北京大学

注意力机制与Transformer

主讲人：董豪　讲义：董豪

内容提要

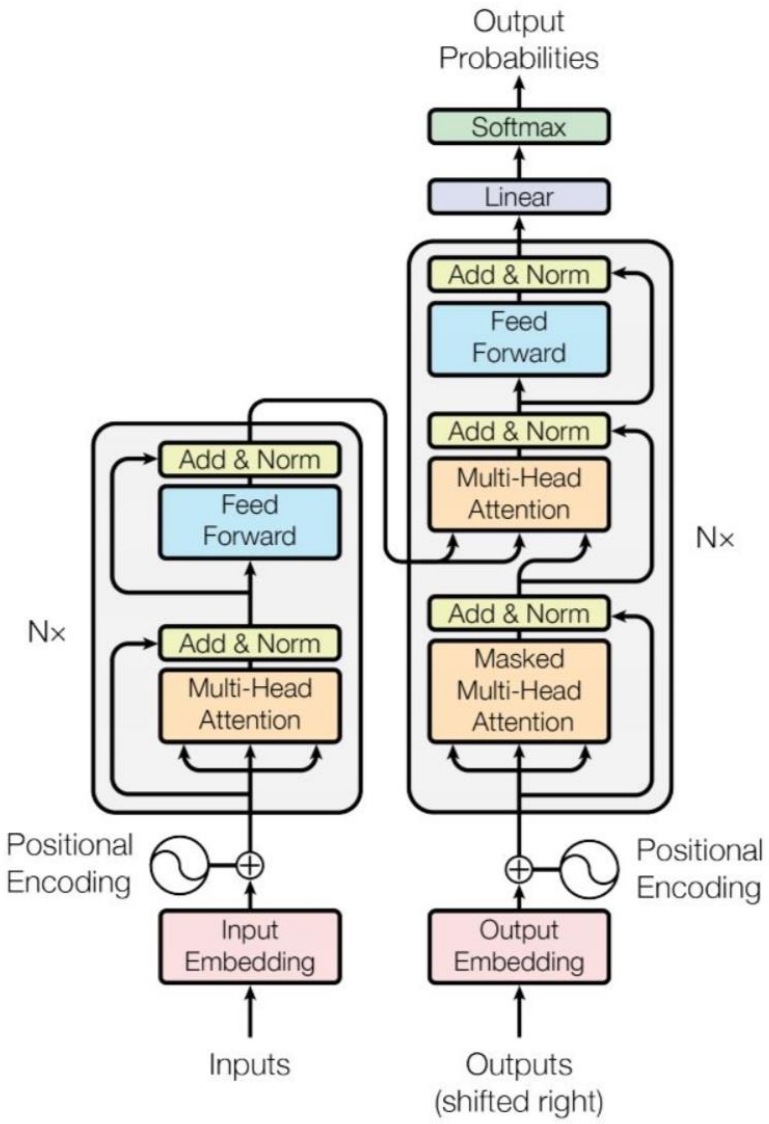
- 动机
- 自注意力机制
- 多头注意力
- 掩码和位置编码
- Transformer
- 常用预训练模型

动机

- 早期序列建模方法具有局限性：
  - RNN（循环神经网络）：RNN可以处理序列数据，但在处理长序列时存在梯度消失与梯度爆炸问题。
  - LSTM（长短时记忆）与GRU（门控循环单元）：为解决RNN的梯度问题，引入了更复杂的结构，如LSTM和GRU，但计算效率相对较低。
  - CNN（卷积神经网络）：虽然主要用于图像处理，但也可以应用于序列数据，如文本分类。然而，CNN的感受野有限，不能直接捕捉全局依赖关系。

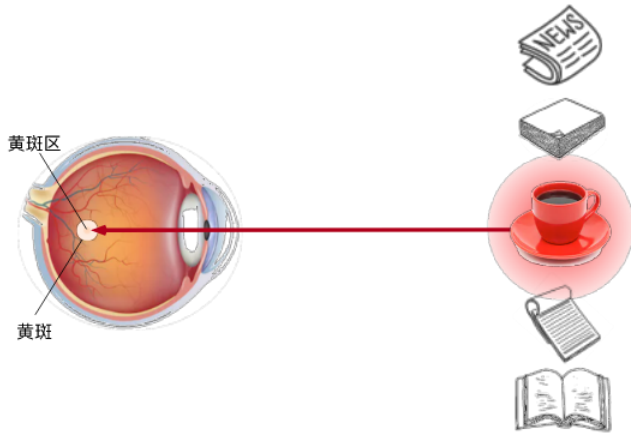
长度依赖问题！

动机

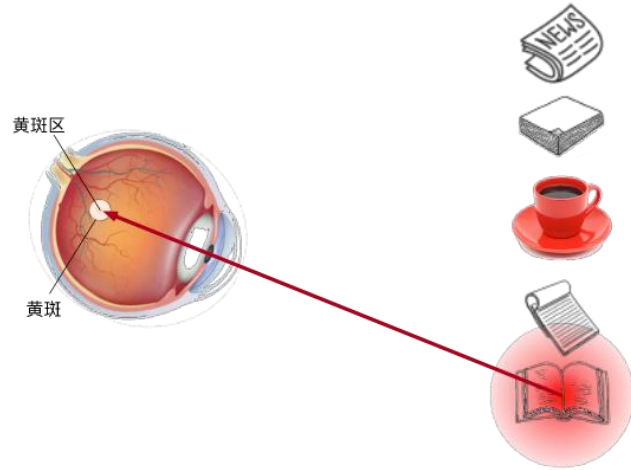

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

动机

- 人类视觉注意力的特点：
  - 有限的处理能力，而忽略无关的信息。
    稀有资源
  - 空间选择性，人类视觉注意力可以在空间中选择性地关注特定位置的信息。
  - 特征选择性，人类视觉注意力还可以根据特征（如颜色、形状、运动等）选择性地关注信息。



非自主性提示：
红杯子吸引人



自主性提示：
想读书

Self-attention

自注意力机制

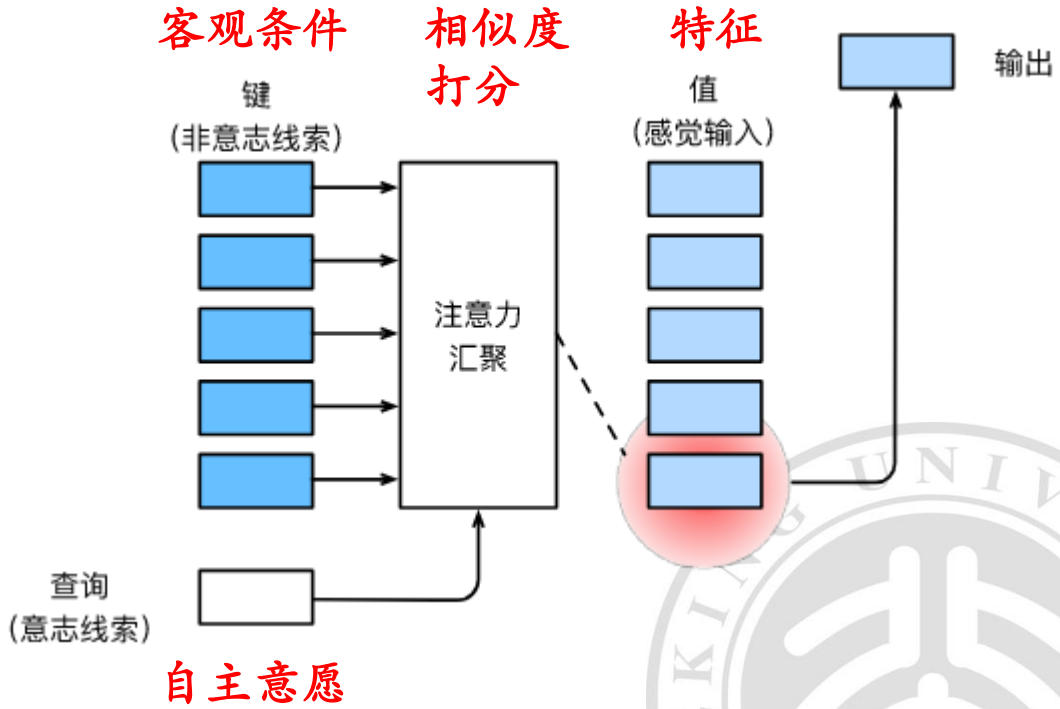
- 注意力提示: 查询、键和值

  - 非自主性提示:
    键（Key）、值（Value）

  - 自主性提示:
    查询（Query）

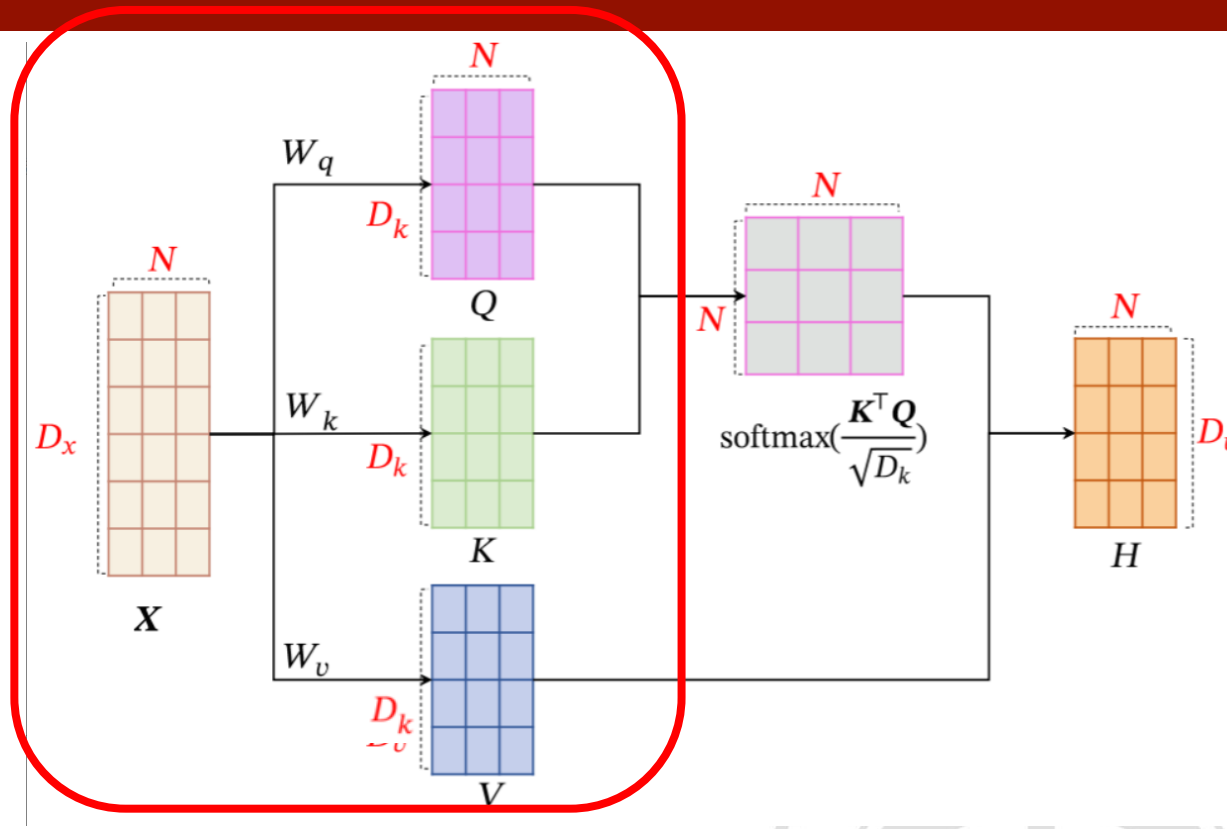
  - 注意力汇聚：**打分函数，注意力权重矩阵，加权平均**

$$f(x) = \sum_{i=1}^{n} \alpha(x, x_i) y_i,$$



自注意力机制

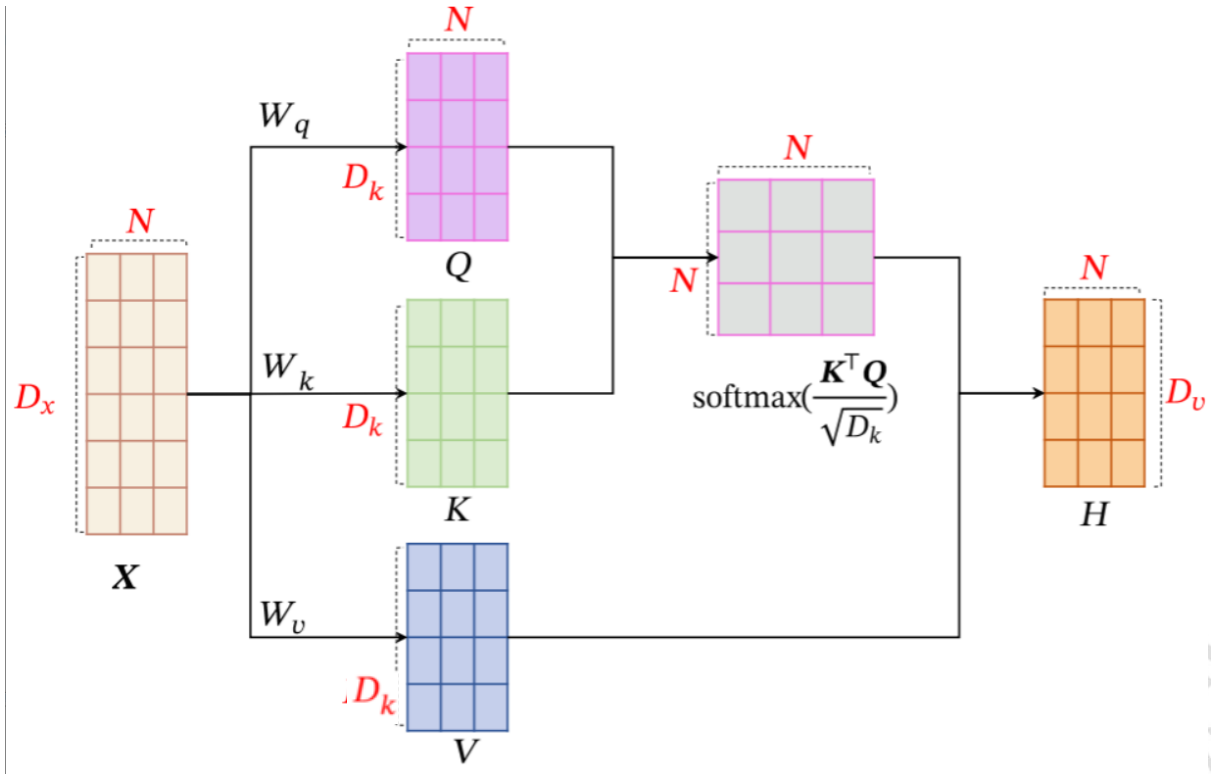
- 词嵌入到Q、K、V
  - Q、K、V是三个不同的空间
  - 我们首先将输入序列转换为嵌入表示，然后，通过三个不同的线性层投影Wq、Wk 和 Wv，分别生成**查询矩阵（Q）、键矩阵（K）和值矩阵（V）**。这三个矩阵将用于计算注意力输出。
  - N是序列长度
  - D_x是嵌入的特征维度
  - D_k是QKV矩阵的维度，也叫head dim



$$\text{softmax}\left(\frac{Q \times K^{\mathrm{T}}}{\sqrt{D}}\right) V = Z$$

自注意力机制

- 打分函数：缩放点积（Scaled Dot-Product）
  - 这种注意力汇聚方法采用点积计算Q和K之间的相似度，然后将结果除以缩放因子以确保梯度稳定。之后，对相似度矩阵应用softmax函数以归一化，得到**注意力权重矩阵**，最后将权重与V相乘并求和。
    - Attention(Q, K, V)：表示注意力汇聚的输出结果。
    - Q：表示查询（Query）矩阵。
    - K：表示键（Key）矩阵。
    - V：表示值（Value）矩阵。
    - d_k：表示键向量的维度。
    - *：表示矩阵乘法。
    - ^T：表示矩阵转置。
    - softmax：表示应用softmax函数，对结果进行归一化。
    - sqrt：表示平方根。

