Attention is all you need
论文解读

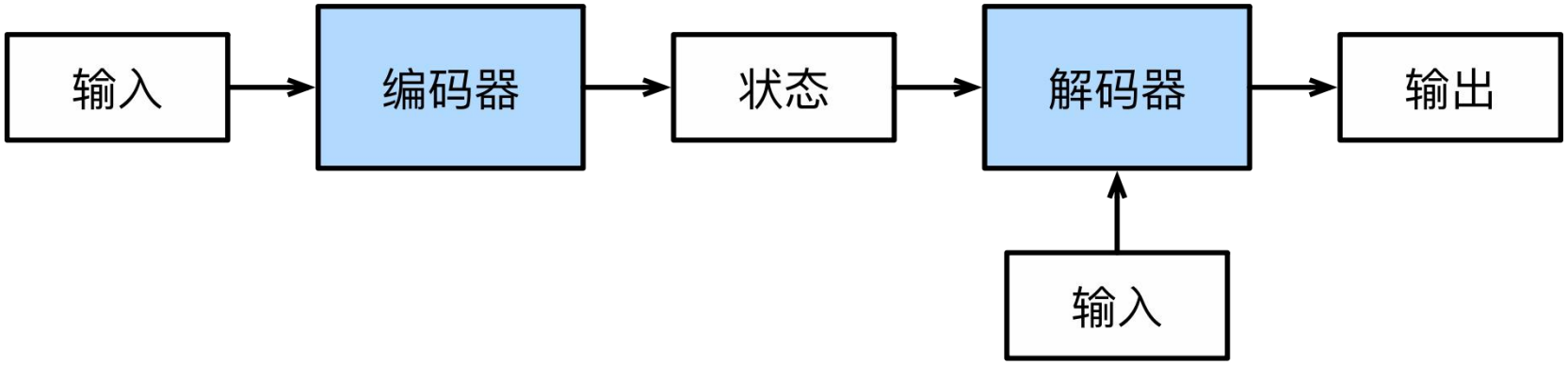
--by Anruihe

目录

CONTENTS

编码器-译码器架构

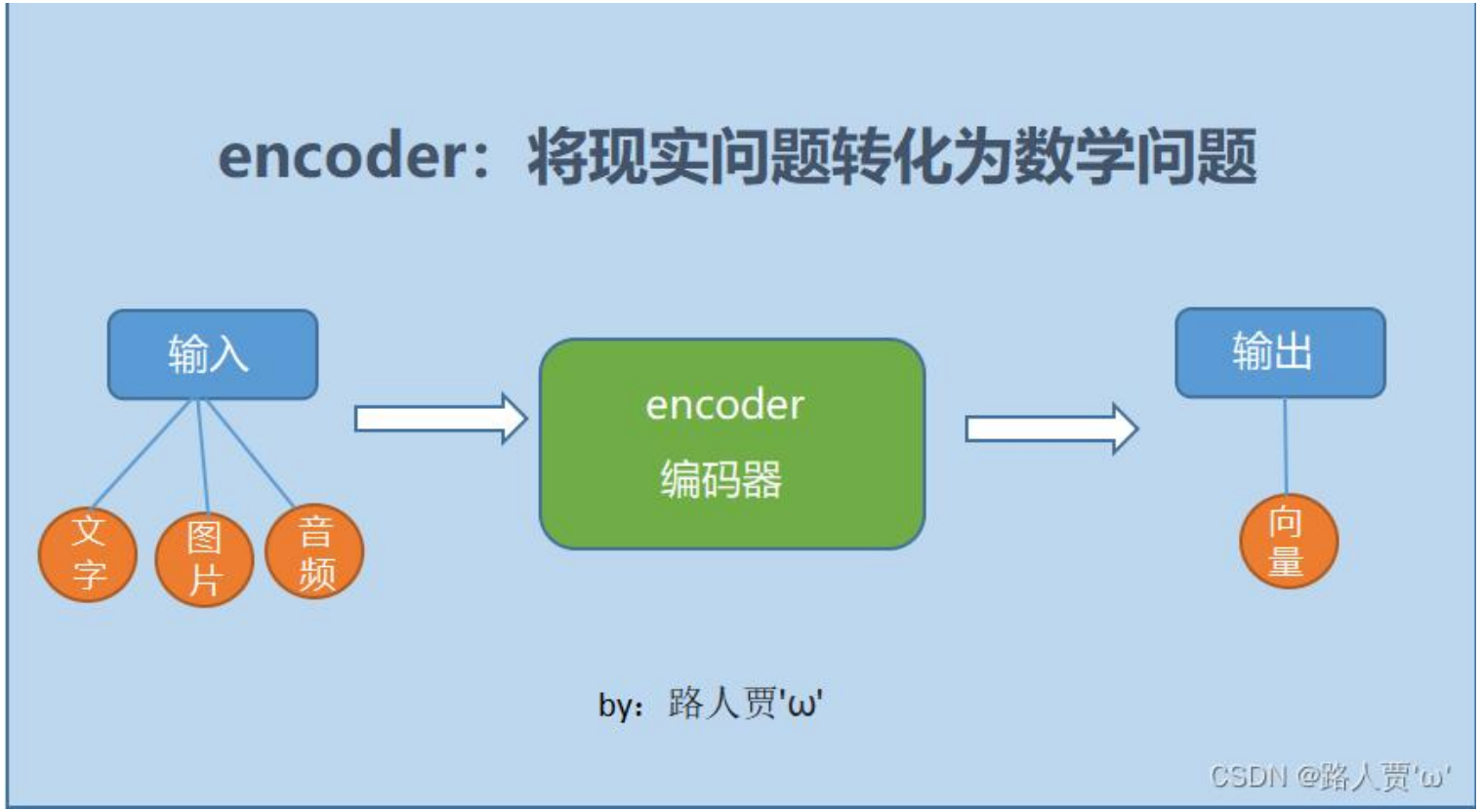
- 机器翻译（machine translation）指的是将序列从一种语言自动翻译成另一种语言。
- 机器翻译是序列转换模型的一个核心问题， 其输入和输出都是长度可变的序列。 为了处理这种类型的输入和输出， 我们可以设计一个包含两个主要组件的架构： 第一个组件是一个编码器（encoder）： 它接受一个长度可变的序列作为输入， 并将其转换为具有固定形状的编码状态。 第二个组件是解码器（decoder）： 它将固定形状的编码状态映射到长度可变的序列。 这被称为编码器-解码器（encoder-decoder）架构， 如图所示。



我们以英语到法语的机器翻译为例： 给定一个英文的输入序列： “They” “are” “watching” “.” 。 首先，这种“编码器 - 解码器”架构将长度可变的输入序列编码成一个“状态”， 然后对该状态进行解码， 一个词元接着一个词元地生成翻译后的序列作为输出： “Ils” “regordent” “.” 。

