boldsymbol{q}_i, \boldsymbol{k}_j) = \begin{cases} \dfrac{\boldsymbol{q}_i \cdot \boldsymbol{k}_j}{\sqrt{d_k}} & j \le i \\ -\infty & j > i \end{cases}$$

- 生成y_{i+1}时的关注权值

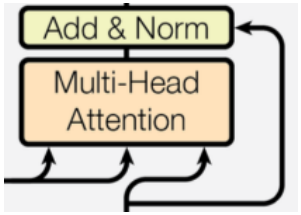
$$\boldsymbol{\alpha}_i = \text{softmax}\left(\left[\frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_0}{\sqrt{d_k}} \quad \cdots \quad \frac{\boldsymbol{q}_i \cdot \boldsymbol{k}_i}{\sqrt{d_k}} \quad -\infty \quad \cdots \quad -\infty\right]\right)$$



多头交叉注意力层

- 在生成y_i关注编码端输出的编码向量H
- key向量与value向量源自编码器

$$K = V = H$$



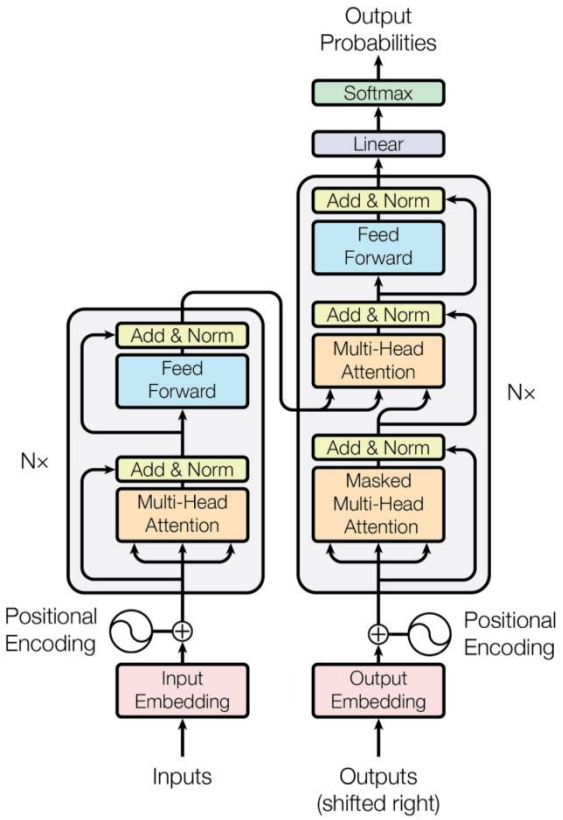
- query向量源自解码器

$$Q = Y$$

$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \cdots, \text{head}_h)W^O$$

单层Transformer向量维度变化

- token向量、位置向量维度 d_{model}
- query、key向量维度d_k
- value向量维度d_v
- $d_k = d_v = d_{model}/h$
- head_i向量维度d_v
- 多头关注向量维度d_{model}
- 前馈神经网隐层维度d_{ff}
- 前馈神经网子层输出维度d_{model}



weight tying和label smoothing

- 什么是weight tying?

  输入层embedding U，输出层embedding V，令：

  $$U = V$$

  提升性能(困惑度)、减少参数

- 什么是label smoothing?

  标准答案(gold)是one-hot分布，使得模型赋予正确类别过大的分值，造成过渡拟合、损害推广能力

  引入soft target分布

  $$y_k^{LS} = y_k(1-\alpha) + \frac{\alpha}{K}$$

Transformer优点

- 可以并行计算，可以充分利用GPU资源
- 可以直接计算每个词之间的相关性，梯度计算无需多步传递
- 由于有残差链接和Layer Norm，多层堆叠可以缓解梯度消失的问题

概要

- 注意力机制
- 残差连接
- Layer Normalization
- Transformer
- 针对Transformer的改进

Transformer中的激活函数

- ReLU 函数

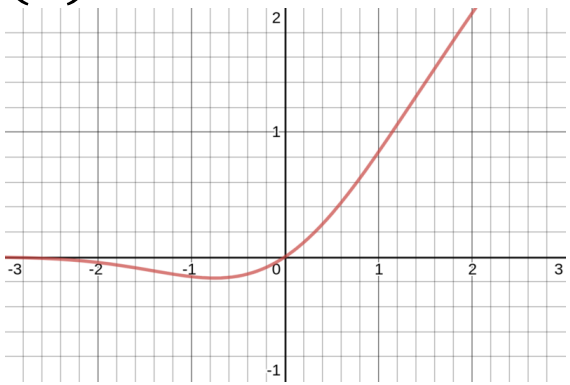
$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