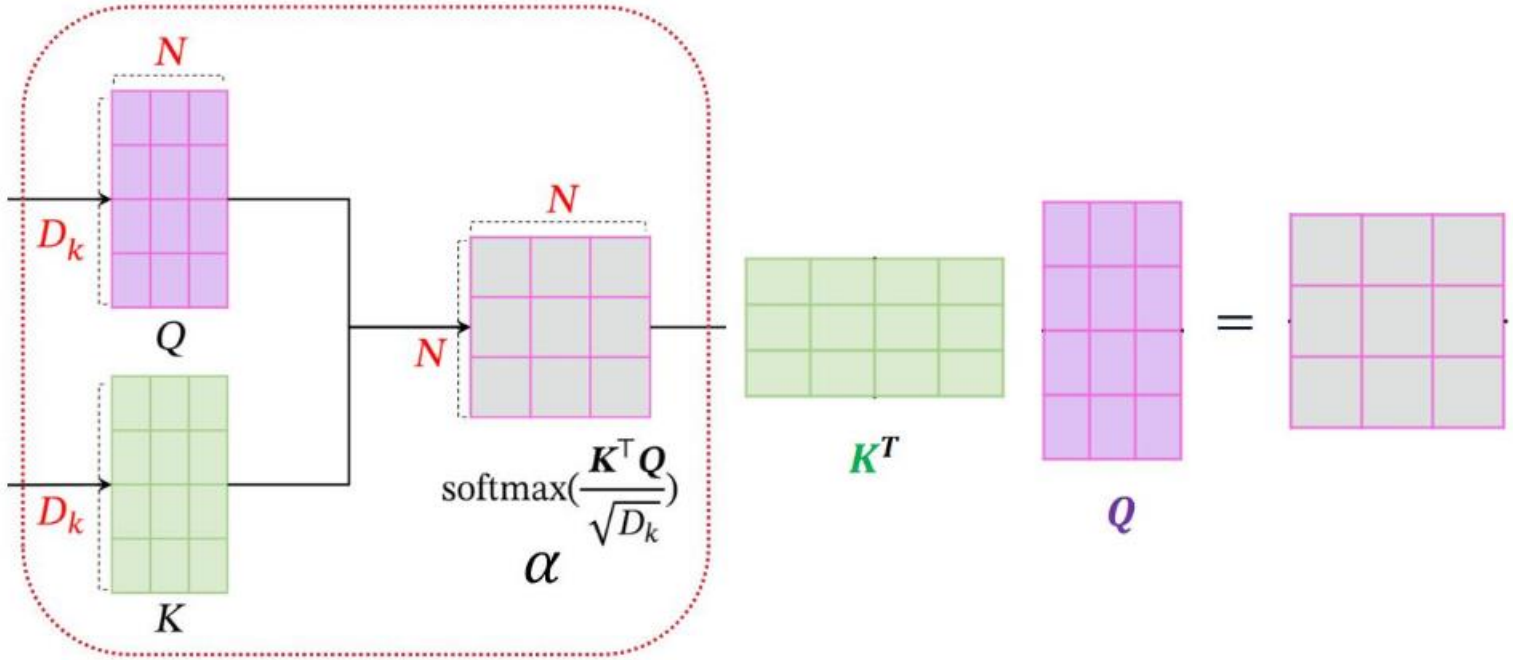



```
Attention(Q, K, V) = softmax(Q * K^T / sqrt(d_k)) * V
```

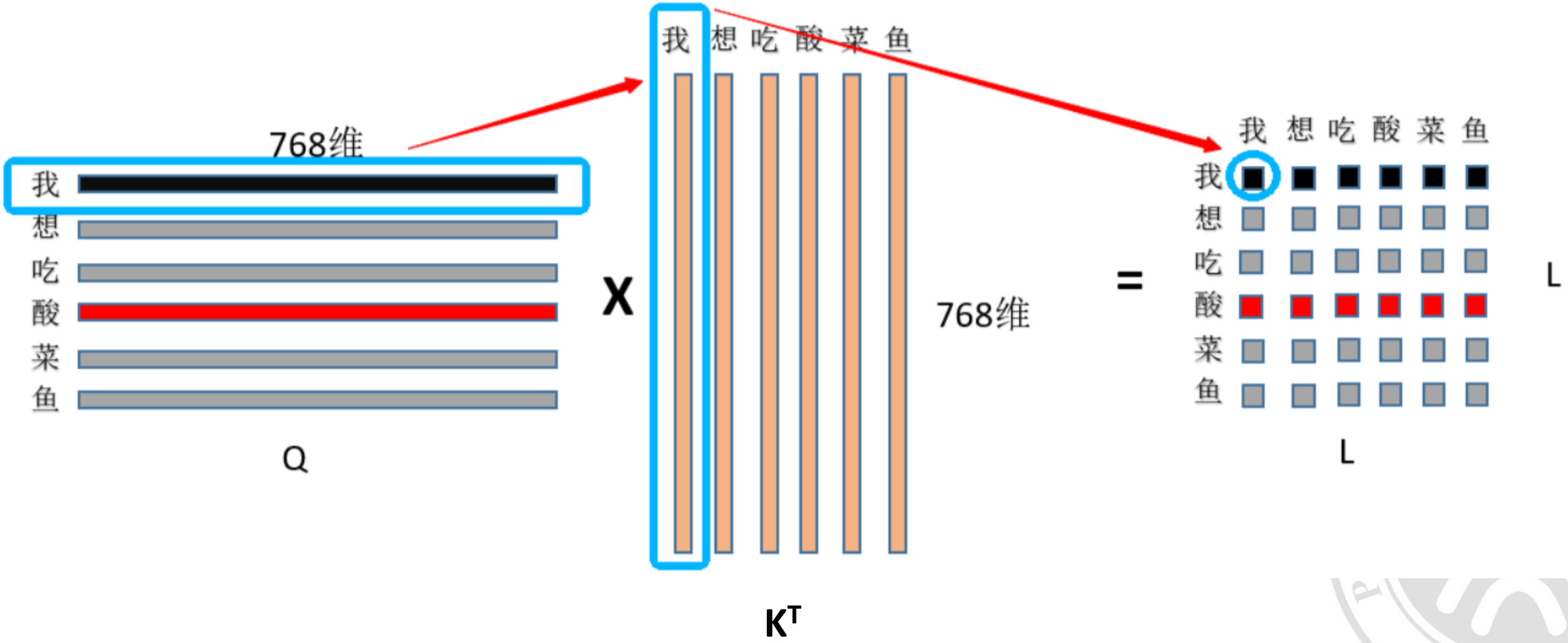
自注意力机制

- 衡量 查询向量与 键向量之间的相似度，以确定输入序列中哪些部分与给定的查询相关。

$$K^T * Q$$



自注意力机制

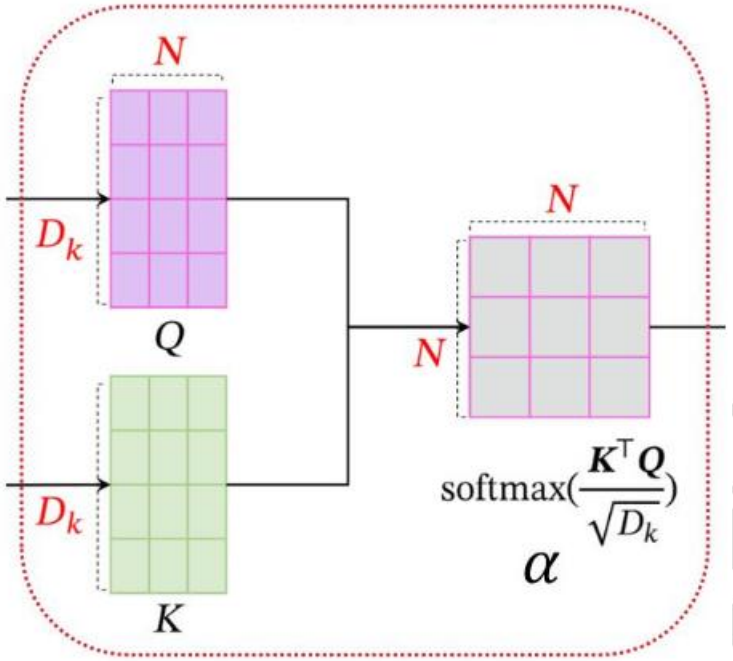


自注意力机制

- 缩放与归一：得到**注意力权重矩阵**

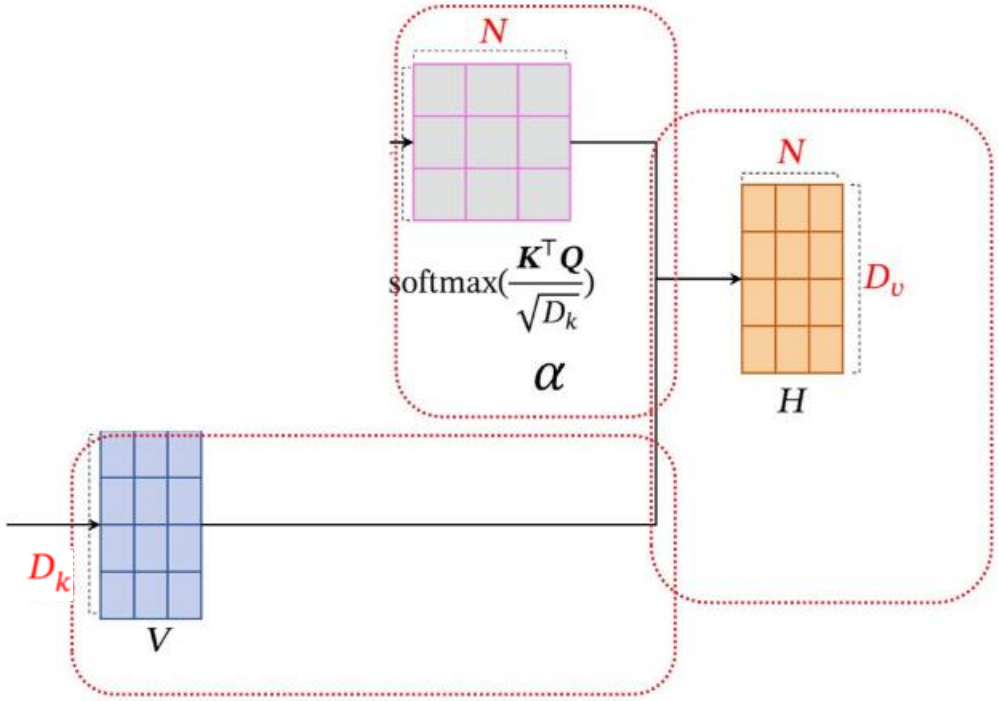
  - 缩放：将每个点积结果除以一个缩放因子，通常是QKV维度的平方根。这样做的目的是防止点积结果过大，导致接下来的softmax操作中梯度过小。

  - Softmax归一化：对计算得到的缩放点积矩阵应用softmax函数。这一步将点积值转换为概率分布，**使得所有注意力权重的和为1。**



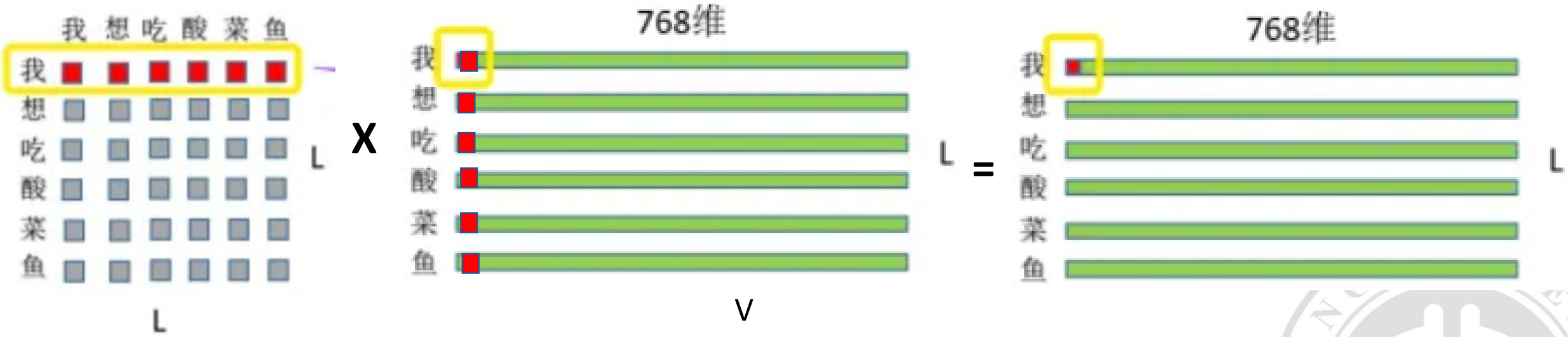
自注意力机制

- 应用注意力汇聚：加权求和，将注意力权重矩阵与值矩阵（V）相乘。




```
Attention(Q, K, V) = softmax(Q * K^T / sqrt(d_k)) * V
```

自注意力机制



用每个字的权重对每个字的特征进行加权求和

自注意力机制

- 打分函数：加性注意力（Additive Attention）
  - 加性注意力是一种用于计算 查询和键 之间相似度的方法。它首先将查询和键分别经过线性变换（全连接层），然后将两者相加，并通过非**线性激活函数**（如tanh）计算相似度。接下来，使用**可学习的权重向量**计算最终的相似度分数。