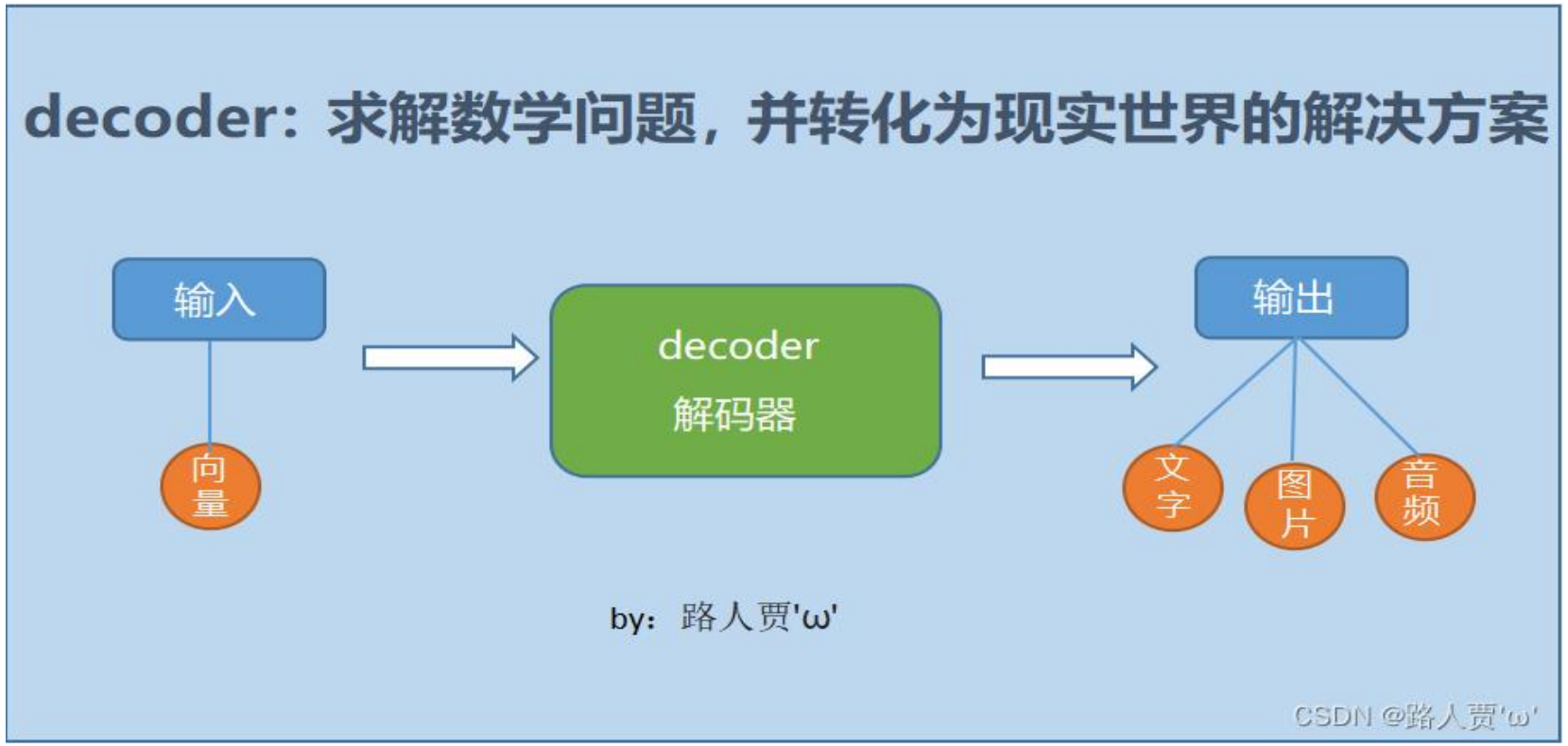
encoder & decoder

- encoder，也就是编码器，负责将输入序列压缩成指定长度的向量，这个向量就可以看成是这个序列的语义，然后进行编码，或进行特征提取（可以看做更复杂的编码）。



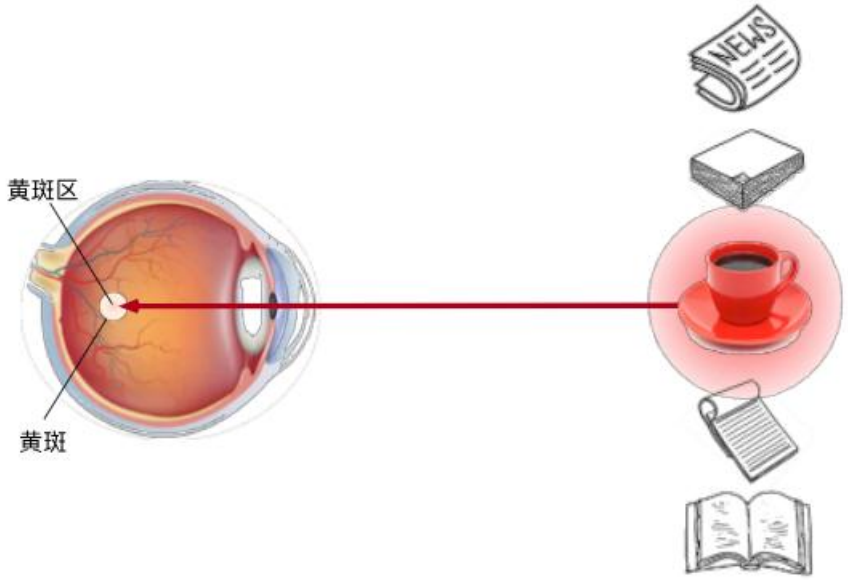
encoder & decoder

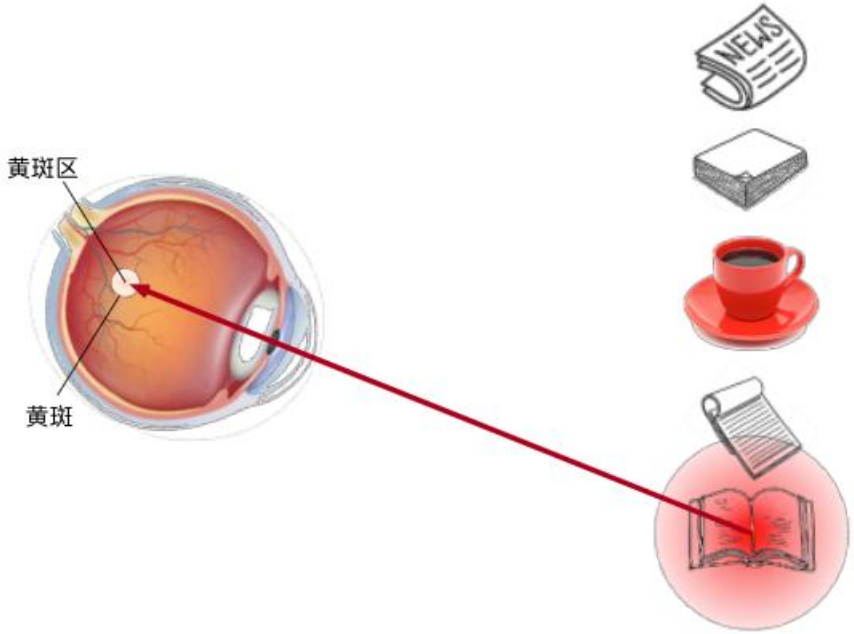
- decoder，也就是解码器，负责根据encoder部分输出的语义向量c来做解码工作。以翻译为例，就是生成相应的译文。
- 简单来说，就是就数学问题，并转换为现实世界的解决方案。



注意力机制

- 注意力是如何应用于视觉世界中的呢？ 这要从当今十分普及的双组件（two-component）的框架开始讲起： 这个框架的出现可以追溯到19世纪90年代的威廉·詹姆斯， 他被认为是“美国心理学之父” (James, 2007)。 在这个框架中，受试者基于非自主性提示和自主性提示有选择地引导注意力的焦点。

