目录

[1 Attention 1](#_Toc13679158)

[1.1 Encoder-Decoder结构 1](#_Toc13679159)

[1.2 Soft Attention 1](#_Toc13679160)

[1.2.1 E-D结构的局限性 1](#_Toc13679161)

[1.2.2 Attention机制的应用与计算方式 1](#_Toc13679162)

[1.2.3 Attention机制的本质思想 2](#_Toc13679163)

[2 Self Attention 4](#_Toc13679164)

[2.1 Self Attention思想 4](#_Toc13679165)

[2.1.1 Self与Soft的差异 4](#_Toc13679166)

[2.1.2 与RNN相比的优势 4](#_Toc13679167)

[2.2 Self Attention的一种实现方式 4](#_Toc13679168)

[3 Transformer模型结构 6](#_Toc13679169)

[3.1 Scaled Dot-Product Attention 6](#_Toc13679170)

[3.2 Multi-HeadAttention 6](#_Toc13679171)

[3.3 Transformer结构 7](#_Toc13679172)

# Attention

## Encoder-Decoder结构



图 1.1E-D结构

一般用于句子对<Source,Target>，我们的目标是给定输入句子Source，期待通过Encoder-Decoder框架来生成目标句子Target。Source和Target可以是同一种语言，也可以是两种不同的语言。而Source和Target分别由各自的单词序列构成：

计算公式：

编码器输出的中间编码

解码器输出的每一维

## Soft Attention

### E-D结构的局限性

由的计算方式可以看出，句子Source中任意单词对生成某个目标单词来说影响力都是相同的，都是根据相同的C计算出来的。而实际上，不同位置的对的影响都是不同的。

比如“Tom chase Jerry”—>“汤姆 追逐 杰瑞”，对的影响明显要比和大。

### Attention机制的应用与计算方式

计算每个的时候，都考虑到Source中每个单词的权重(注意力分配概率)信息，这时每个计算时的C就不再相同了，如图1.2所示。

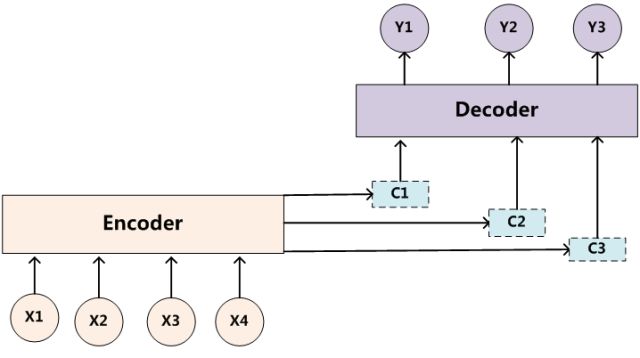


图 1.2加入Attention之后的E-D结构

计算公式：

代表相对与的权重，或者叫注意力分配系数

函数h是对输入的某种转换函数，比如Encoder用的RNN模型,h函数的结果往往是某个时刻输入后隐层节点的状态值。

在解码器中其实就是根据C和计算出来的状态值。

这里，所有相对于的权重向量的计算方式如下：

### Attention机制的本质思想

抛开Encoder-Decoder模型，Attention机制的本质思想如图1.3：

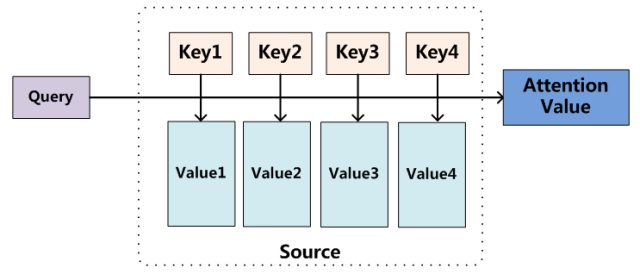


图 1.3Attention机制

其中，Source可以看成由一系列<Key,Value>数据对组成， Query为Target中要预测的元素，Attention的目的就在于通过Query和Key计算出相应Value对于Query而言的权重系数W，最后根据W对Vaule值加权求和就得到了Attention Value。

在很多E-D结构中，Key和Value是相同的，都使用的是词的编码表示。

# Self Attention

## Self Attention思想

### Self与Soft的差异

Soft Attention是为了计算Target中与Source中所有之间的关系

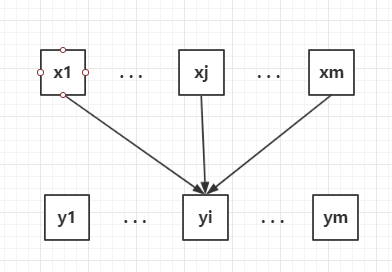


图 2.1 Soft Attention思想

Self Attention则是为了计算Source中所有之间的相互关系

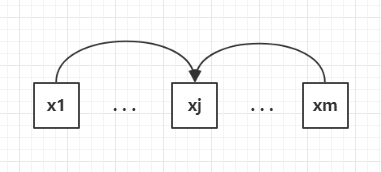


图 2.2Self Attention思想

### 与RNN相比的优势

RNN模型在计算时是不断的循环累积，距离越远的单词保留的信息就越少，在处理中长句时很难捕捉远距离单词之间的联系。

Self Attention计算的则是句中的每个单词与其他所有单词之间的关系，因此摆脱了远距离依赖问题。

## Self Attention的一种实现方式

《Attention is all you need》中提出了Self Attention的一种实现：Scaled Dot-Product Attention，结构如图2.3所示。