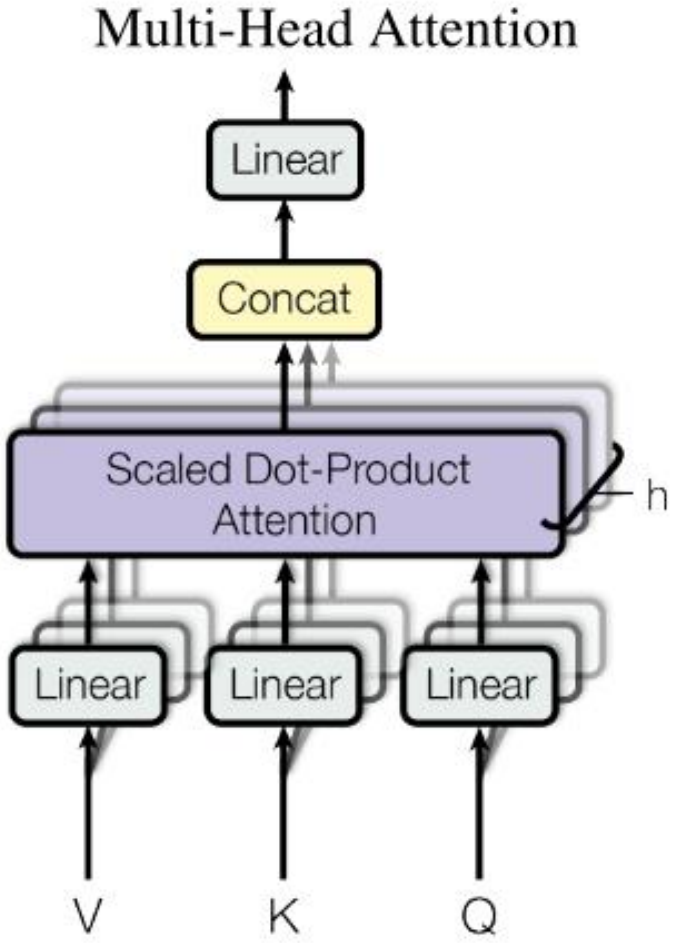
$$Attention(Q, K, V) = softmax(tanh(Q * W_q + K * W_k) * V_a^T) * V$$

- Attention(Q, K, V)：表示注意力汇聚的输出结果。
- Q：表示查询（Query）矩阵。
- K：表示键（Key）矩阵。
- V：表示值（Value）矩阵。
- W_q：表示查询的线性变换矩阵。
- W_k：表示键的线性变换矩阵。
- **V_a：表示可学习的权重向量。**
- +：表示矩阵相加。
- **tanh：表示应用tanh非线性激活函数。**

Multi-head Attention

多头注意力

- 多头注意力（multihead attention）
  - h个self-attention组合
  - 映射到h个空间，关注不同的特征维度
  - 将这h个注意力汇聚的输出拼接在一起，并且通过另一个可以学习的线性投影进行变换，以产生最终输出。



多头注意力

- 对于一个输入序列，假设其长度为 N，**嵌入维度（特征维度）为 D**，头数为 h（num_heads），则对于一个多头注意力（multi-head attention）计算过程中各项的形状如下：
- 输入矩阵：
  - Q = K = V，形状为：(N, D)
- 对于一共h个注意力头，i表示注意力头的索引
- 每个注意力头独立的QKV权重矩阵：

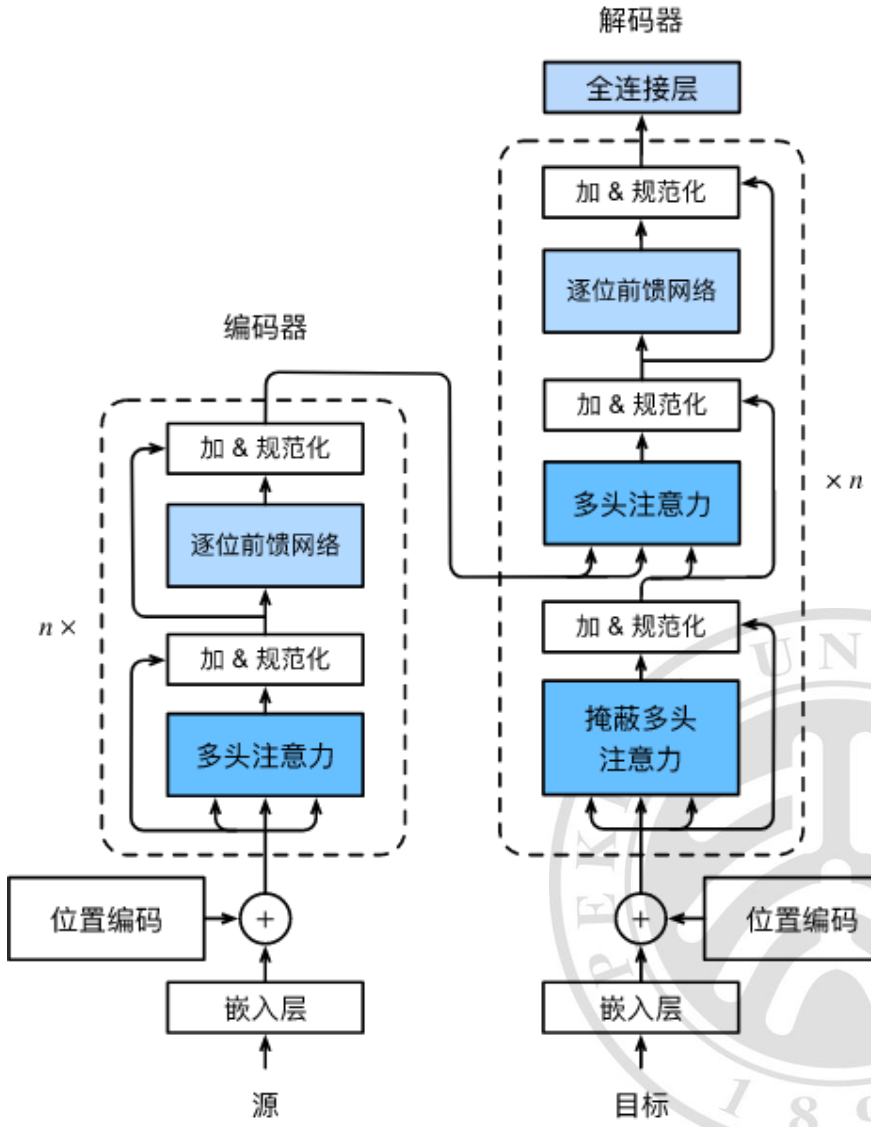
  W_{Qi}、W_{Ki}、W_{Vi}，形状为：$(D, head\_dim)$

  其中 head_dim = D/h
- 线性变换后的查询、键和值矩阵：
  - Q_i、K_i、V_i，形状为：$(N, head\_dim)$
- 缩放点积注意力计算中的相似度矩阵：
  - $Qi * K_i^T$，形状为：(N, N)
- 缩放点积注意力的输出：
  - $Attention_i(Qi, Ki, Vi)$，形状为：$(N, head\_dim)$
- 多头注意力的拼接输出：
- Concat_Attention，形状为：(N, D)，因为我们拼接了 h 个头，每个头的维度为 head_dim，**总维度为 h * head_dim = D。**
- 线性变换后的输出矩阵：
  - W_O，形状为：(D, D)
- 多头注意力的输出：
  - $Multihead\_Attention(Q, K, V)$，形状为：$(N, D)$

Transformer

Transformer

- Transformer： 整体架构
  - 整体结构由编码器（Encoder）和解码器（Decoder）两部分组成
  - 编码器和解码器都是由多个相同的层组成，每层中包含了一个**多头注意力机制**和一个**全连接前馈网络**，这些层之间通过**残差连接和层归一化（Layer Normalization）**相连。



位置编码

- 位置嵌入

- 回忆：词嵌入和RNN
  - Embedding序列

- 句子 = 词 + 位置顺序排列
- 词信息 >>> 词嵌入
- 位置信息 >>> 位置嵌入

- 表达句子的向量序列 = 词嵌入向量 + 位置嵌入向量

对于一个序列长度为N，嵌入维度为d_{model}的输入，位置嵌入矩阵的形状为(N, d_{model})。每个位置嵌入向量的形状为$(d_{model},)$。

$$\mathrm{PE}_{(pos,2i)} = \sin(pos/10000^{2i/d_{\mathrm{model}}})$$
$$\mathrm{PE}_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\mathrm{model}}})$$

$\mathrm{PE}(pos, 2i)$

表示位置嵌入矩阵的第 pos 行、第 2i 列的值

$\mathrm{PE}(pos, 2i+1)$

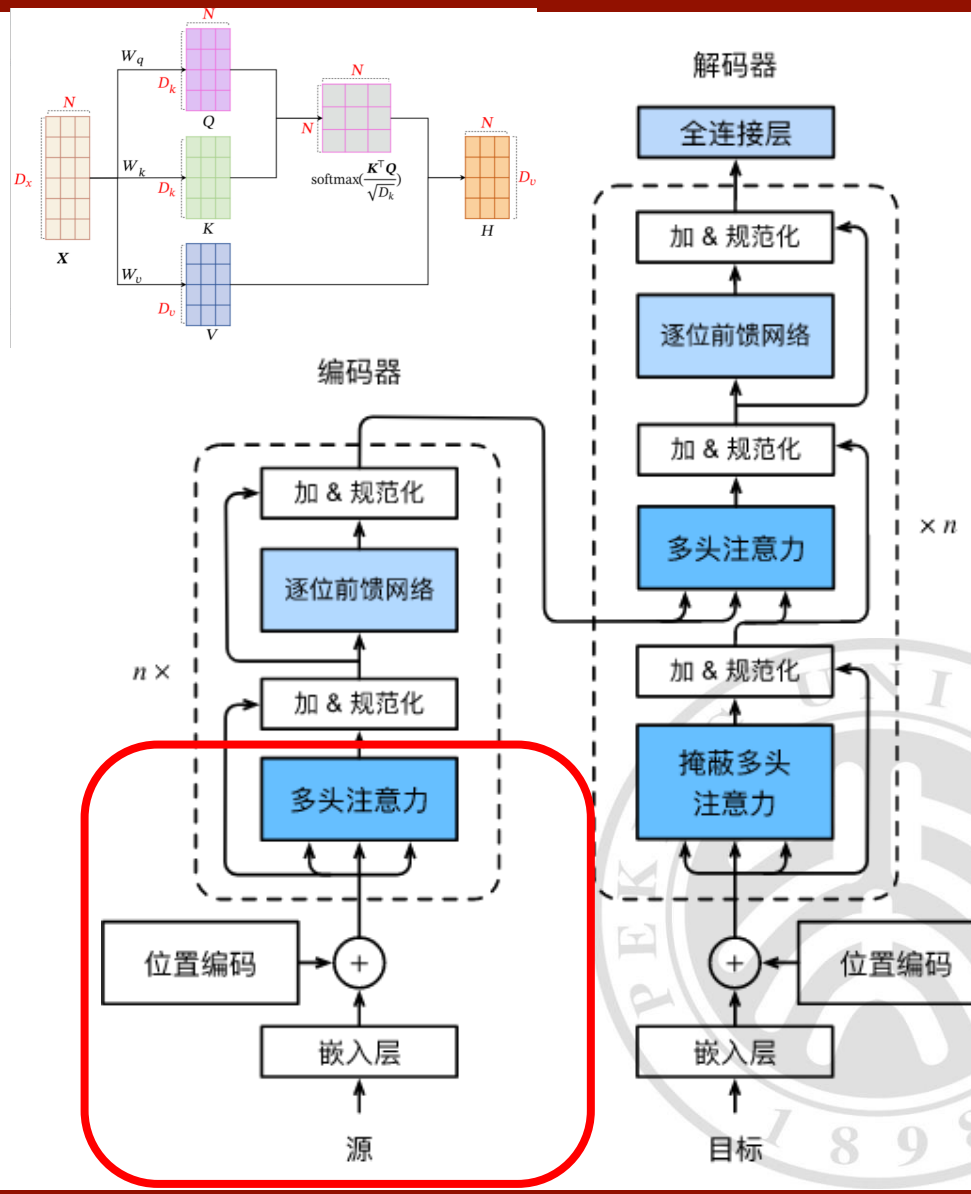
表示位置嵌入矩阵的第 pos$行、第 2i+1列的值。

i：是嵌入向量的索引

d_{model}：嵌入向量的维度，pos是位置编码的位置。

- **计算过程：编码器**

1. 源输入序列经过嵌入层的处理得到**词嵌入**，并加入**位置编码**，得到最终的输入向量。
2. 通过**多头注意力**的**h组**线性层对词嵌入表示进行变换，将其映射为三个不同的空间，得到查询矩阵Q、键矩阵K和值矩阵V。
3. 计算**缩放点积注意力输出**。使用Q、K、V三个矩阵，按照缩放点积注意力的计算公式进行计算，得到输出矩阵。这个输出矩阵包含了每个词向量对于其他所有词向量的注意力权重。



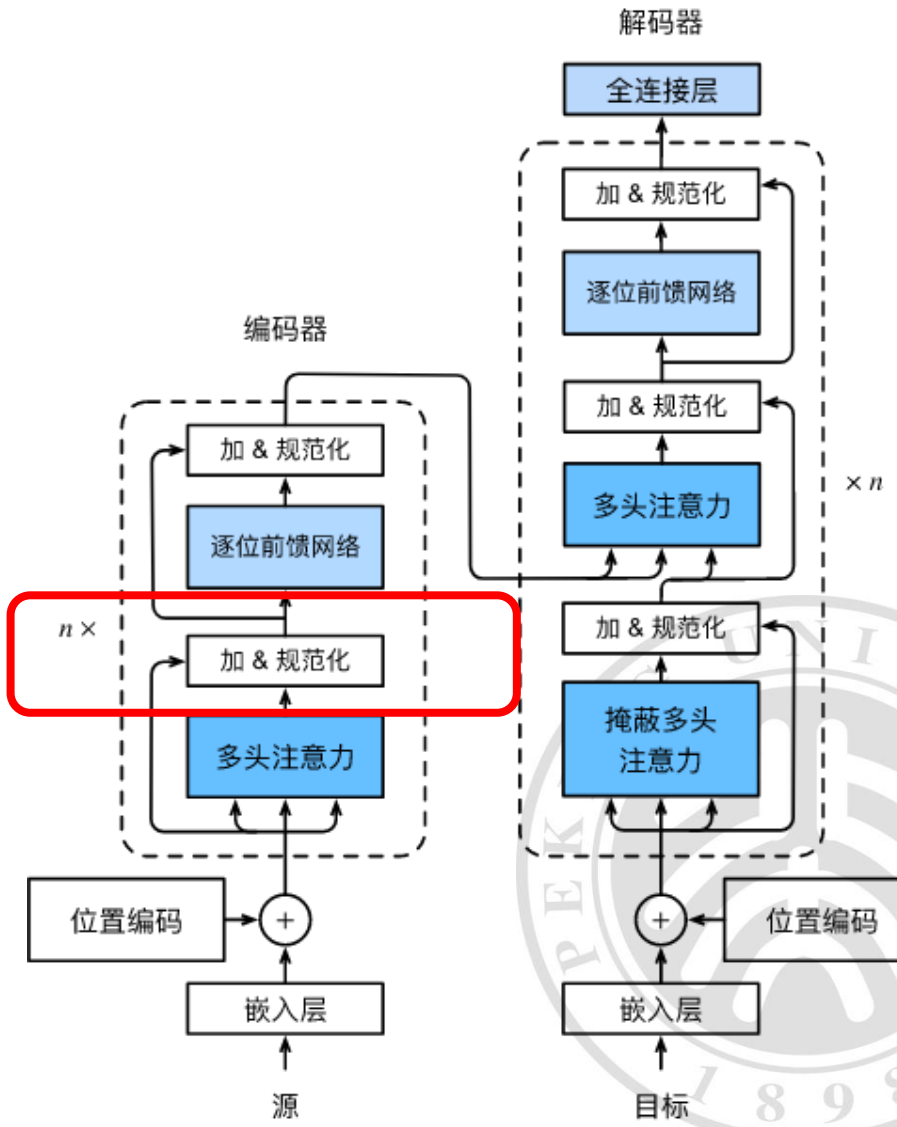
Transformer

- **计算过程：残差连接和加&层规范化（AddNorm）**

4. 将输入向量和注意力输出向量相加，得到残差连接的结果。

5. 对残差连接的结果进行**层规范化（Layer Normalization）**
   - 将每层的输出值归一化到均值为0、方差为1的范围内。
   - 层规范化将每个神经元的输出值 x减去它在这一层的均值μ，然后除以标准差 σ，最后乘以可学习的缩放因子 α

$$\mathrm{LayerNorm}(x) = \alpha \odot \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta$$