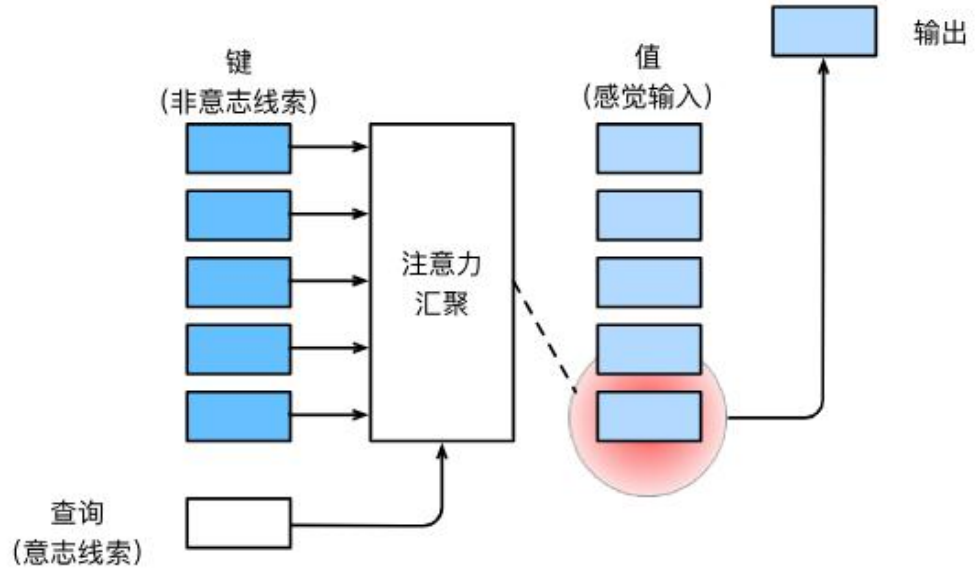




查询、键和值

- 查询（Query）： 指的是查询的范围，自主提示，即主观意识的特征向量
- 键（Key）： 指的是被比对的项，非自主提示，即物体的突出特征信息向量
- 值（Value） ： 则是代表物体本身的特征向量，通常和Key成对出现

- 注意力机制是通过Query与Key的注意力汇聚（给定一个 Query，计算Query与 Key的相关性，然后根据Query与Key的相关性去找到最合适的 Value）实现对Value的注意力权重分配，生成最终的输出结果。



注意力机制通过注意力汇聚将查询（自主性提示）和键（非自主性提示）结合在一起，实现对值（感官输入）的选择倾向

- 从算法上来理解，我们可以把注意力机制和池化做类比，即将卷积神经网络中的池化看成一种特殊的平均加权的注意力机制，或者说注意力机制是一种具有对输入分配偏好的通用池化方法(含参数的池化方法)。

注意力模型

- 在注意力机制的背景下，自主性提示被称为查询（query）。 给定任何查询，注意力机制通过注意力汇聚（attention pooling） 将选择引导至感官输入（sensory inputs，例如中间特征表示）。在注意力机制中，这些感官输入被称为值（value）。 更通俗的解释，每个值都与一个键（key）配对， 这可以想象为感官输入的非自主提示。
- **注意力汇聚**：查询（自主提示）和键（非自主提示）之间的交互形成了注意力汇聚； 注意力汇聚有选择地聚合了值（感官输入）以生成最终的输出。

exp.:
当你用上淘宝购物时，你会敲入一句关键词（比如：羽绒服），这个就是Query。
搜索系统会根据关键词这个去查找一系列相关的Key（商品名称、图片）。
最后系统会将相应的 Value （具体的衣服）返回给你。
在这个例子中，Query， Key 和 Value 的每个属性虽然在不同的空间，其实他们是有一定的潜在关系的，也就是说通过某种变换，可以使得三者的属性在一个相近的空间中。

注意力评分函数

- 输入Query、Key、Value：
- **阶段一**：根据Query和Key计算两者之间的相关性或相似性（常见方法点积、余弦相似度，MLP网络），得到注意力得分；

点积： $\mathbf{Similarity}(\mathbf{Query}, \boldsymbol{Key}_i) = \mathbf{Query} \cdot \boldsymbol{Key}_i$

Cosine 相似性： $\mathbf{Similarity}(\mathbf{Query}, \boldsymbol{Key}_i) = \frac{\mathbf{Query} \cdot \boldsymbol{Key}_i}{||\boldsymbol{Query}|| \cdot ||\boldsymbol{Key}_i||}$

MLP 网络： $\mathbf{Similarity}(\mathbf{Query}, \boldsymbol{Key}_i) = \mathbf{MLP}(\mathbf{Query}, \boldsymbol{Key}_i)$

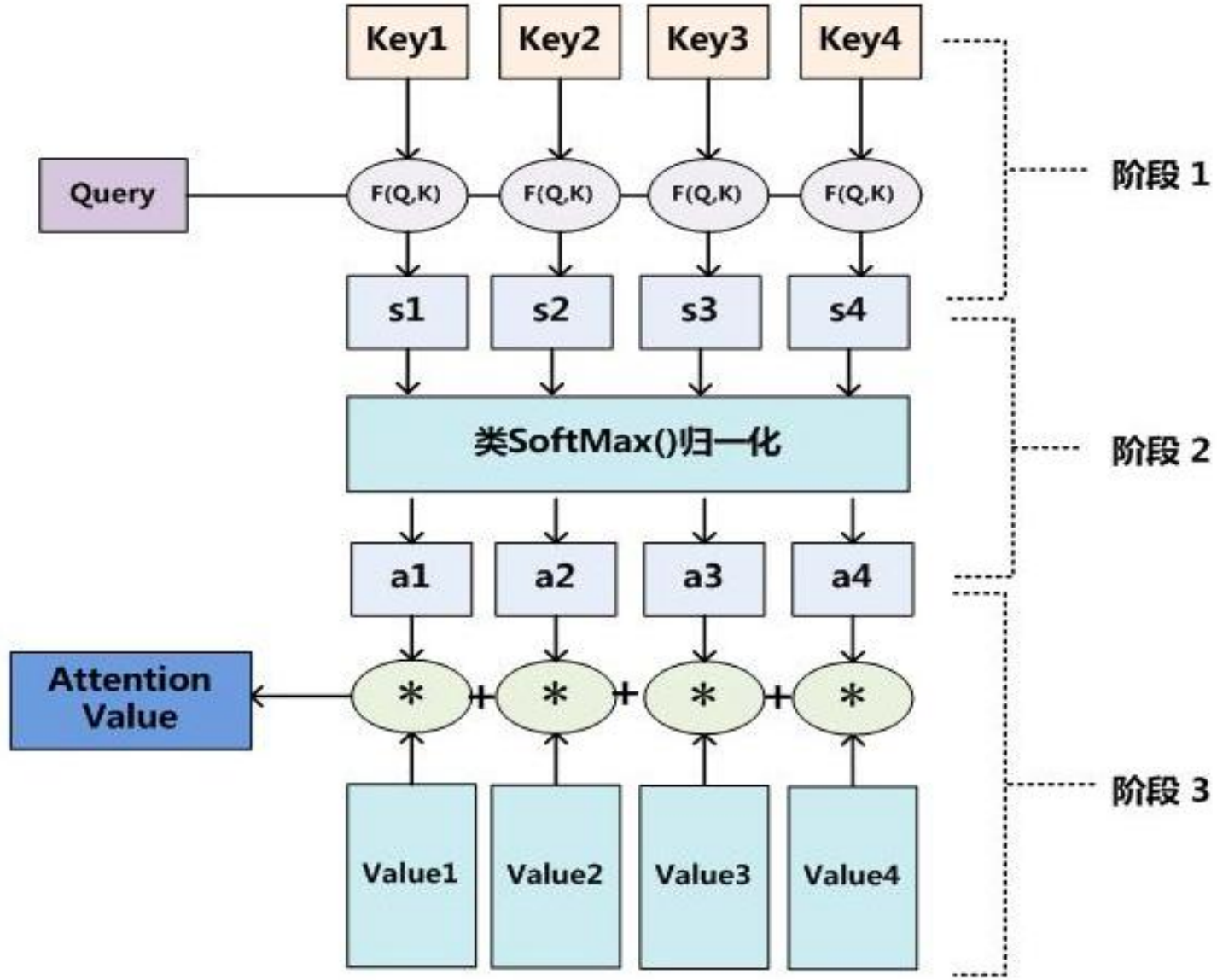
- 阶段二：对注意力得分进行缩放scale（除以维度的根号），再softmax函数，一方面可以进行归一化，将原始计算分值整理成所有元素权重之和为1的概率分布；另一方面也可以通过softmax的内在机制更加突出重要元素的权重。一般采用如下公式计算：