图 2.3Self Attention的一种实现

传统的Soft Attention中，K=V为Source中词的编码表示，Q为Target中词的编码表示。但在Self Attention中，Q=K=V都是Source中词的编码表示，学习的是Source内部词与词之间的联系。

## 结合翻译任务分析Self Attention

在翻译任务中，E-D结构计算出的中间语义编码C可以理解成语言在更高层次上的表达，对具体不同的语言都具有普适性。而Self Attention所学习的也正是句子内部词与词之间的联系，可以理解为句子的语法语义等高级信息，和中间语义编码C的性质就比较相似。



# Transformer模型结构

## Scaled Dot-Product Attention

一种特殊的Attention结构，输入三个矩阵：Q(query)、K(key)、V(value)。



图 3.1 Scaled Dot-Product Attention

**数学公式：**

计算公式：

最终得到一个的矩阵。

其中：Scale层是正规化，Mask层实现一步的除法操作

## Multi-HeadAttention

并行多个Scaled Dot-Product Attention模块，然后将结果融合，也叫做自注意力(self-Attention)模型。

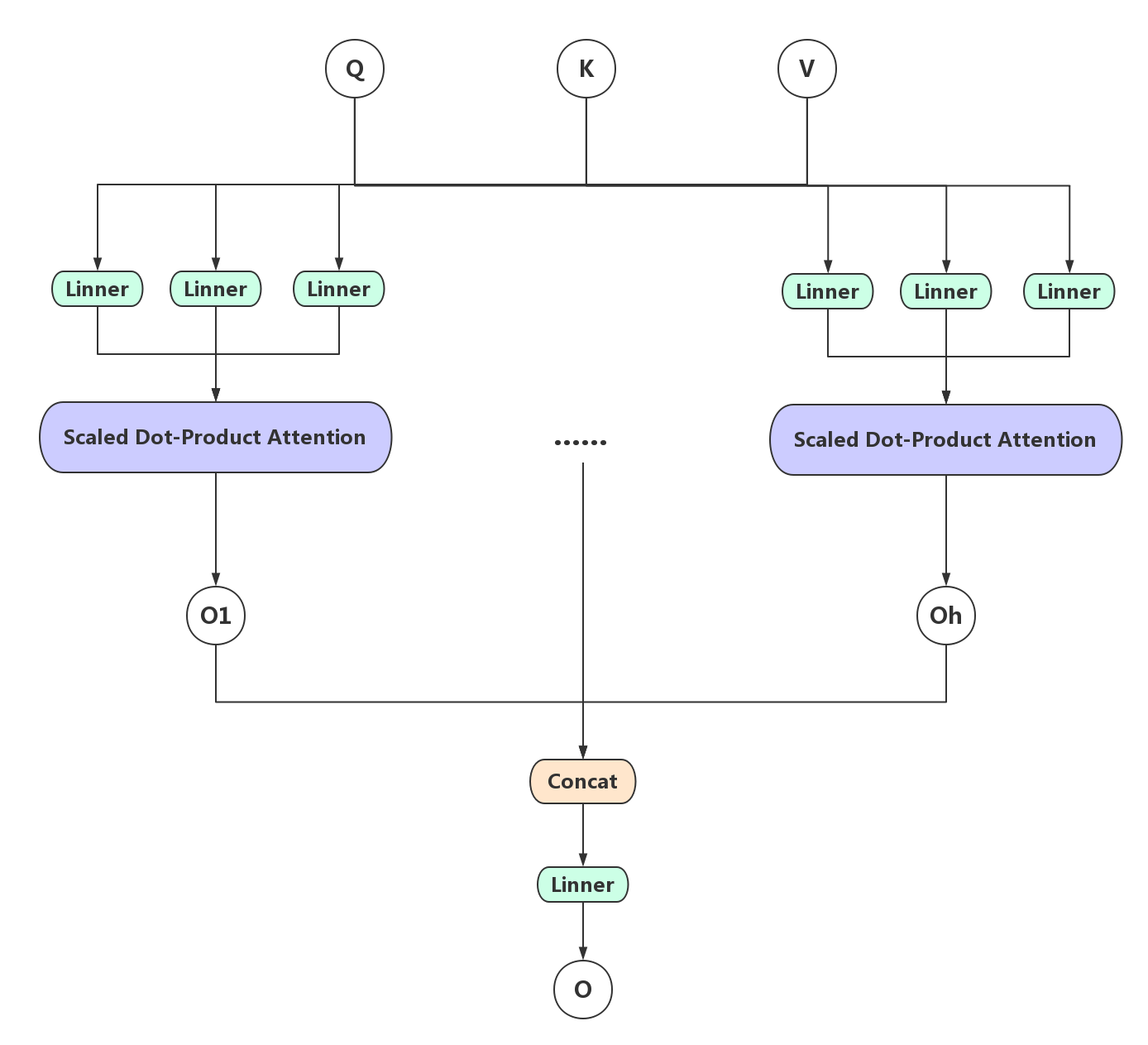


图 3.2Multi-HeadAttention

**各层说明：**

Linner: 线性映射层，即Dense层

Concat: 矩阵融合操作，将若干个矩阵和在一起，Concat(n\*m, n\*m)=n\*2m，最后再使用一个Dense层将维度还原为n\*m

## Transformer结构

网络模型结构如图3.3所示。



图 3.3Transformer结构

Add: 矩阵加法 Norm:正规化

Positional Encoding：位置编码，引入上下文信息，可以参见ELMO和BERT

其中，左侧的Multi-HeadAttention+Feed Forward构成一个Encoder模块，右侧的两个Multi-HeadAttention+Feed Forward构成一个Decoder模型。在论文中，N取值为6，因此Transformer结构的结构也可以表示成图3.4所示。

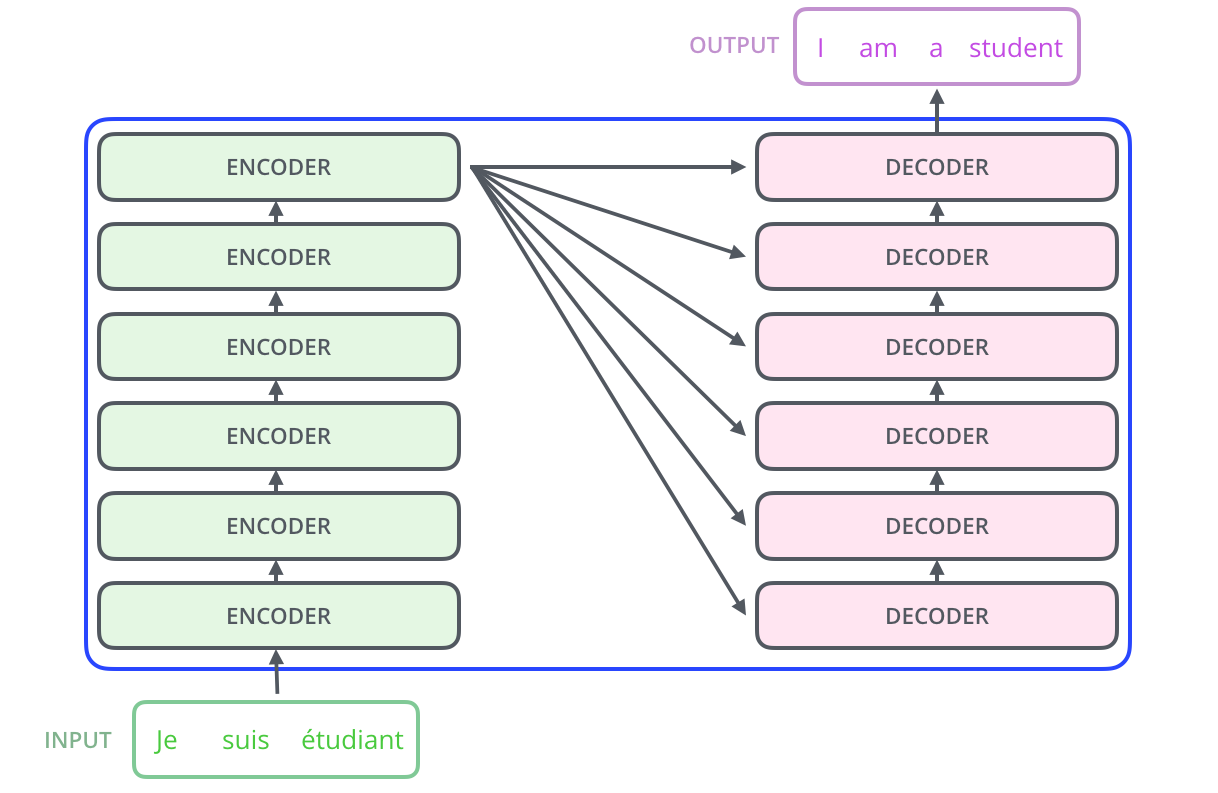


图 3.4Transformer的另一种表示

由上图可以看出，Encoder和Decoder都是一层层串联的，最后一层Encoder的结果会传给所有Decoder作为部分输入。

## 训练与测试