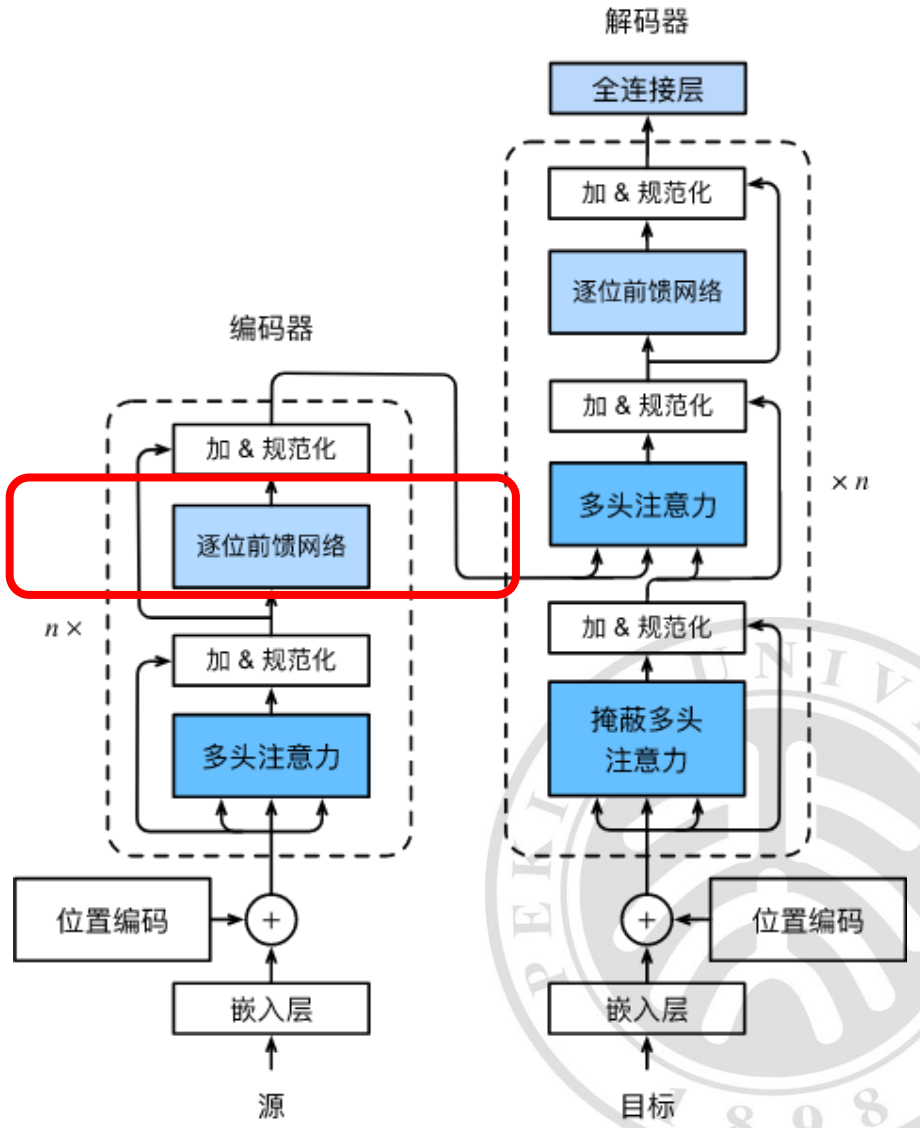
- **计算过程：逐位前馈神经网络**

6. 逐位前馈网络（Feed-Forward Network, FFN）用于在自注意力机制之后进行非线性变换。计算过程如下

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

其中，x表示输入向量，W_1和b_1是第一个全连接层的权重和偏置项，W_2和b_2是第二个全连接层的权重和偏置项。$\max(0, xW_1 + b_1)$表示激活函数ReLU的输出，对于任何小于0的值都将被设置为0。

逐位前馈网络由两个全连接层和一个ReLU激活函数组成。

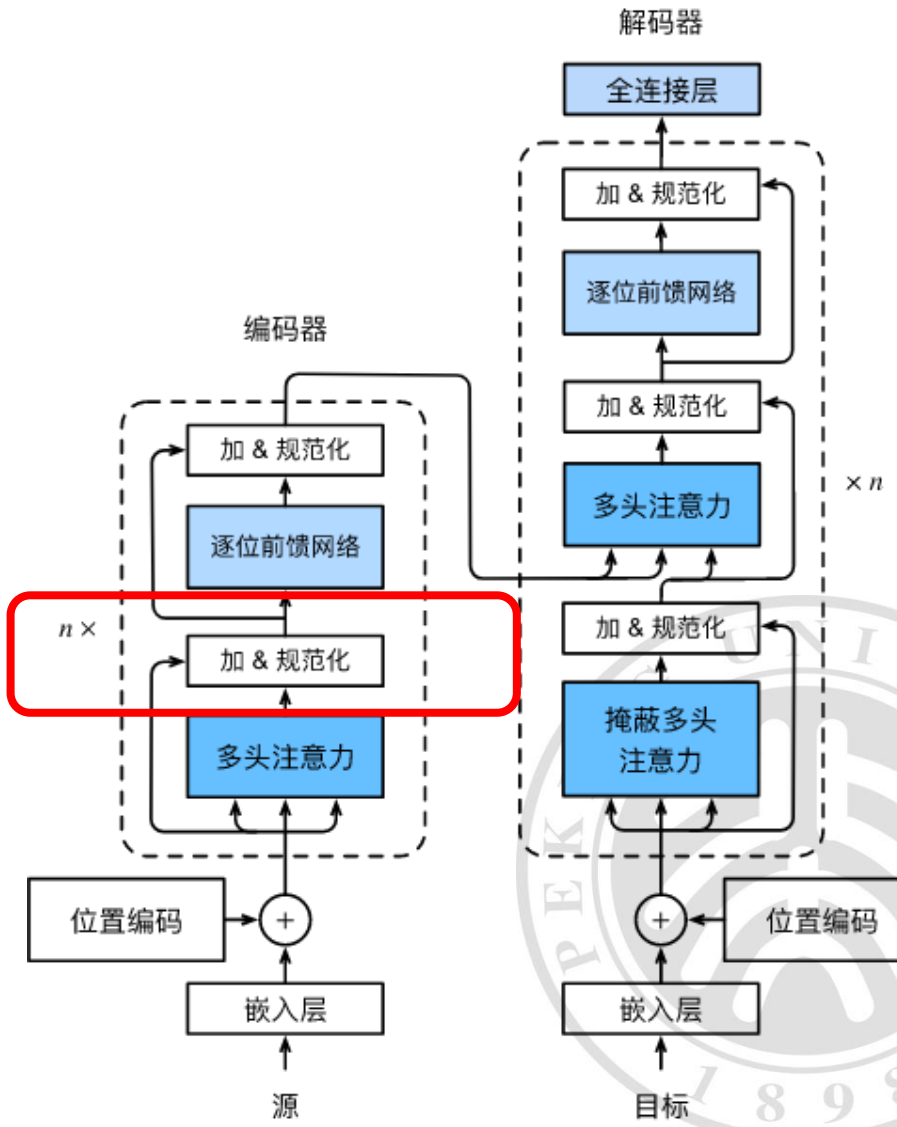


Transformer

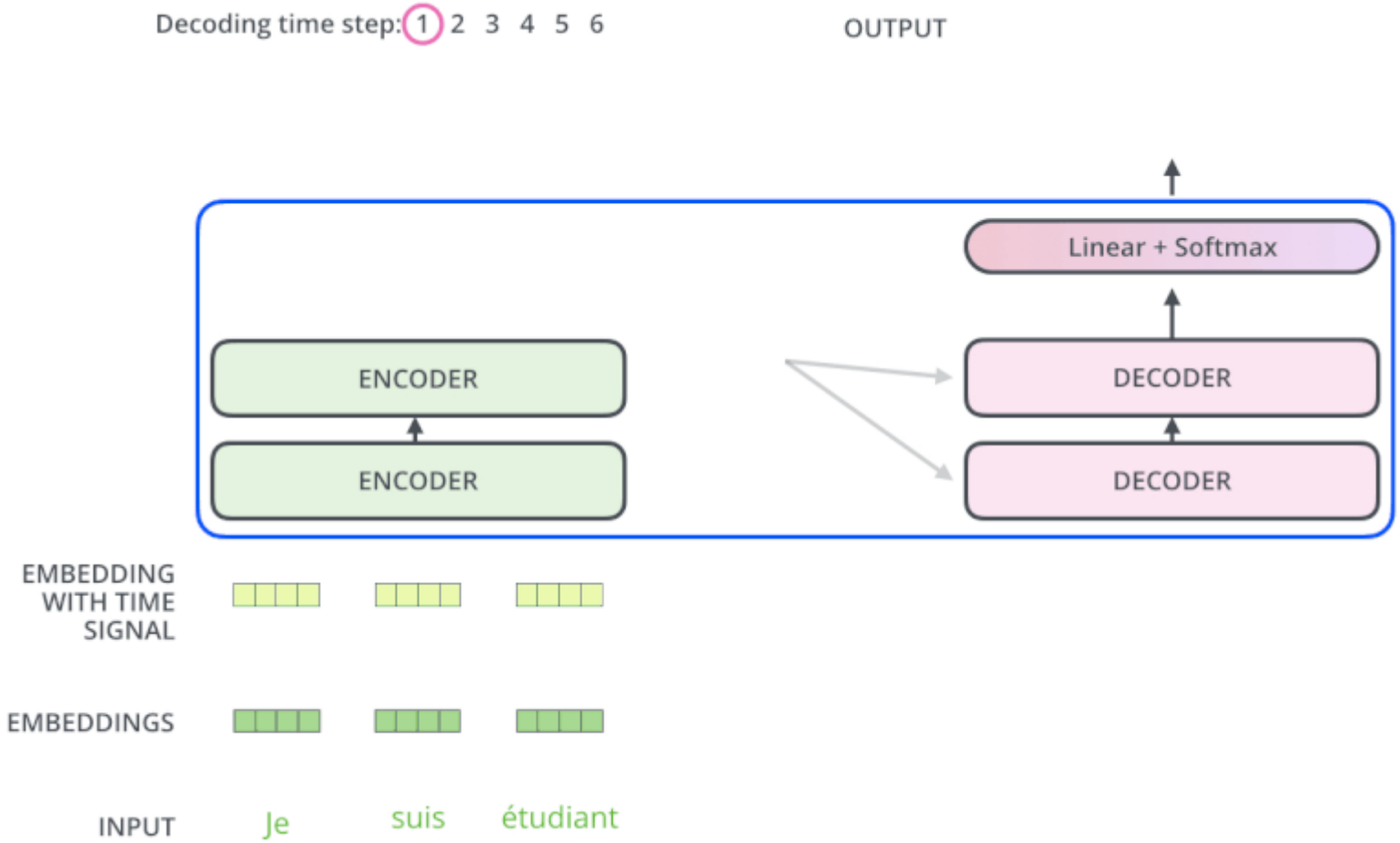
- 计算过程：多个编码器堆叠

$$\text{Transformer}(\mathbf{x}) = \text{Encoder}_n(\cdots(\text{Encoder}_2(\text{Encoder}_1(\mathbf{x}))))$$

7. 如果有 n 个编码器，则可以将它们依次串联起来。其中，第一个编码器的输入是词嵌入向量加上位置编码向量。
8. 最后一个编码器的输出就是**最终的编码表示**，它将用于传递给**解码器**进行下一步的处理。