$$a_i = Softmax(Sim_i) = \frac{e^{Sim_i}}{\sum_{j=1}^{L_x} e^{Sim_j}}$$

- （softmax函数：它能将一个含任意实数的K维向量z“压缩”到另一个K维实向量σ(z)中，使得每一个元素的范围都在(0,1)之间，并且所有元素的和为1。该函数多用于多分类问题中。）

- 阶段三：根据权重系数对Value值进行加权求和，得到Attention Value

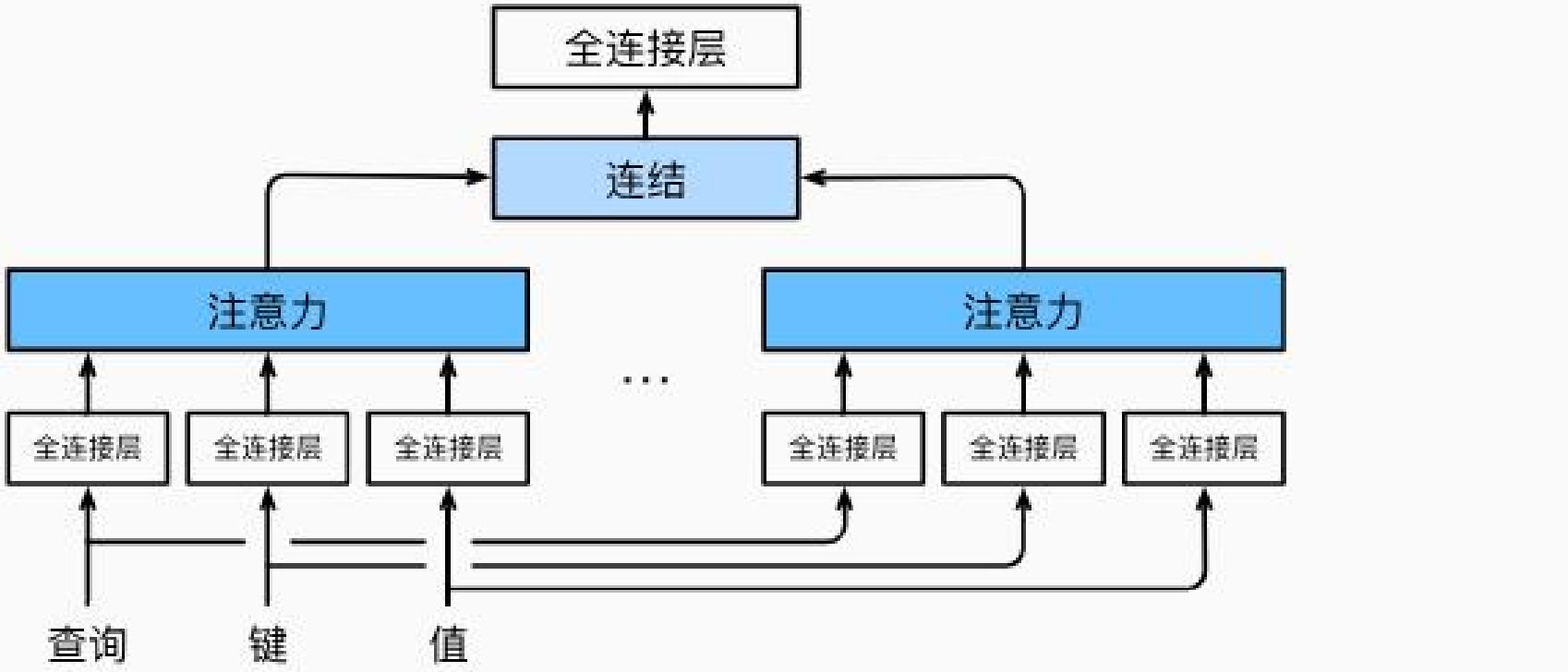
$$\mathbf{Attention}(\mathbf{Query}, \mathbf{Source}) = \sum_{i=1}^{L_x} a_i \cdot Value_i$$


Key1
Key2
Key3
Key4
Query
F(Q,K)
F(Q,K)
F(Q,K)
F(Q,K)
s1
s2
s3
s4
类SoftMax()归一化
a1
a2
a3
a4
Attention Value
*
*
*
*
Value1
Value2
Value3
Value4
阶段 1
阶段 2
阶段 3

多头注意力 MultiHead Attention

- 在实践中，当给定相同的查询、键和值的集合时，我们希望模型可以基于相同的注意力机制学习到不同
（例如，短距离
空间表示（r

- 为此，
（linear proje
意力汇聚中。
影进行变换，
注意力汇聚输
可学习的线



自注意力机制：Self-Attention

- 自注意力机制实际上是注意力机制中的一种，也是一种网络的构型，它想要解决的问题是神经网络接收的输入是很多大小不一的向量，并且不同向量向量之间有一定的关系，但是实际训练的时候无法充分发挥这些输入之间的关系而导致模型训练结果效果极差。比如机器翻译(序列到序列的问题，机器自己决定多少个标签)，词性标注(Pos tagging一个向量对应一个标签)，语义分析(多个向量对应一个标签)等文字处理问题。
- 针对全连接神经网络对于多个相关的输入无法建立起相关性的这个问题，通过自注意力机制来解决，**自注意力机制实际上是想让机器注意到整个输入中不同部分之间的相关性。**
- 自注意力机制是注意力机制的变体，其减少了对外部信息的依赖，**更擅长捕捉数据或特征的内部相关性。**自注意力机制的关键点在于，Q、K、V是同一个东西，或者三者来源于同一个X，三者同源。通过X找到X里面的关键点，从而更关注X的关键信息，忽略X的不重要信息。
- 有了注意力机制之后，我们将词元序列输入注意力池化中， 以便同一组词元同时充当查询、键和值。 具体来说，每个查询都会关注所有的键 - 值对并生成一个注意力输出。**不是输入语句和输出语句之间的注意力机制，而是输入语句内部元素之间或者输出语句内部元素之间发生的注意力机制。** 由于查询、键和值来自同一组输入，因此被称为 自注意力（self-attention）。

循环注意力机制-seq2seq

- Seq2Seq解决问题的主要思路是通过深度神经网络模型（常用的是LSTM，长短记忆网络，一种循环神经网络）。将一个作为输入的序列映射为一个作为输出的序列，这一过程由编码（Encoder）输入与解码（Decoder）输出两个环节组成，前者负责把序列编码成一个固定长度的向量，这个向量作为输入传给后者，输出可变长度的向量。