- GeLU函数(Gaussian error Linear Unit)

$$\mathrm{GELU}(x) = xP(X \leq x) = x\Phi(x)$$

其中，$X \sim \mathcal{N}(0,1)$

平滑版的ReLU函数



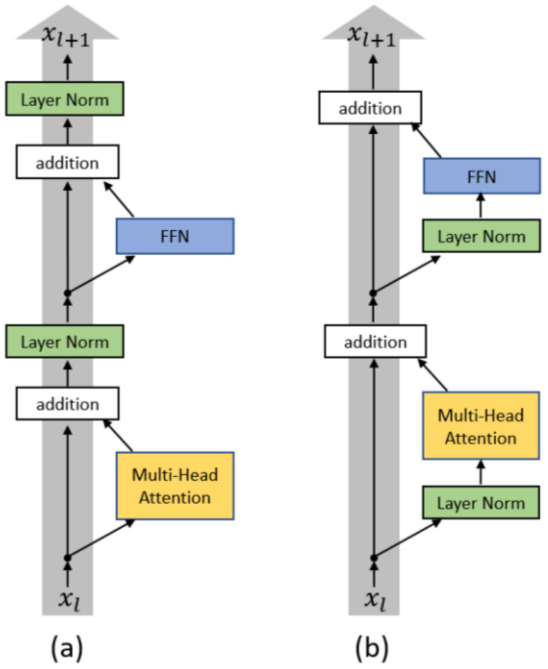
- GEGLU

$$\mathrm{GEGLU}(\boldsymbol{x}, W, V) = \mathrm{GeLU}(W\boldsymbol{x}) \odot V\boldsymbol{x}$$

Transformer中的Layer Normalization

- Post-LN transformer Layer
  $x^{(i)} \leftarrow \text{LayerNorm}\left(x^{(i-1)} + \text{Sublayer}\left(x^{(i-1)}\right)\right)$
- LN位于两个残差模块之间
- 阻断了跨层信息直通路径

- Pre-LN transformer Layer
  $x^{(i)} \leftarrow x^{(i-1)} + \text{Sublayer}\left(\text{LayerNorm}(x^{(i-1)})\right)$
- 将LN置于残差模块内部和子层前端
- 训练更加容易、加快收敛速度



相对位置编码

- 绝对位置编码 编码token的绝对位置
- 相对位置编码 编码相对于查询位置的相对位置
- 在绝对位置编码中

$$q_i = (x_i + p_i)W_Q$$
$$k_j = (x_j + p_j)W_K$$
$$v_j = (x_j + p_j)W_V$$

$$q_i k_j^\top = (x_i + p_i)W_Q W_K^\top (x_j + p_j)^\top$$

$$h_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + p_j W^V)$$

相对位置编码

$$q_i k_j^\top = \underbrace{x_i W_Q W_K^\top x_j^\top}_{(a)} + \underbrace{x_i W_Q W_K^\top p_j^\top}_{(b)} + \underbrace{p_i W_Q W_K^\top x_j^\top}_{(c)} + \underbrace{p_i W_Q W_K^\top p_j^\top}_{(d)}$$

- 将key的绝对位置编码改用相对位置编码，可有不同的变体

(1) $$q_i k_j^\top = \underbrace{x_i W_Q W_{K,E}^\top x_j^\top}_{(a)} + \underbrace{x_i W_Q W_{K,R}^\top \mathrm{R}_{i-j}^\top}_{(b)} + \underbrace{u W_{K,E}^\top x_j^\top}_{(c)} + \underbrace{v W_{K,R}^\top R_{i-j}^\top}_{(d)}$$

(2) $$q_i k_j^\top = \underbrace{x_i W_Q W_K^\top x_j^\top}_{(a)} + \underbrace{x_i W_Q R_{i-j}}_{(b)}$$

(3) $$q_i k_j^\top = \underbrace{x_i W_Q W_K^\top x_j^\top}_{(a)} + r_{i-j}$$

相对位置编码

- value向量中的位置编码

  ⑴ $h_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + R_{i-j}^V)$

  ⑵ $h_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V)$

- 相对位置编码可以采用三角函数计算，也可以采用模型参数进行训练

注意力机制的稀疏化

- 自注意力计算复杂度$O(n^2)$，计算瓶颈
- 对于x_i而言，需要关注所有的token表示

$$x_1, x_2, \cdots, x_n$$

- 稀疏注意力机制，对此进行改进，对x_i只需要关注所有token表示的一个子集

$$S_i \subset \{x_1, x_2, \cdots, x_n\}$$

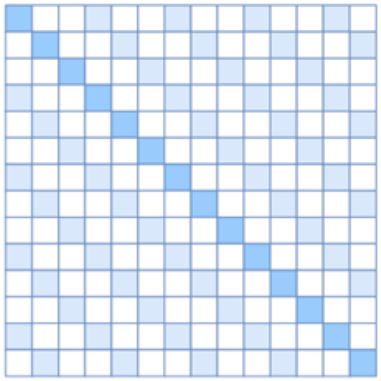
- 对于x_i而言，关注向量可做如下计算

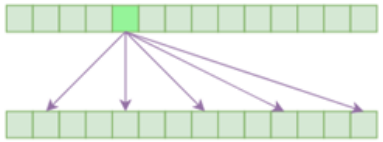
$$a(x_i, S_i) = \text{softmax}\left(\frac{(W_q x_i) K_{S_i}^{\top}}{\sqrt{d}}\right) V_{S_i}$$

$$K_{S_i} = \left(W_k x_j\right)_{j \in S_i}, V_{S_i} = \left(W_v x_j\right)_{j \in S_i}$$