Transformer

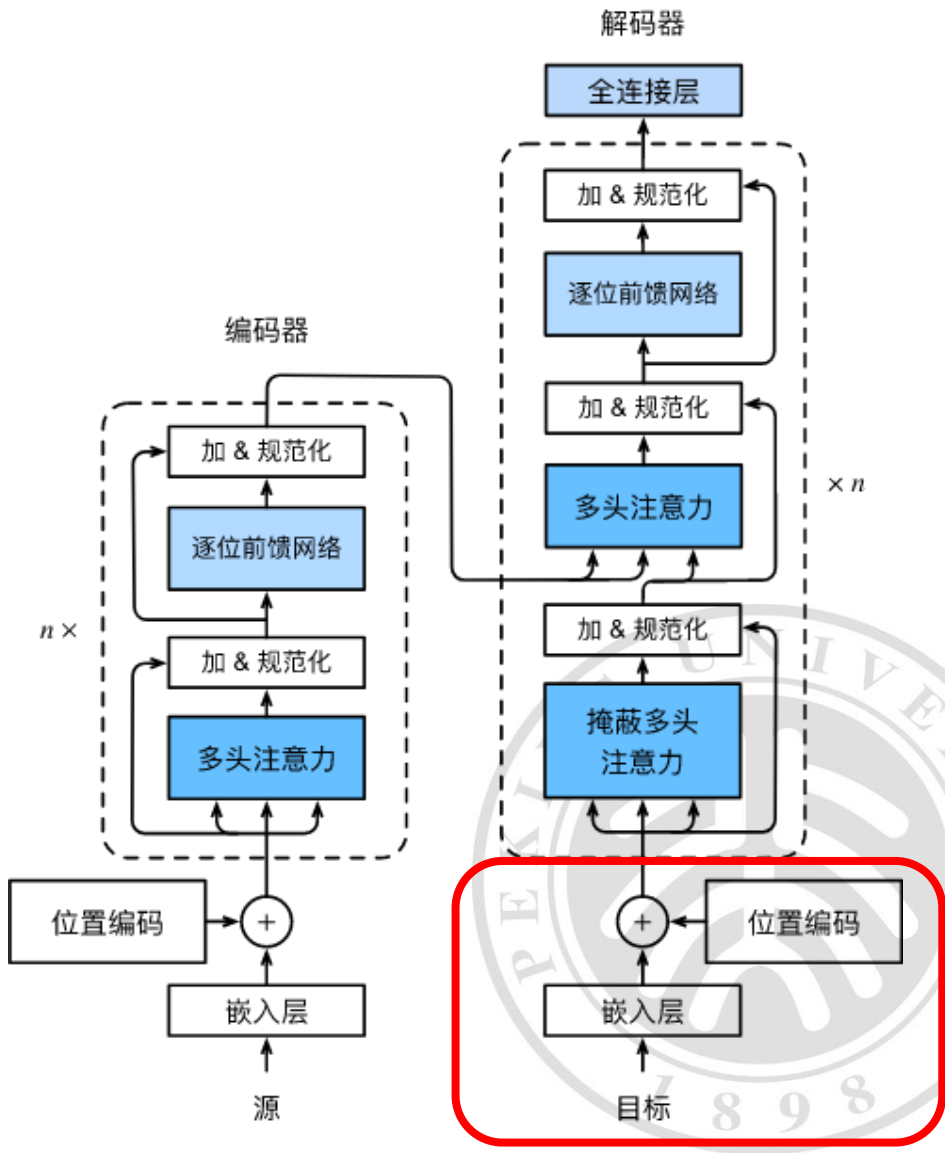


Transformer

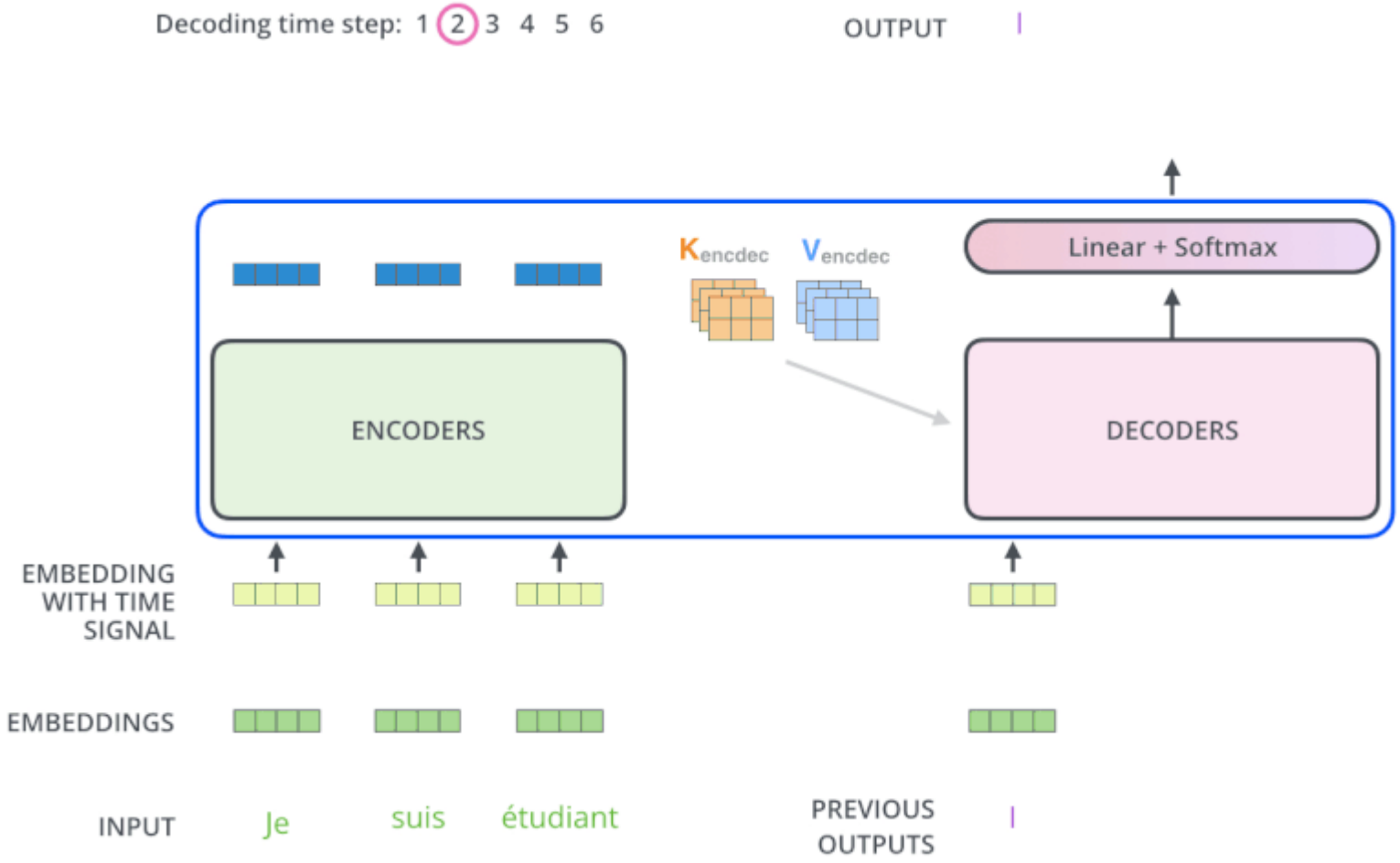
- **计算过程：解码器**

  encoder的输出并没直接作为decoder的直接输入

  - 解码器的输入被称为**目标序列**。初始decoder的time step为1时(也就是第一次接收输入)，其输入为一个特殊的token，即**目标序列开始的token(如<BOS>)**，也可能是其它视任务而定的输入等等，其目标则是预测翻译后的第1个单词(token)是什么；
  - 然后<BOS>和预测出来的第1个单词一起，再次作为decoder的输入，得到第2个预测单词；3后续依此类推。直到遇到特殊的**结束token（如<EOS>）**或者达到最大输出长度为止。



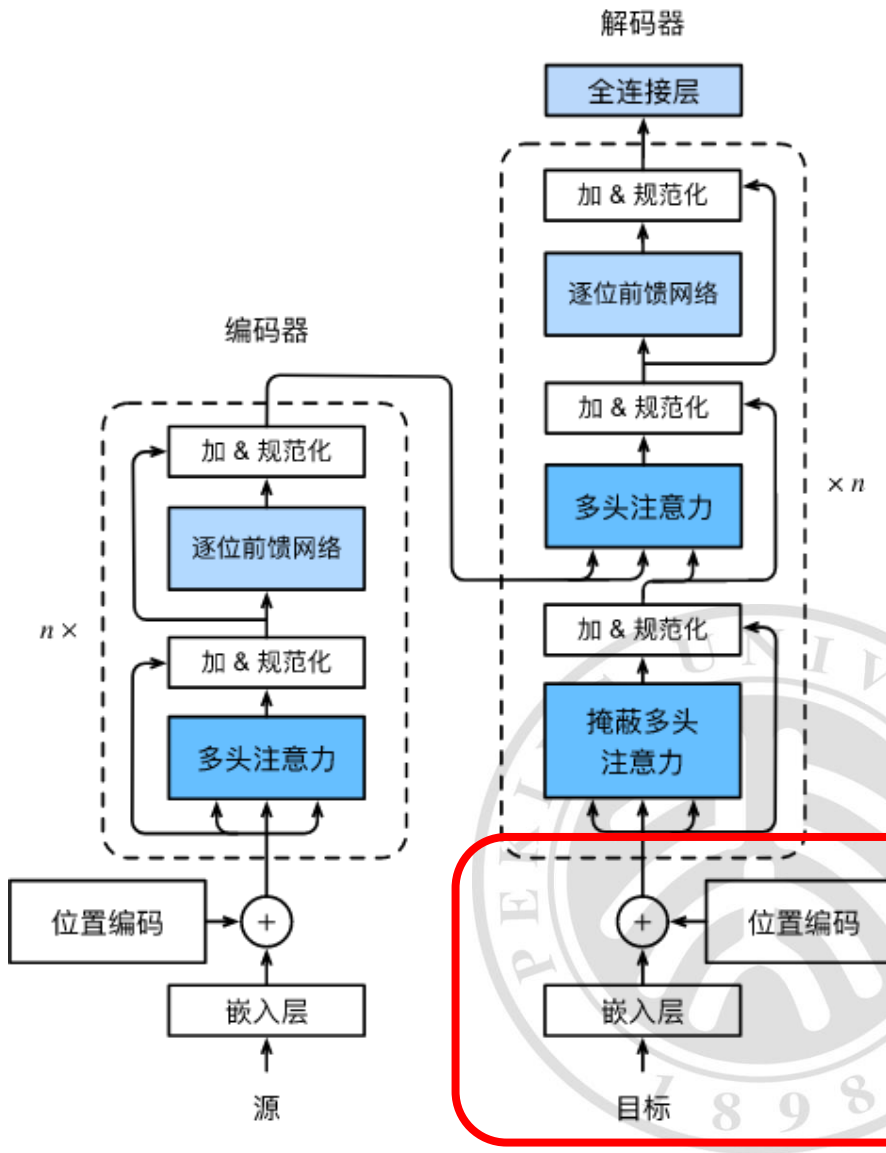
Transformer



Transformer

- **计算过程：解码器**

1. 目标序列嵌入和位置编码：与编码器类似，目标序列首先经过嵌入层的处理得到词嵌入，并加入位置编码，得到最终的输入向量。

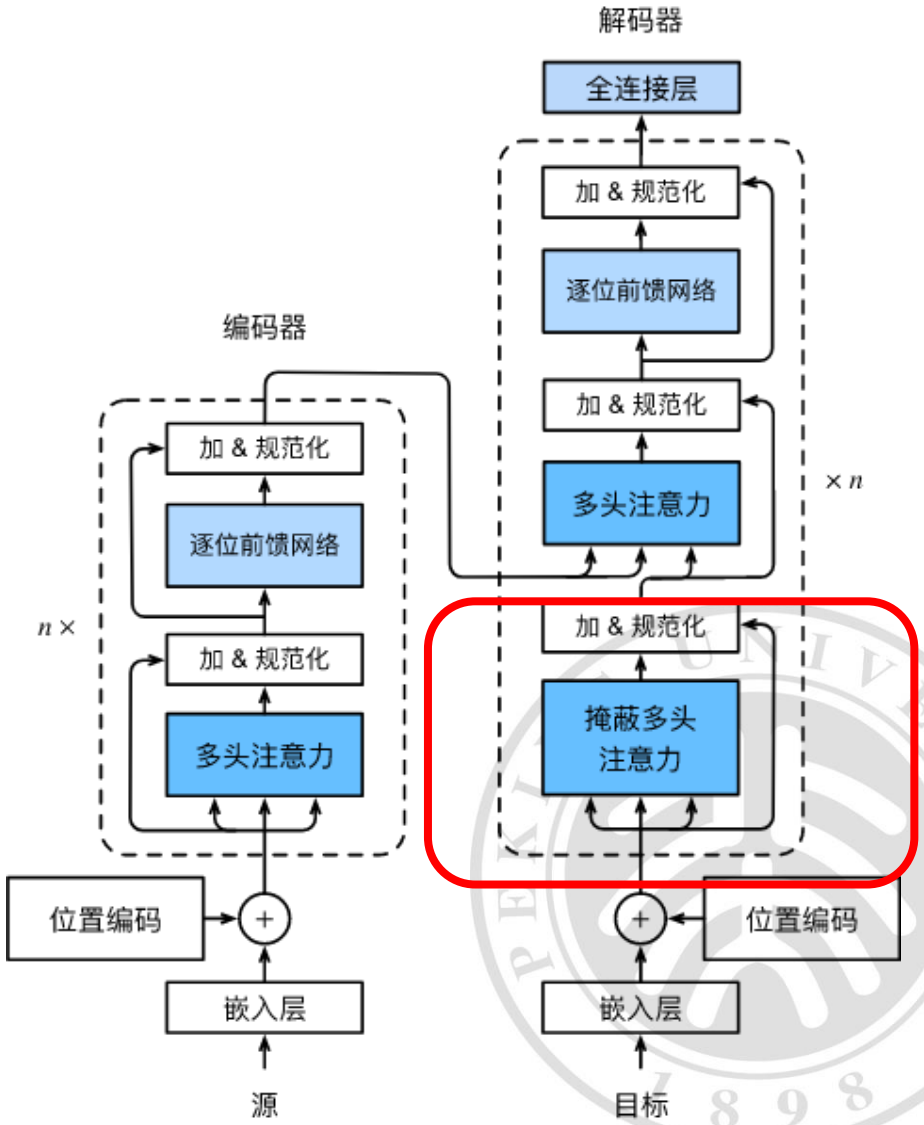


Transformer

- 计算过程：

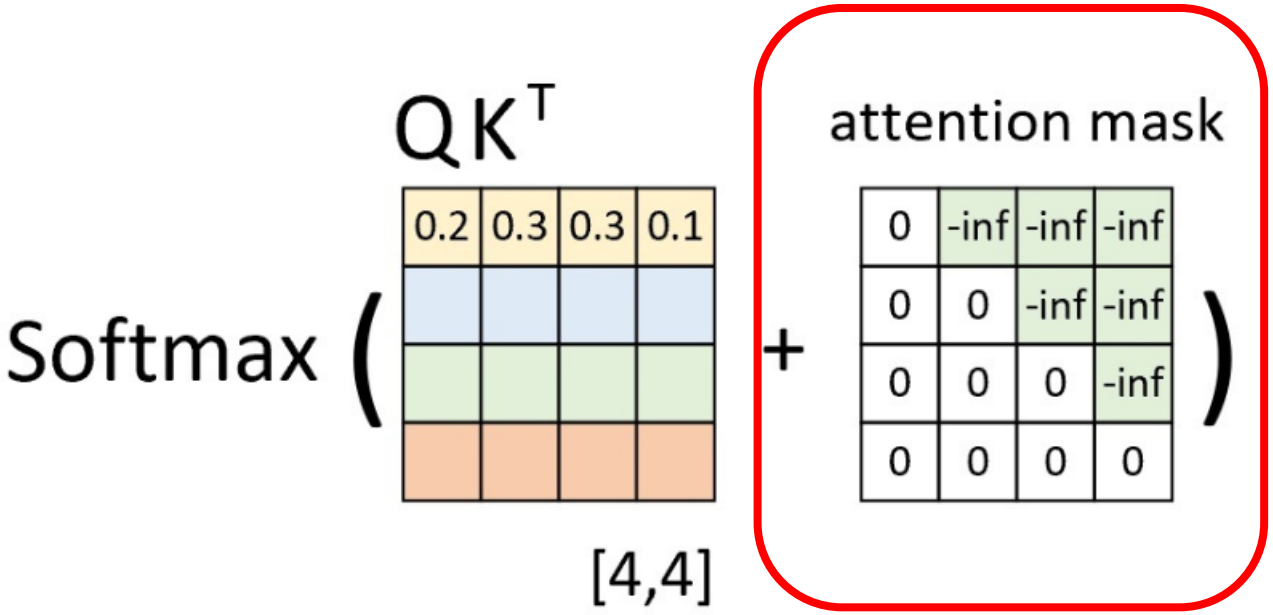
2. 通过**掩蔽多头注意力层**（masked multi-head attention）生成解码器自注意力矩阵，用于捕捉当前解码器状态与之前解码器状态的依赖关系。具体计算过程与编码器的自注意力矩阵计算过程类似。