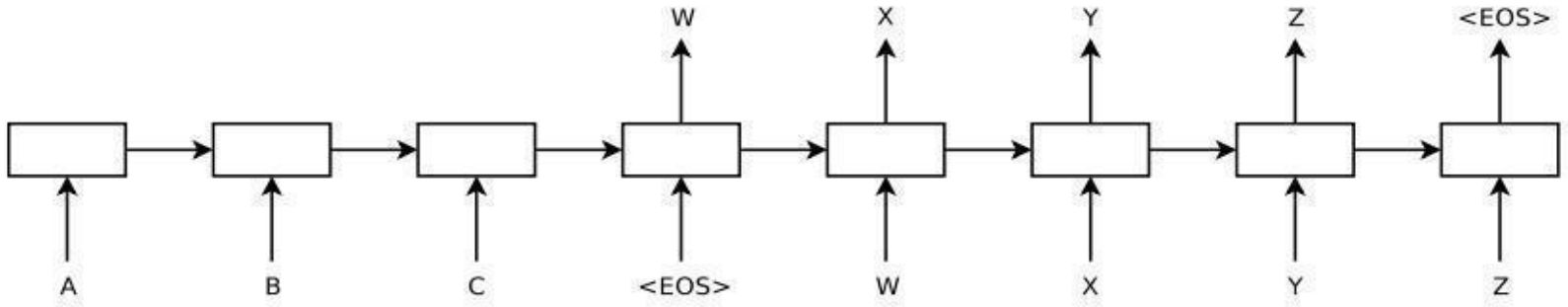
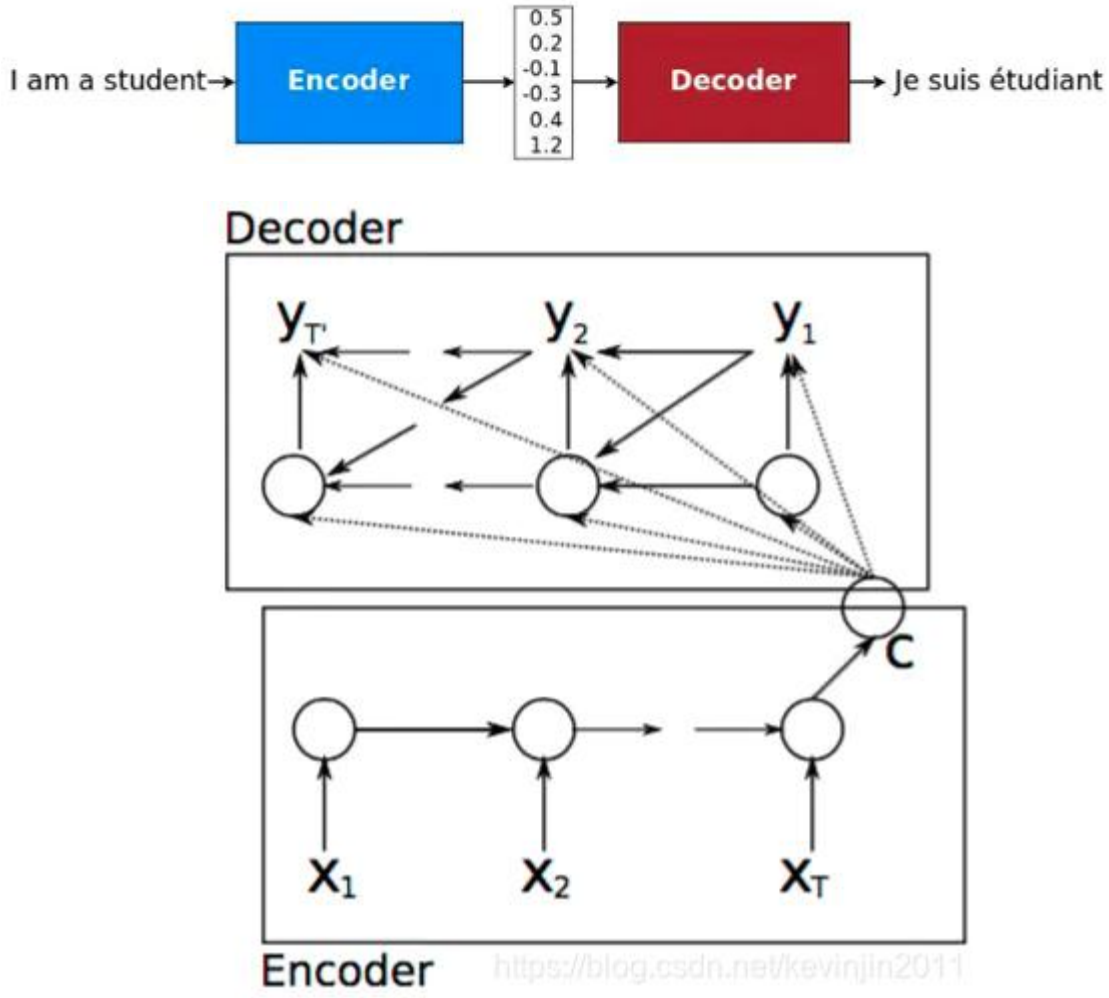


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

- 由上图所示，在这个模型中每一时间的输入和输出是不一样的，比如对于序列数据就是将序列项依次传入，每个序列项再对应不同的输出。比如说我们现在有序列“A B C EOS” （其中EOS = End of Sentence，句末标识符）作为输入，那么我们的目的就是将“A”，“B”，“C”，“EOS”依次传入模型后，把其映射为序列“W X Y Z EOS”作为输出。

seq2seq

- Seq2seq包括三个部分：编码器，解码器，以及连接两者的固定大小的状态向量。Encoder通过学习输入，将其编码成一个固定大小的状态向量，然后将状态向量传给Decoder，Decoder再通过对状态向量的学习来进行输出。
- **Encoder** 是一个RNN，也可以是LSTM、GRU等，接收的是每一个单词的词向量，和上一个时间点的隐藏状态。输出的是这个时间点的隐藏状态。
- **Decoder** 是个RNN，也可以是LSTM、GRU等，将encoder得到的语义向量作为初始状态输入到Decoder的RNN中，得到输出序列。可以看到上一时刻的输出会作为当前时刻的输入，而且其中语义向量只作为初始状态参与运算，后面的运算都与语义向量无关。
- **隐藏状态**是指模型中编码器（Encoder）和解码器（Decoder）之间传递的状态向量，它承载了输入序列的信息并传递给解码器以生成输出序列。



Attention Is All You Need

Abstract—摘要

- 主流的序列转换模型都是基于复杂的循环神经网络或卷积神经网络，且都包含一个encoder和一个decoder。表现最好的模型还通过attention机制把encoder和decoder联接起来。我们提出了一个新的、简单的网络架构，Transformer. 它只基于单独的attention机制，完全避免使用循环和卷积，拥有更强的并行能力，训练效率也得到较高提升。

Introduction

- 之前语言模型和机器翻译的方法和不足
- 方法： RNN、LSTM、GRU、Encoder-Decoder
- 不足：
- ①从左到右一步步计算，因此难以并行计算
- ②过早的历史信息可能被丢弃，时序信息一步一步向后传递
- ③内存开销大，训练时间慢
- 近期工作和问题
- 近期一些工作通过分解技巧和条件计算提高了计算效率，但是顺序计算的本质问题依然存在
- 本文改进
- （1）引入注意力机制： 注意力机制可以在RNN上使用，通过注意力机制把encoder的信息传给decoder，可以允许不考虑输入输出序列的距离建模。
- （2）提出Transformer ： 本文的 Transformer 完全不用RNN，这是一种避免使用循环的模型架构，完全依赖于注意机制来绘制输入和输出之间的全局依赖关系，并行度高，计算时间短。