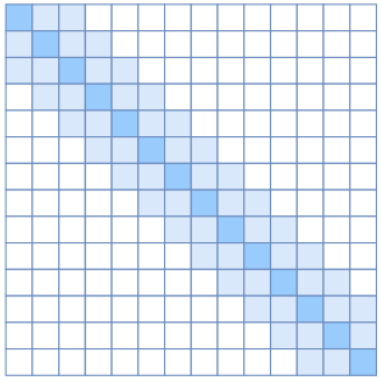
稀疏Transformer

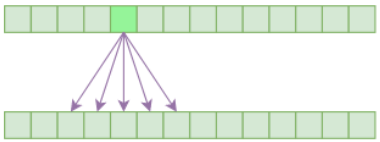
- 设置多种关注模式，通过传递性关注所有token表示





跨步关注

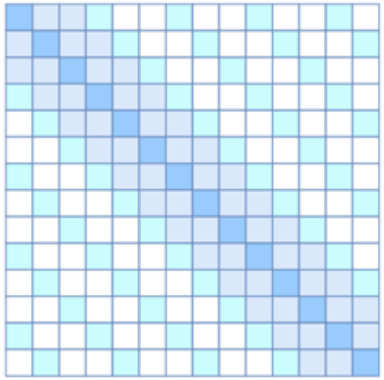


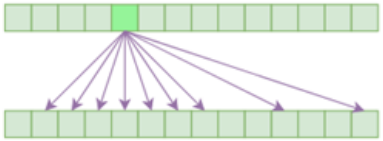


局部关注

逐层交替使用或者合并使用

$$O(n^2) \rightarrow O(n\sqrt{n})$$





子层并行化

- 在标准transformer中，子层计算按照先后串行进行

- 并行子层

$$y = x + \mathrm{MLP}\big(\mathrm{LayerNorm}(x)\big) + \mathrm{Attention}\big(\mathrm{LayerNorm}(x)\big)$$

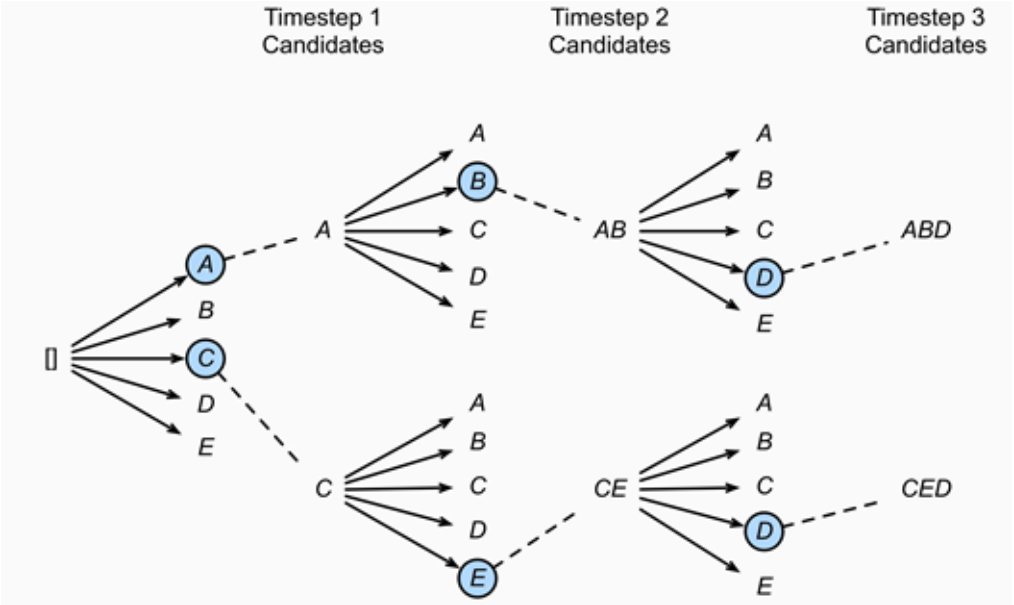
- 并行子层有利于加速

生成模型中的解码策略

$$P(x_i|x_{<1}) = \frac{\exp(u_i)}{\sum_j \exp(u_j)}$$

(1) 贪婪解码

(2) 柱形搜索



生成模型中的解码策略

(3) 随机采样

$$\hat{x} \sim p(x_i|x_{<i})$$

(4) 带温度控制的随机采样

$$p(x_i|x_{<i}) = \frac{\exp(u_i/t)}{\sum_j \exp(u_j/t)}$$

$$\hat{x} \sim p(x_i|x_{<i})$$

(5)top-K采样 只有概率最大的K个词有机会

(6) 核心采样 动态确定可被采样的词集(概率和不小于p)

$$\sum_{x \in V^{(p)}} P(x_i|x_{1:i-1}) \geq p$$

Transformer使用

- 作为encoder-decoder架构使用
  - 序列转换
  - 预训练模型BART、T5…
- 仅使用encoder架构
  - 序列标注
  - BERT、 RoBERTa 、…
- 仅使用decoder架构
  - 语言模型
  - GPT、GPT-2、GPT-3…