Mask机制屏蔽了未来的序列信息。



掩码

- **掩码（Mask）**
- 防止在输入序列中的某些位置上的信息泄露到当前位置上。
- 在语言建模任务中，需要在**预测当前词时，只利用前面的词，而不使用后面的词。**
  - 创建一个掩码矩阵，形状为 (N, N)，其中 N 是输入序列的长度。
  - 对于掩码矩阵中的每个元素，如果该**元素对应的位置是要掩盖的位置，则将其设置为负无穷（-inf），否则将其设置为0。**
  - 在softmax计算之前，**将掩码矩阵与注意力分数相加**，这样，被掩盖的位置的注意力分数就变成了负无穷，softmax之后，它们**对应的注意力权重就会变成0**，即不会对当前位置产生影响。

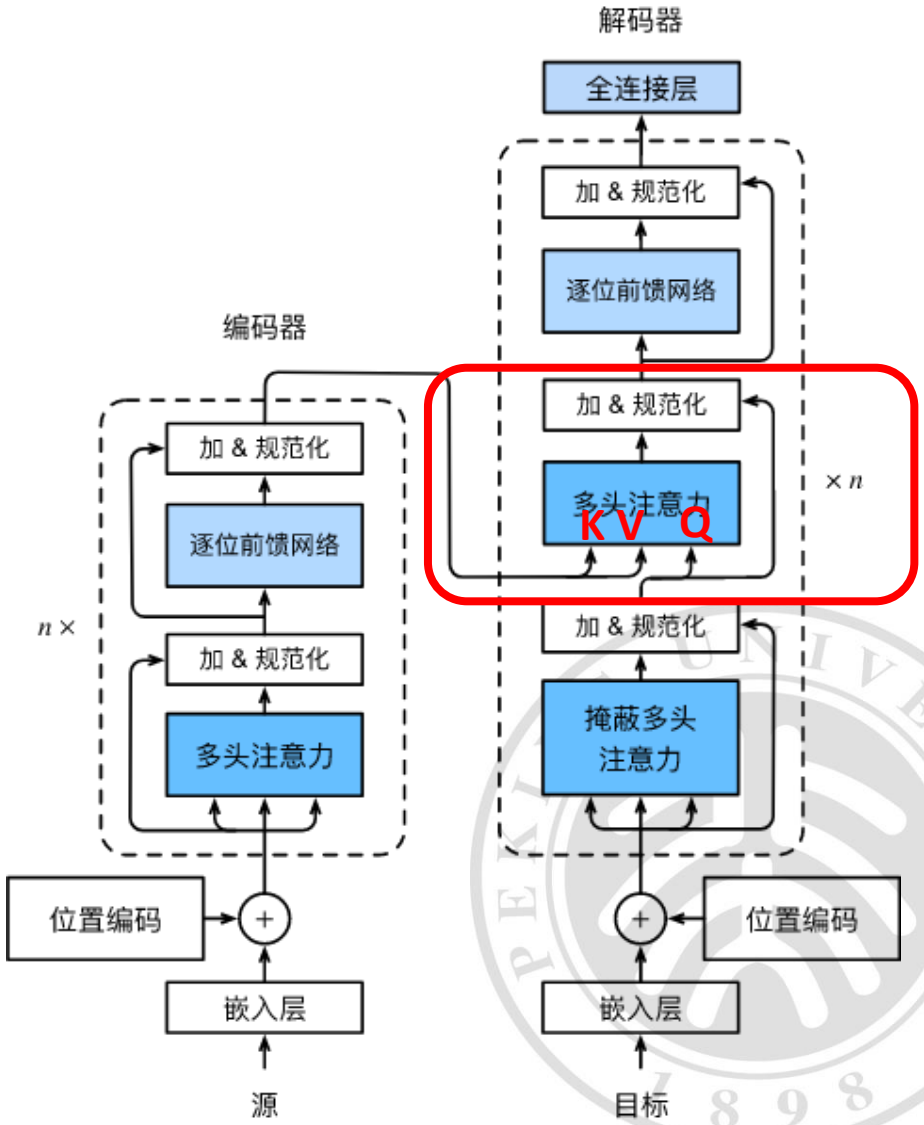


$$Attention(Q, K, V, mask) = softmax(\frac{QK^T}{\sqrt{d_k}} + mask)V$$

- 计算过程：编码器-解码器注意力

3. 接下来**编码器-解码器注意力层**（encoder-decoder attention）将编码器输出的信息有效地融入到当前解码器状态中，以帮助解码器更好地进行下一步预测。

在编码器—解码器注意力中，Q来自前一个解码器层的输出，而K和V来自编码器的输出。



Transformer

- **计算过程：**

4. 接下来**逐位前馈网络和**AddNorm与解码层一致。

5. 堆叠n个解码器，前一个解码器的输出和对应的编码器的输出是当前解码器的输入

第 1 个解码器层：

$$Out_0 = PositionEncoding(Y)$$

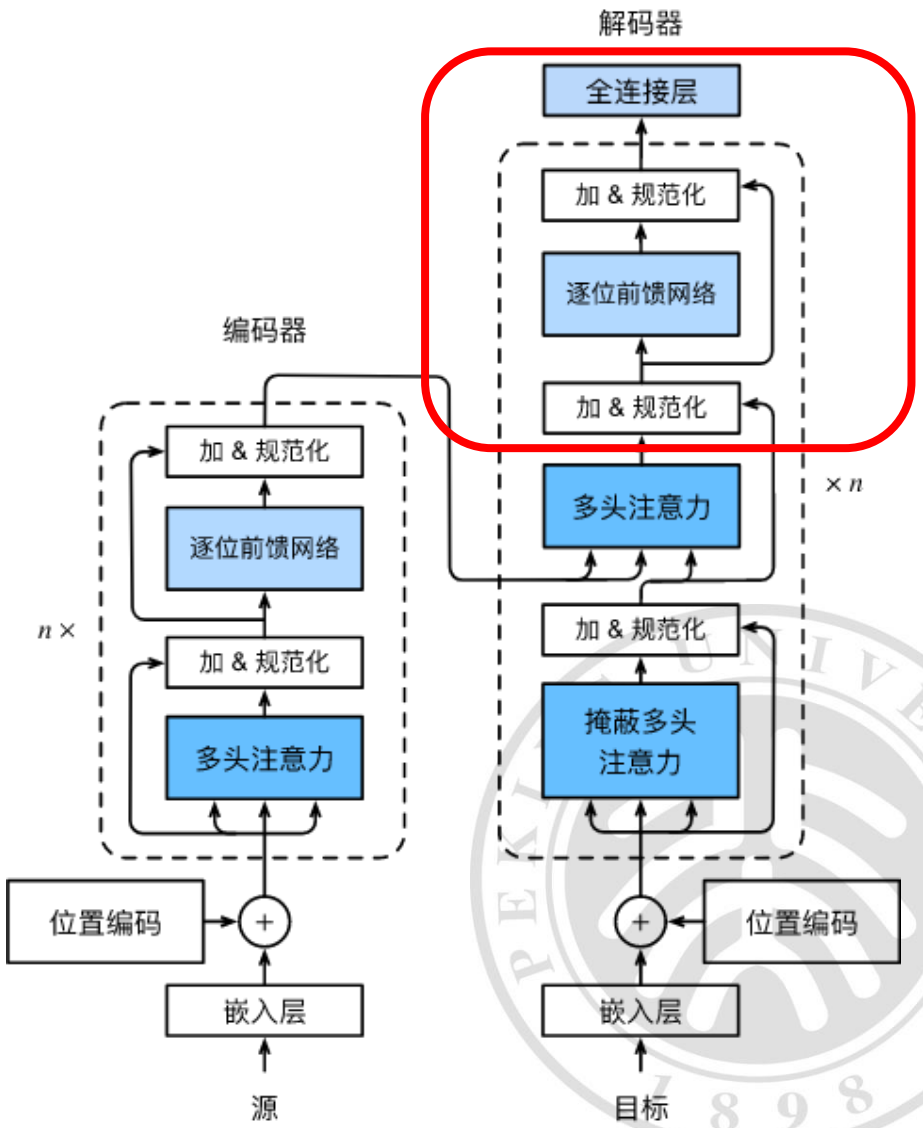
$$Out_1 = decoder_1(Out_0, E)$$

第 i 个解码器层（1 < i <= n）：

$$Out_i = decoder_i(Out_{i-1}, E)$$

最后，我们可以对最后一个解码器层的输出施加Softmax函数来计算词汇表的概率分布：

$$P_vocab = Softmax(W_out * Out_n + b_out)$$

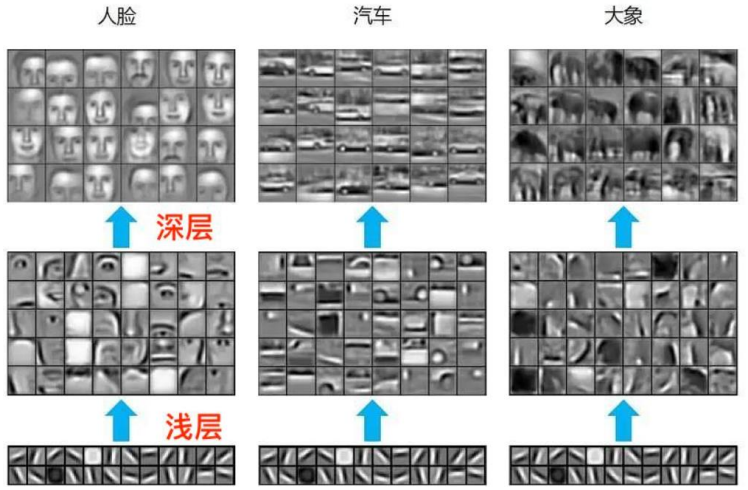


Pre-trained Models

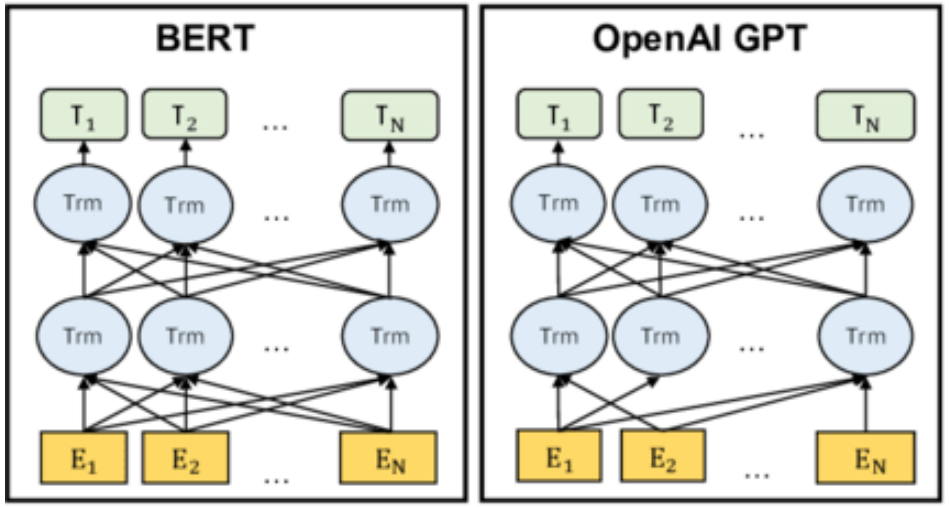
NLP的预训练模型

- 预训练模型是一种强大的技术，用于在**无标签数据**上学习**通用语言表示**，从而提高NLP 任务的性能。

1. 预训练阶段：在大量无标签文本数据上进行预训练，学习通用的语言表示。
2. 微调阶段：针对特定任务进行**微调fine-tune**，使模型适应各种 NLP 任务。



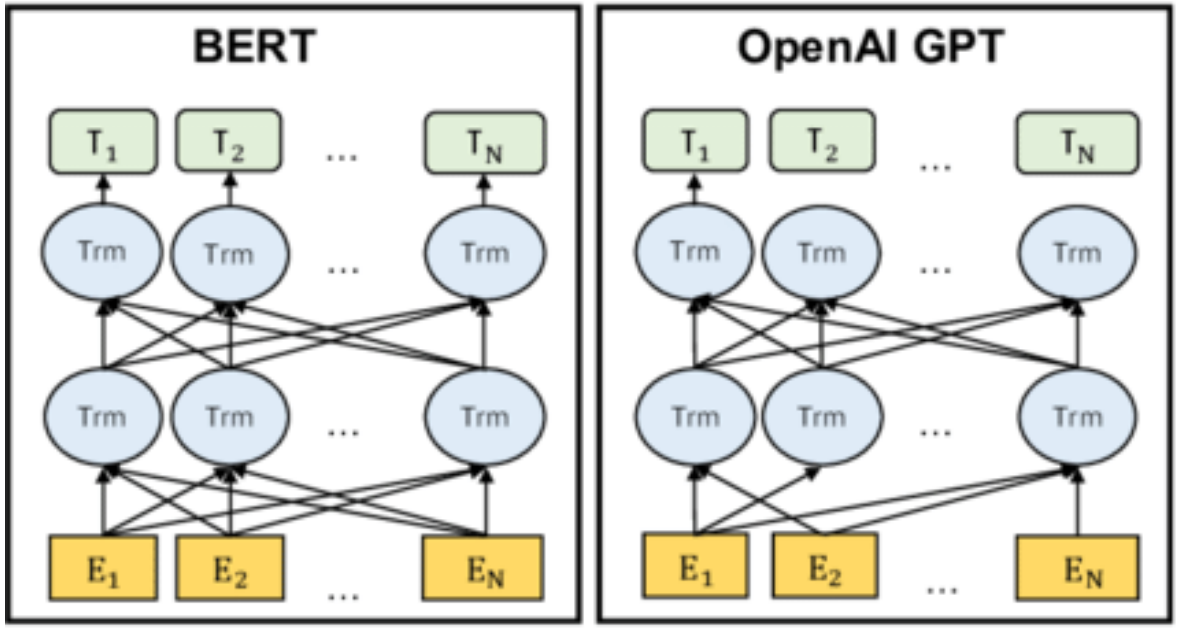
通用提取图片特征



通用语言表示

NLP的预训练模型

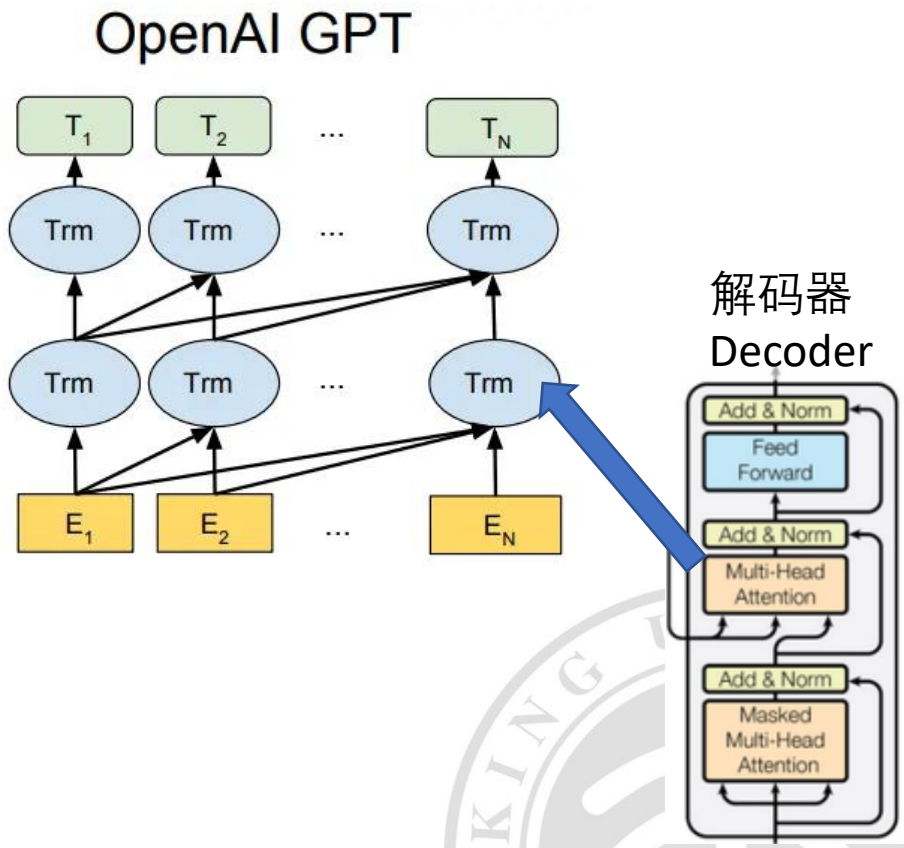
- 常见预训练模型
  - GPT：基于 Transformer 的预训练生成模型，学习**单向**上下文信息。
  - BERT：基于 Transformer 的双向编码器表示模型，学习**双向**上下文信息。