Background

- *减少序列计算的目标也成就了 Extended Neural GPU [16],ByteNet[18],和ConvS2S[9]的基础,它们都使用了卷积神经网络作为基础模块，并行计算所有输入和输出位置的隐藏表示。在这些模型中，将来自两个任意输入或输出位置的信号关联起来所需的操作数，随位置间的距离而增长，ConvS2S为线性增长，ByteNet为对数增长。这使得学习远距离位置之间的依赖性变得更加困难[12].* 在Transformer中，这种情况被减少到了常数次操作，虽然代价是由于平均注意力加权位置信息降低了有效分辨率，如第3.2节所述，我们用多头注意力抵消这种影响。
- 已经被证明，端到端的记忆网络使用循环attention机制（seq2seq)替代序列对齐的循环，在简单的语言问答和语言建模任务中表现良好。
- 然而，据我们所知，Transformer是第一个完全依赖于self-attetion来计算其输入和输出表示而不使用序列对齐的RNN或卷积的转换模型，在下面的章节中，我们将描述Transformer，motivate ，self-attention，并讨论它相对于[17,18]和[9]等模型的优势。

Model Architecture—模型结构

大多数有竞争力的序列转换模型都有encoder-decoder结构构。这里，encoder将符号表示的输入序列(x 1 , . . . , x n) 映射成一个连续表示的序列z = (z 1 , . . . , z n)。
给定z，解码器以一次生成一个字符的方式生成输出序列(y 1 , . . . , y m) 。在每一步，模型都是自回归的[10]，在生成下一个字符时，将先前生成的符号作为附加输入。

Transformer遵循这个总体架构，使用堆叠的self-attention层、point-wise和全连接层，分别用于encoder和decoder，如图左半部分和右半部分所示。

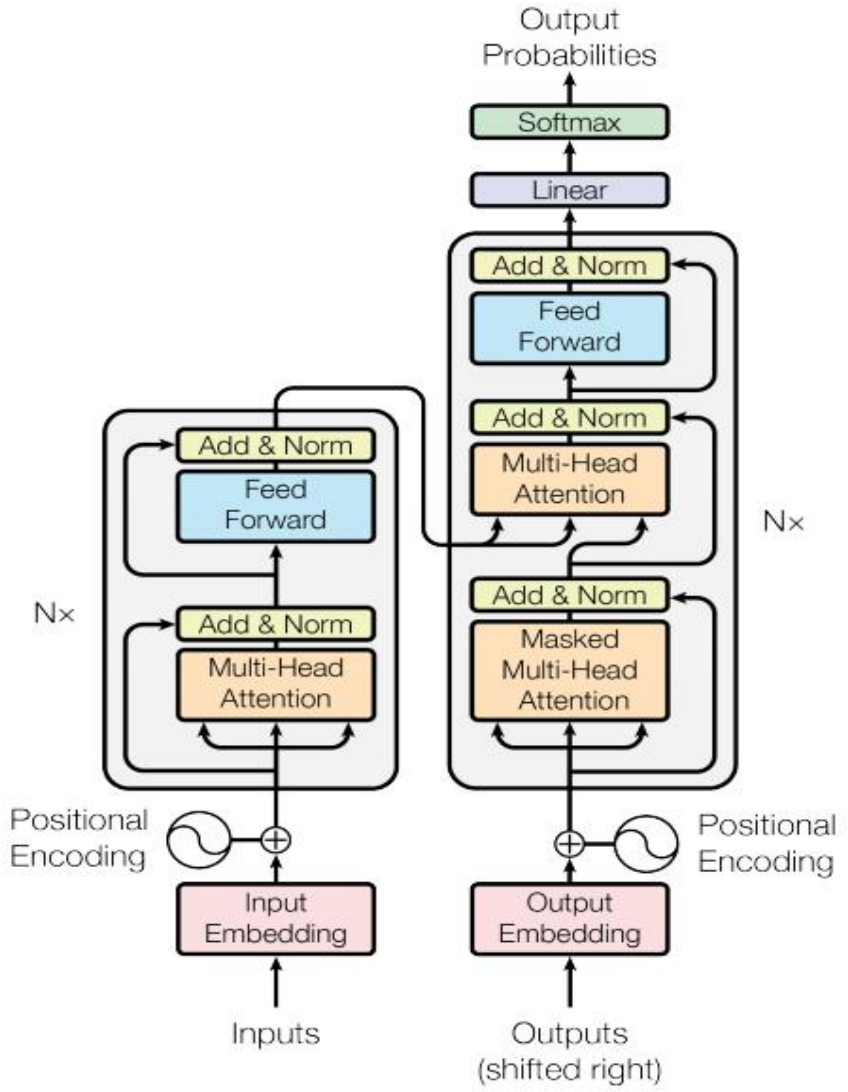
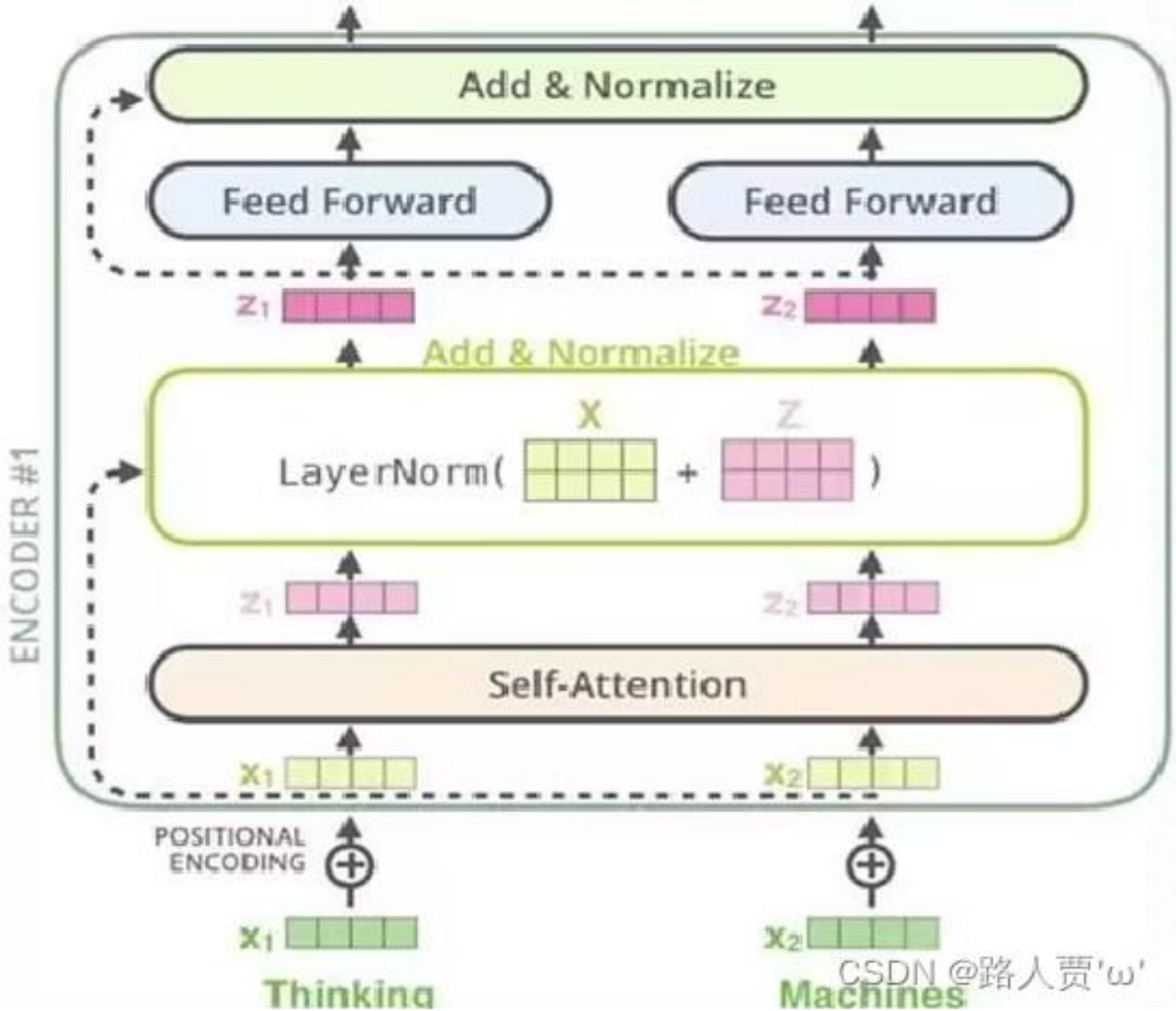


Figure 1: The Transformer - model architecture.