GPT

- Generative Pre-Trained Transformer

- GPT 是一种基于 Transformer 的预训练**生成式语言模型**，由 OpenAI 提出。
- GPT 采用单向上下文表示，适用于生成式任务。
- 特点：
  - 预训练：在大量**无标签文本数据**上进行预训练，学习通用的语言表示。
  - **单向**上下文表示：仅使用左侧的上下文信息来预测当前词。
  - 使用 多个Transformer 的**解码器**堆叠
  - 自回归训练：预测下一个词的概率分布，即 P(wi|w1,⋯,wi−1)。



$$L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k},\ldots,u_{i-1};\Theta)$$

Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training[J]. 2018.

GPT

- GPT的预训练阶段
- 无监督：无标注文本数据
- 目标：根据上文，最大化对下一个词的预测概率

$$L_1(U) = \sum_i log\, P(u_i \mid u_{i-k}, \cdots, u_{i-1}; \Theta)$$

- 计算过程：

模型的输入向量h0 为：

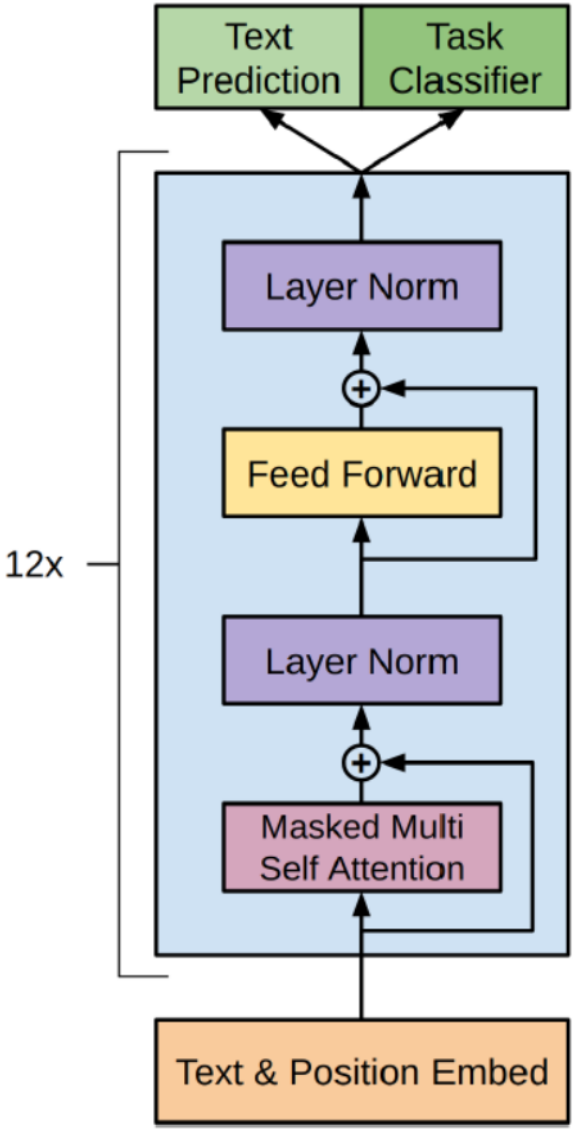
$$h_0 = UW_e + W_p$$

其中，$U = (u_{-k}, \cdots, u_{-1})$表示的表示当前词 u 的 k 个上文词，W e 是词向量矩阵，W p 表示的是位置向量矩阵，h_0表示当前词及其位置信息的组合。

经过多个Transformer层后，第l层的输出就是：

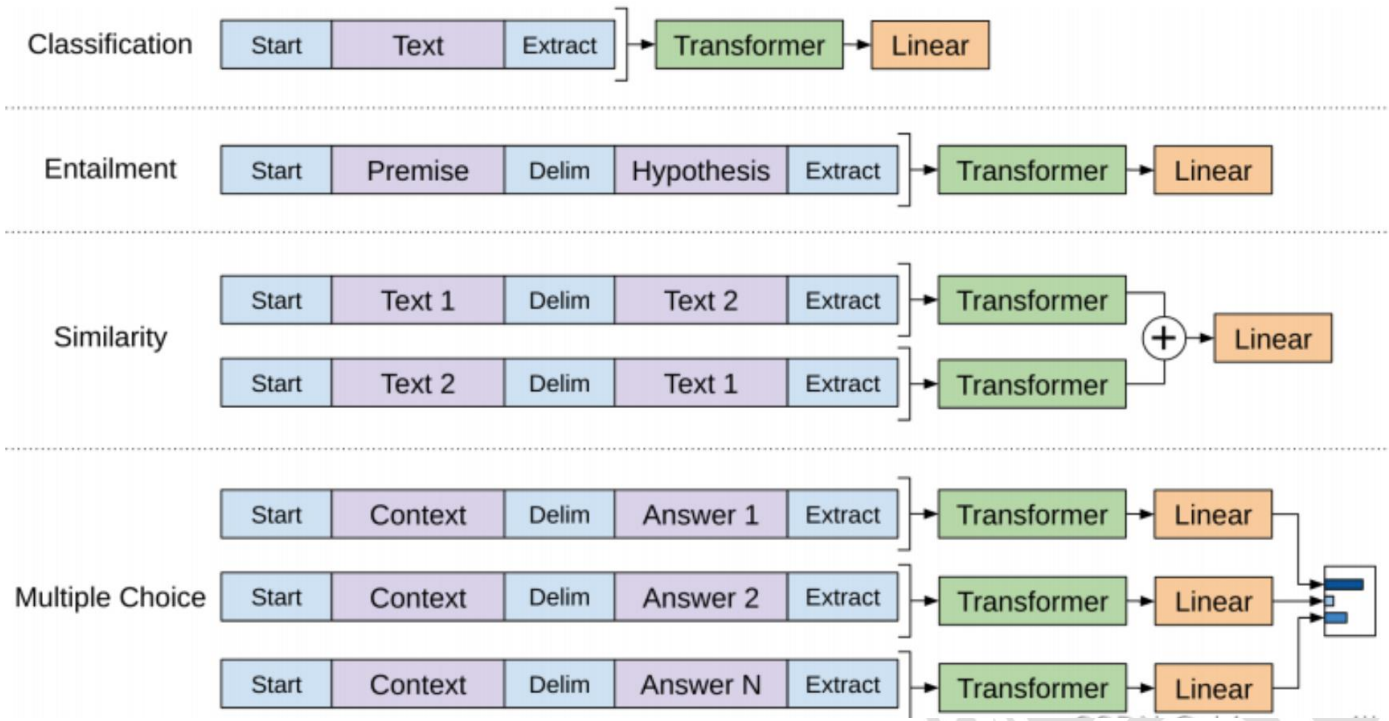
$$h_l = transformer\_block(h_{l-1})$$

最终输出当前词的位置是词u 的概率：$P(u) = softmax(h_n W_e^T)$



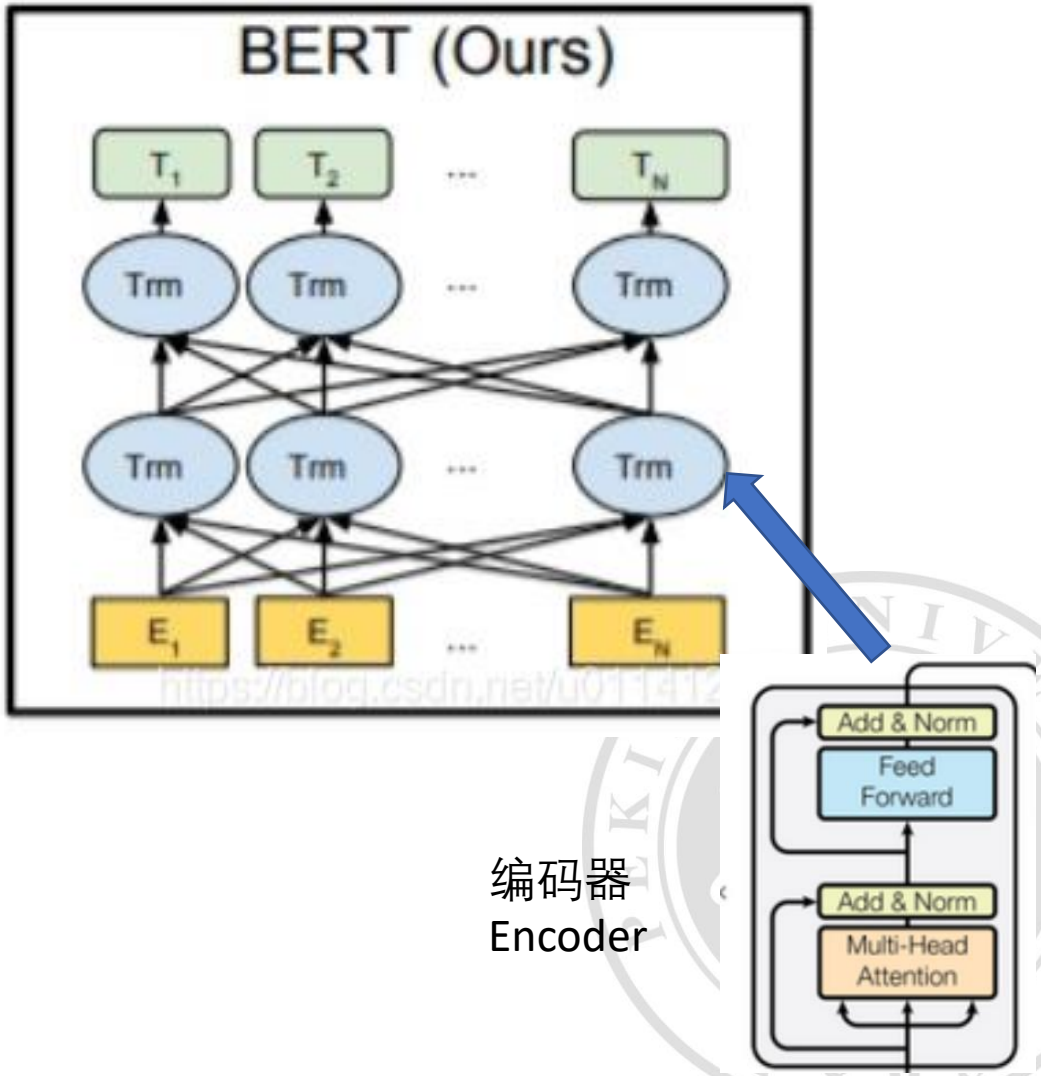
GPT

- GPT 应用于 NLP 任务
- 微调阶段
  - 为特定任务添加任务相关的头部 Head（如分类层、序列标注层等）。
  - 使用标注的任务数据心理进行微调，优化任务相关的损失函数。
- 应用示例
  - 文本分类：情感分析、主题分类等。
  - 文本生成：生成连贯的文本、摘要、翻译等。
  - 命名实体识别：识别文本中的实体类型，如人名、地点等。



BERT

- Bidirectional Encoder Representations from Transformers (BERT)
- BERT 是一种基于 Transformer 的预训练语言模型，由 Google 提出。
- BERT 采用双向上下文表示，适用于各种 NLP 任务。
- 特点：
  - 预训练：在大量无标签文本数据上进行预训练，学习通用的语言表示。
  - **双向上下文表示**：同时使用左侧和右侧的上下文信息来预测当前词。
  - 使用多个 Transformer 的**编码器**堆叠
  - Masked Language Model（MLM）训练：**预测被遮盖词的概率分布**，即 P(wi|w1,⋯,wi−1,wi+1,⋯,wn)。

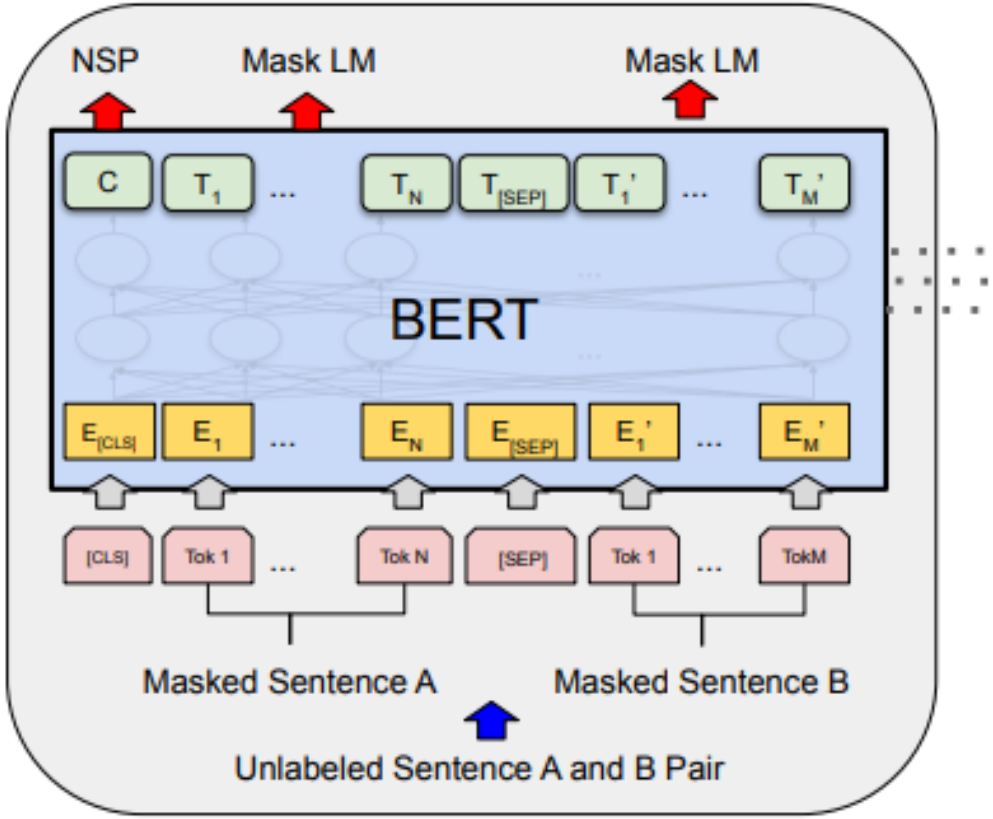


编码器
Encoder

Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

BERT

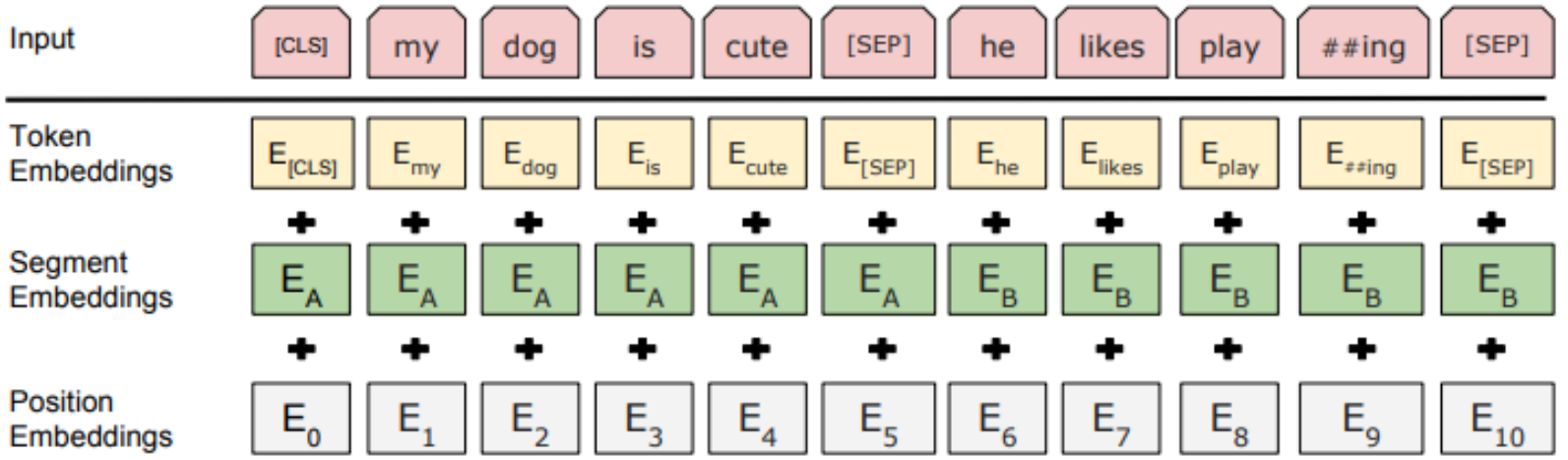
- BERT 预训练阶段
- 两个任务:
  - MLM（Masked Language Modeling）:
    - BERT接受双向的信息，挖掉一些需要预测的词让BERT补全，类似**完形填空**(Cloze)
    - 输入数据中随机选择15%的词用于预测，这些词中:
    1. 80%的词向量输入时被替换为<MASK>
    2. 10%的词的词向量在输入时被替换为其他词的词向量
    3. 另外10%保持不动
  - NSP(Next Sentence Prediction)



Pre-training

BERT

- NSP任务：
  - 往Transformer中输入连续的两个句子，左边的句子前面加上一个<CLS>标签，它的输出被用来判断两个句子之间是否是连续上下文关系。采用负采样的方法，正负样本各占50%。
- Segment Embedding（段向量）用于表示句子的位置信息，使模型能够区分和处理多个句子



BERT

1. Sentence Pair Classification

1. 输入：[CLS] 句子 A [SEP] 句子 B [SEP]
2. 输出：[CLS] 对应的输出向量
3. 应用：自然语言推理、文本蕴含等
4. 层：线性分类层

2. Single Sentence Classification

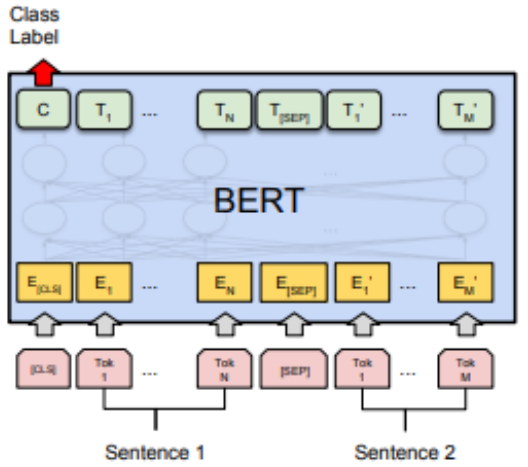
1. 输入：[CLS] 句子 [SEP]
2. 输出：[CLS] 对应的输出向量
3. 应用：情感分析、文本分类等
4. 层：线性分类层

3. Question Answering (QA)

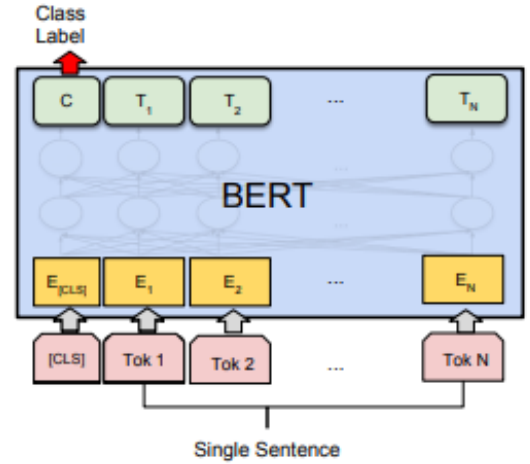
1. 输入：[CLS] 问题 [SEP] 段落 [SEP]
2. 输出：BERT 编码后的每个位置
3. 应用：SQuAD 等问答任务
4. 层：线性层（起始位置和结束位置）

4. Single Sentence Tagging

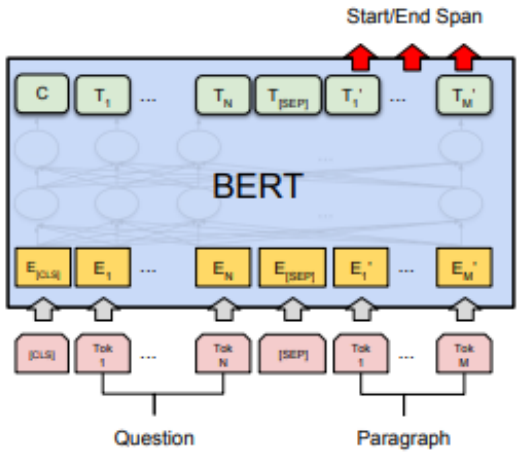
1. 输入：句子 [SEP]
2. 输出：BERT 编码后的每个词
3. 应用：命名实体识别、分词等
4. 层：线性层（标签预测）



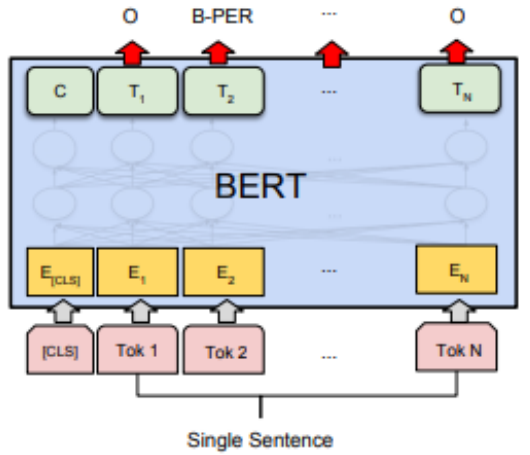
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

大模型进化史

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Transformer 2017

GPT 2018.06

BERT 2018

...

GPT-2 2019.11

RoBERTa 2019

...

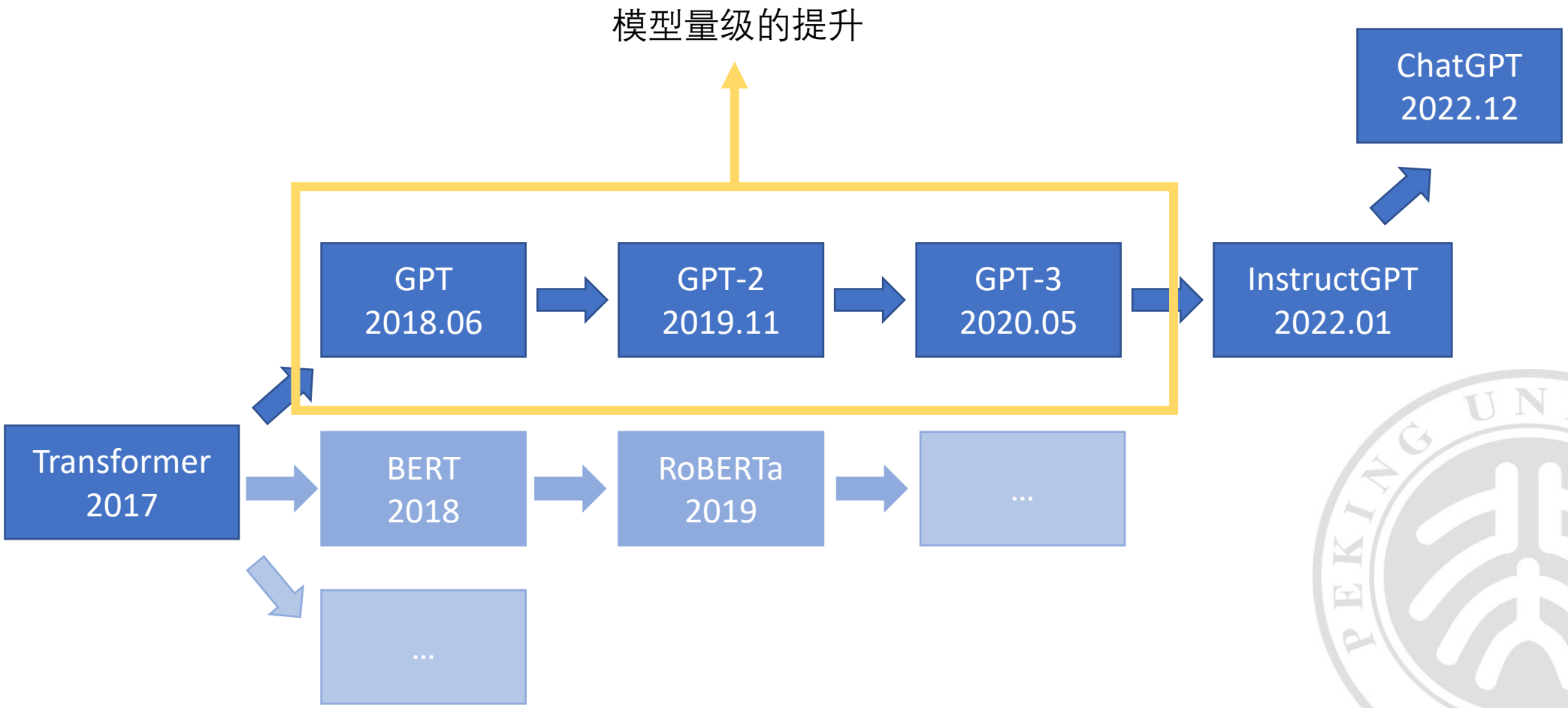
GPT-3 2020.05

InstructGPT 2022.01

ChatGPT 2022.12

模型结构的选择

大模型进化史


模型量级的提升
ChatGPT
2022.12
GPT
2018.06
GPT-2
2019.11
GPT-3
2020.05
InstructGPT
2022.01
Transformer
2017
BERT
2018
RoBERTa
2019
...
...

大模型进化史

- **参数规模提升带来能力“涌现Emergent”**
  - 大型语言模型（LLM）在训练过程中学到的一种自发性的任务完成能力
  - 模型基本结构和训练方式基本不变，只增大模型和数据规模，训练出的模型“智能”程度明显提高
- 原因
  1. **大量的训练数据**：人类积累的所有信息
  2. **模型容量**：充分学习数据
  3. **自回归和无监督训练**：不需要标注海量数据
  4. **迁移学习和微调**：适应不同任务
  5. **多任务学习**：提高泛化性

模型	发布时间	层数	头数	词向量长度	参数量	预训练数据量
GPT-1	2018年6月	12	12	768	1.17亿	约5GB
GPT-2	2019年2月	48	-	1600	15亿	40GB
GPT-3	2020年5月	96	96	12888	1750亿	45TB

回顾总结

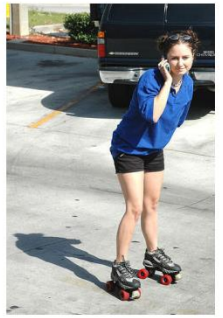
- 动机
  - 长度依赖问题，人脑的注意力机制，自主性提示/非自主性提示
- 自注意力机制 Self-Attention
  - QKV, 注意力汇聚, 打分函数，注意力矩阵, 缩放点积, 加性函数
- 多头注意力 Multihead Attention
  - 映射到h个空间
- 掩码和位置编码 Mask & Positional Embedding
  - -inf，句子=词向量+位置向量
- Transformer
  - 编码器，解码器，层规范化LayerNorm
- 预训练模型
  - 无监督，GPT，BERT，MLM，NSP，Fine-tune
- 大模型进化史

作业

- 作业1:
- 用循环神经网络在 IMDB 数据集上实现电影评论文本情感识别
- 要求:
  1. 阅读示例代码，在此基础上修改网络结构
  2. 使用nn.RNN,nn.LSTM,nn.GRU等接口搭建模型，训练后测试集准确率要求不低于85%
  3. 手写实现RNN和LSTM的模型，训练后测试集准确率要求不低于80%
  4. 调整网络结构、损失函数、训练流程，观察对训练效果的影响
  5. 总结实验报告

作业

- 作业2:
- 用Transformer在 Multi30k 数据集上训练，实现德语->英语翻译
- 要求:
  1. 阅读示例代码，包括Transformer和Attention的实现代码
  2. 训练20个epoch，测试不同的语句的翻译效果
  3. 调整网络结构、训练流程，观察对训练效果的影响
  4. 总结实验报告

1. Brick layers constructing a wall.		1. The two men on the scaffolding are helping to build a red brick wall.
2. Maurer bauen eine Wand.		2. Zwei Mauerer mauern ein Haus zusammen.
1. Trendy girl talking on her cellphone while gliding slowly down the street		1. There is a young girl on her cell-phone while skating.
2. Ein schickes Mädchen spricht mit dem Handy während sie langsam die Straße entlangschwebt.		2. Eine Frau im blauen Shirt telefoniert beim Rollschuhfahren.
(a) Translations		(b) Independent descriptions

谢谢