Encoder and Decoder Stacks—编码器栈和解码器栈

encoder

- 将一个长为n的输入
- encoder由n个相同
- 每个sub-layer的输
- 残差连接需要输入
- 残差连接在Transf
- 与CNN不一样的是



（机器可以理解的向

s，每个sub-layers

vise fully

rm(x +

里都是固定的，都是

缓解梯度消失和梯

annel维度上升。

残差连接

- 缓解梯度消失： 在深层网络中，梯度在反向传播中可能会变得非常小，导致底层的权重更新几乎不可见。通过使用残差连接，模型可以直接跳过一层或多层的操作，使梯度能够更轻松地传递到较早的层次，从而缓解梯度消失问题。

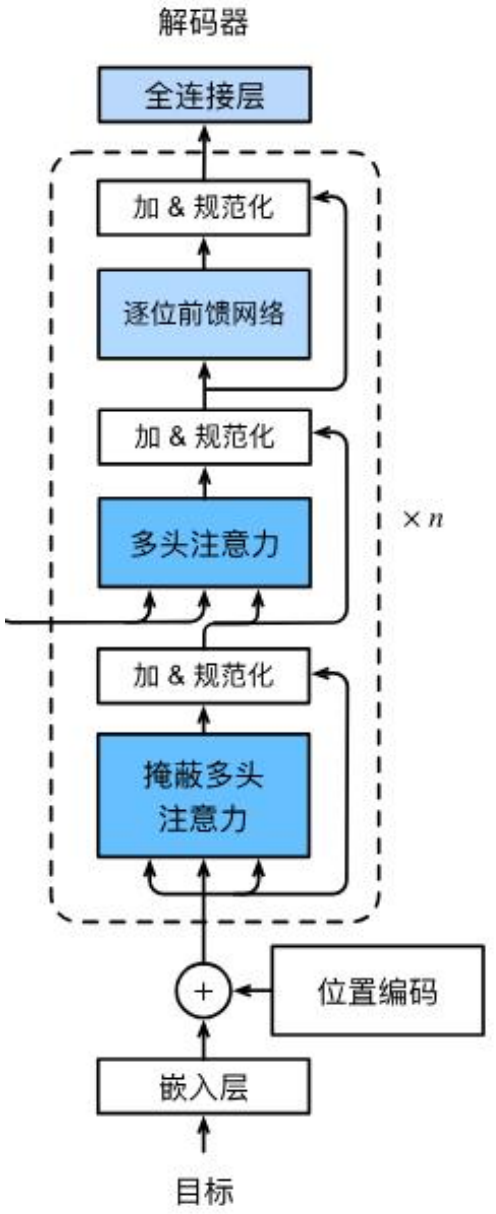
- 加速训练收敛： 残差连接使得模型在训练初期更容易收敛。通过直接学习残差（即残差块的输出与输入的差异），模型可以更快地适应恒等映射（identity mapping），从而使得训练的开始阶段更加稳定。

- 简化学习： 残差连接引入了跨层的直接路径，简化了网络的学习任务。模型只需学习恒等映射的残差，而不必学习复杂的非线性变换。这有助于提高网络的表达能力。

- 防止梯度爆炸： 在某些情况下，梯度可能会变得非常大，导致权重更新过大。残差连接可以有助于抑制梯度爆炸，因为它提供了一个稳定的、直接的路径，减少了梯度的变化。

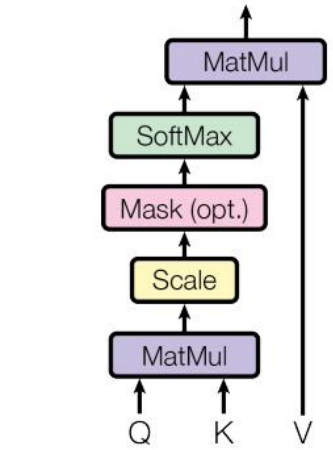
Decoder

- decoder也由N(N=6)个完全相同的layer堆叠而成.除了每个编码器层中的两个子层之外，解码器还插入第三个子层，该子层对编码器堆栈的输出执行multi-head attention操作，与encoder相似，我们在每个子层的后面使用了残差连接，之后采用了layer normalization。我们也修改了decoder stack中的 self-attention 子层，以防止当前位置信息中被添加进后续的位置信息。这种掩码与偏移一个位置的输出embedding相结合， 确保对第i ii个位置的预测 只能依赖小于i的已知输出。
- 掩码注意力层就是将t时刻后的数据权重设置为 0，该层还是自注意力的。
- 在掩蔽多头解码器自注意力层（第一个子层）中，查询、键和值都来自上一个解码器层的输出。关于序列到序列模型（sequence-to-sequence model），在训练阶段，其输出序列的所有位置（时间步）的词元都是已知的；然而，在预测阶段，其输出序列的词元是逐个生成的。因此，在任何解码器时间步中，只有生成的词元才能用于解码器的自注意力计算中。为了在解码器中保留自回归的属性，其掩蔽自注意力设定了参数dec_valid_lens，以便任何查询都只会与解码器中所有已经生成词元的位置（即直到该查询位置为止）进行注意力计算。



Scaled Dot-Product Attention

- Scaled Dot-Product Attention是特殊attention，输入包括查询Q和键K的维度dk 以及值V的维度dv 。计算查询和键的点积，将每个结果除√dk ，然后用 softmax() 函数来获得值的权重。
- 通过缩放查询和键的点积来解决梯度消失问题。
- 在实际使用中，我们同时计算一组查询的注意力函数，并一起打包成矩阵 Q。键和值也一起打包成矩阵 K 和 V。我们按照如下方式计算输出矩阵：



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

查询向量和键向量之间的注意力分数通过取查询和键向量的点积计算。
为了缓解点积中大值的影响，将结果缩小，缩放因子是键向量维度的平方根。
这种缩放有助于防止训练过程中梯度变得太小。

MLP：simple，position-wise fully connected feed-forward network

- 除了attention子层，我们encoder-decoder框架中每一层都包含一个全连接的前馈网络，它分别相同地应用于每个位置。它由两个线性变换和中间的一个ReLU激活函数组成公式

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

- 1.Position-wise（位置相关）： 这表示层内的操作独立地应用于输入序列的每个位置或元素。在Transformer模型的背景下，序列是并行处理的，操作不依赖于元素的相对位置。
- 2.Fully connected（全连接）： 这表示输入序列中的每个元素与该层中的每个其他元素相连接。在全连接层中，每个神经元与前一层和后一层中的每个神经元相连接。
- 3.Feed-forward network（前馈网络）： 该层执行前馈操作，这涉及在没有循环连接或反馈循环的情况下从输入传播信息到输出。该层中的计算基于输入特征的加权和，然后是激活函数。

- 在Transformer架构中，位置相关的全连接前馈网络通常应用于每个编码器和解码器层的自注意力机制之后。它通过一系列全连接层独立地处理每个位置的输入，有助于捕获输入序列中的复杂非线性关系。

Embeddings and Softmax —词嵌入和 softmax

- Embedding： 特征嵌入，embedding是可以简单理解为通过某种方式将词向量化，即输入一个词输出该词对应的一个向量。（embedding可以采用训练好的模型如GLOVE等进行处理，也可以直接利用深度学习模型直接学习一个embedding层，Transformer模型的embedding方式是第二种，即自己去学习的一个embedding层。）

- embeddings将输入和输出tokens转换为向量，线性变换和softmax函数将decoder输出转换为预测的写一个token概率。

Positional Encoding——位置编码

- 因为transformer模型不包含循环或卷积，输出是V的加权和（权重是 Q与K的相似度，与序列信息无关），对于任意的K-V，将其打乱后，经过注意力机制的结果都一样
- 但是它顺序变化而值不变，在处理时序数据的时候，一个序列如果完全被打乱，那么语义肯定发生改变，而注意力机制却不会处理这种情况。

- 方法：
- 在注意力机制的输入中加入时序信息，位置在encoder端和decoder端的embedding之后，用于补充Attention机制本身不能捕捉位置信息的缺陷。
- 一次词在嵌入层表示成一个512维的向量，用另一个512维的向量表示位置数字信息的值。用周期不一样的sin和cos函数计算。

why self-attention

- 一是每层的总计算复杂度低。
- 另一个是可以并行化的计算量，以所需的最小序列操作数衡量。
- 第三个是网络中长距离依赖关系之间的路径长度。

在许多序列转换任务中，学习长距离依赖性是一个关键的挑战。影响学习这种依赖关系能力的一个关键因素是网络中向前和向后信号必须经过的路径的长度。输入和输出序列中任意位置组合之间的这些路径越短，越容易学习长距离依赖。因此，我们还比较了在由different layer types组成的网络 中的任意两个输入和输出位置之间的最大的路径长度。

和CNN、RNN比较

- n表示序列长度，d是隐藏层维度，k表示卷积核尺寸，r表示受限自注意力的窗口大小

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

self-attention layer用常数次(O（1）)的操作连接所有位置，而recurrent layer需要O(n)顺序操作。
在计算复杂度方面，当序列长度N小于表示维度D时，self-attention layers比recurrent layers更快，这是使用最先进的机器翻译模型表示句子时的常见情况，例如word-piece [38] 和byte-pair [31] 表示。
为了提高包含很长序列的任务的计算性能，可以仅在以输出位置为中心，半径为r的的领域内使用self-attention。这将使最大路径长度增长到O（n / r）。

Results—结果

- WMT 2014英语-德语翻译任务表现
- （1）评分更高： 取得了28.4的BLEU评分。在现有的表现最好模型的基础上，包括整合模型，提高了2个BLEU评分。
- （2）成本更小： 训练成本只是这些模型的一小部分
- WMT 2014英语-法语翻译任务表现
- （1）评分更高： 大型模型的BLEU得分为41.0，超过了之前发布的所有单一模型
- （2）成本更小： 训练成本低于先前最先进模型的1／4

Model Variations—模型变体

- 在表3的行（A）中，我们改变attention head的数量和attention key和value的维度，保持计算量不变，如3.2.2节所述。 虽然只有一个head attention比最佳设置差0.9 BLEU，但质量也随着head太多而下降。
- 在表3行（B）中，我们观察到减小key的大小dk会有损模型质量。 这表明确定兼容性并不容易，并且比点积更复杂的兼容性函数可能更有用。
- 我们在行（C）和（D）中进一步观察到，如预期的那样，更大的模型更好，并且dropout对避免过度拟合非常有帮助。
- 在行（E）中，我们用学习到的positional encoding[9]来替换我们的正弦位置编码，并观察到与基本模型几乎相同的结果。

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

English Constituency Parsing—英文选区分析

- 为了评估Transformer是否可以扩展到其他任务，我们进行了英语选区解析的实验。这项任务提出特别的挑战：输出受到很强的结构性约束，并且比输入要长很多。 此外，RNN序列到序列模型还没有能够在小数据[37]中获得最好的结果。
- 我们用dmodel = 1024 在Penn Treebank[25]的Wall Street Journal（WSJ）部分训练了一个4层的transformer，约40K个训练句子。 我们还使用更大的高置信度和BerkleyParser语料库，在半监督环境中对其进行了训练，大约17M个句子[37]。 我们使用了一个16K词符的词汇表作为WSJ唯一设置，和一个32K词符的词汇表用于半监督设置。
- 我们只在开发集的Section 22 上进行了少量的实验来选择dropout、attention 和residual（第5.4节）、learning rates和beam size，所有其他参数从英语到德语的基础翻译模型保持不变。在推断过程中，我们将最大输出长度增加到输入长度+300。 对于WSJ和半监督设置，我们都使用beam size = 21 和α = 0.3 。
- 表4中我们的结果表明，尽管缺少特定任务的调优，我们的模型表现得非常好，得到的结果比之前报告的Recurrent Neural Network Grammar [8]之外的所有模型都好。
- 与RNN序列到序列模型[37]相比，即使仅在WSJ训练40K句子组训练时，Transformer也胜过BerkeleyParser [29]。

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

Parser

- 词法分析器（Lexical Parser）： 词法分析器主要关注文本中的词汇单元，即词法单位，如单词。它负责分割文本为一个个的词汇单元，通常包括分词和词干提取等任务。
- 句法分析器（Syntactic Parser）： 句法分析器关注语法结构，即句子中词汇单元之间的关系。它可以分析句子的结构，确定句子中不同短语的组成方式，生成树形结构，通常被称为语法树。常见的句法分析方法包括上下文无关文法（CFG）和依存句法分析。
- 语义分析器（Semantic Parser）： 语义分析器关注理解文本的意义和语境。它试图将自然语言文本映射到语义表示，以便计算机能够更好地理解文本的含义。语义分析器通常用于提取实体、关系、事件等语义信息。
- 情感分析器（Sentiment Parser）： 情感分析器专注于识别文本中的情感色彩，即文本所包含的情绪或情感倾向。这对于理解用户对产品、服务或事件的情感反馈非常有用。
- 依存句法分析器（Dependency Parser）： 依存句法分析器专注于识别句子中词汇之间的依存关系，即一个词与另一个词之间的语法关系。通过分析这些依存关系，可以构建依存树。

- RNN（循环神经网络）：
- 介绍： RNN是一种具有循环连接的神经网络，能够处理序列数据。在每个时间步，RNN都接收输入和前一个时间步的隐藏状态，并生成输出和新的隐藏状态。
- 问题： RNN在长序列上容易出现梯度消失或梯度爆炸的问题，限制了其对长时依赖关系的建模能力。
- LSTM（长短时记忆网络）：
- 介绍： LSTM是为了解决RNN的长时依赖问题而设计的。它引入了三个门（输入门、遗忘门、输出门）来控制信息的流动，有效地处理长期记忆。
- 特点： LSTM通过门控机制，能够更好地捕捉和保持序列中的长期依赖关系，适用于需要长时间记忆的任务。
- GRU（门控循环单元）：
- 介绍： GRU是对LSTM的一种简化，仅使用两个门（更新门和重置门）。它减少了参数数量，计算效率更高。
- 特点： GRU在某些任务上表现得与LSTM相当，但由于参数较少，通常更容易训练。