拆 Transformer 系列一：Encoder-Decoder 模型架构详解

随时学丫
凤凰网 推荐算法工程师
关注她

人工智能的发展非常迅速，翻译的准确性越来越高，以至于有人担心译员是否会失业，甚至有的家长担心孩子学习外语专业将来找不到工作。哎呀，扯远了，本人认为：机器翻译可以便于人们更好的获得国外的信息，提高专业译员的翻译速度，但是更深层次的思考，仍然还要依赖人工翻译。

机器翻译目前只在通用领域 (商业领域，口语领域等) 的短句上效果不错，复杂结构的长句，需要结合知识尝试和上下文判断歧义句，特定领域的翻译上，一些小众语种以及一些特别复杂的语言结构，效果还远不能让人满意。

研究算法，除了调参就是洗数据，数据不好再收集更多数据，接着洗数据，调参，调参。在我们调参结束的时候，我们有没有思考过，我们对调参的这个模型的原理真的理解吗?

我曾经在跑一个 Bert 模型，我只是粗略的看了大致的原理，总以为自己很懂了，由于 Bert 模型效果确实挺不错，以至于我用一个模型打天下了很久，直到某天我遇到了一个问题，我着实不懂如何解决，后来，翻看源码，改了其中某一个参数值，总算解决了。

所以，我们工作之余，对一些模型，架构的思想要理解，更要明白原理，今天，我们就先来学习 Encoder-Decoder 框架，本着死磕模型原理的原则，面对心中一堆的疑问，我觉得有必要好好谈谈 Encoder-Decoder。

追溯 Encoder-Decoder 的由来

Encoder-Decoder 通常称作 编码器-解码器，是深度学习中常见的模型框架，很多常见的应用都是利用编码-解码框架设计的，如：

- 无监督算法的 auto-encoding 就是利用编码-解码结构设计的。
- image caption 的应用也是利用 CNN-RNN 的编码-解码框架。
- 神经网络机器翻译 NMT 模型，就是 LSTM-LSTM 的编码-解码框架。

综合上述的应用，我们可以知道 Encoder-Decoder 并不是一个具体的模型，而是一个通用的框架。Encoder 和 Decoder 部分可以是任意文字，语音，图像，视频数据，模型可以是 CNN，RNN，LSTM，GRU，Attention 等等。所以，基于 Encoder-Decoder，我们可以设计出各种各样的模型。

上面提到的编码，就是将输入序列转化转化成一个固定长度向量。解码，就是讲之前生成的固定向量再转化出输出序列。

由此，Encoder-Decoder 有 2 点需要注意：

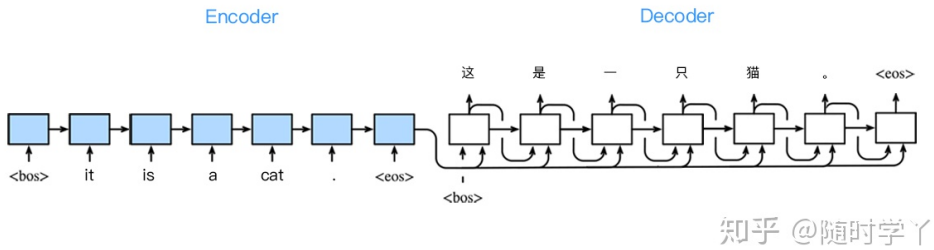
- 不管输入序列和输出序列长度是什么，中间的「向量 c」长度都是固定的，这也是它的一个缺陷。
- 不同的任务可以选择不同的编码器和解码器 (RNN，CNN，LSTM，GRU)。

Encoder-Decoder 有一个比较显著的特征就是它是一个 End-to-End 的学习算法，以机器翻译为力，可以将法语翻译成英语。这样的模型也可以叫做 Seq2Seq。

Seq2Seq (Sequence-to-sequence 的缩写)，就如字面意思，输入一个序列，输出另一个序列。这种结构最重要的地方在于输入序列和输出序列的长度是可变的。

Seq2Seq 强调目的，不特指具体方法，满足输入序列，输出序列的目的，都可以统称为 Seq2Seq 模型。Seq2Seq 使用的具体方法基本都是属于 Encoder-Decoder 模型的范畴。

在机器翻译里面，如下图，将英语 「it is a cat.」翻译成汉语 「这是一只猫。」，输入 4 个单词，输出 5 个汉字。



在训练数据集中，我们可以在每个句子后附特殊字符 ”“ (end of sequence) 以表示序列终止，每个句子前用到了特殊字符 "" (begin of seqence) 表示序列开始。Encoder 在最终时间步的隐状态作为输入句子表征和编码信息。Decoder 在各个时间步中使用输入句子的编码信息和上一个时间步的输出以及隐藏状态作为输入。

> **案例：英文 it is a cat. 翻译成中文的过程。**
> 1. 对序列进行建模，得到概率最大的译词，如第一个词为 “这”。将生成的词加入译文序列，重复上述步骤，不断迭代。
> 2. 直到终止符号被模型选择出来，停止迭代过程，并进行反符号化处理，得到译文。
> 3. 先将整个源句子进行符号化处理，以一个固定的特殊标记作为翻译的开始符号和结束符号。此时句子变成 it is a cat .
> 4. 对序列进行建模，得到概率最大的译词，如第一个词为 “这”。将生成的词加入译文序列，重复上述步骤，不断迭代。
> 5. 直到终止符号被模型选择出来，停止迭代过程，并进行反符号化处理，得到译文。

Encoder-Decoder 的缺陷

与其说是 Encoder-Decoder 的局限，不如说是 RNN 的局限，在机器翻译中，输入某一序列，通过 RNN 将其转化为一个固定向量，再将固定序列转化为输出序列，即上面所讲的将英文翻译成中文。

不管输入序列和输出序列长度是什么，中间的「向量 c」长度都是固定的。所以，RNN 结构的 Encoder-Decoder 模型存在长程梯度消失问题，对于较长的句子，我们很难寄希望于将输入的序列转化为定长的向量而保存所有有效信息，即便 LSTM 加了门控机制可以选择性遗忘和记忆，随着所需翻译的句子难度怎能更加，这个结构的效果仍然不理想。

Attention 机制的引入

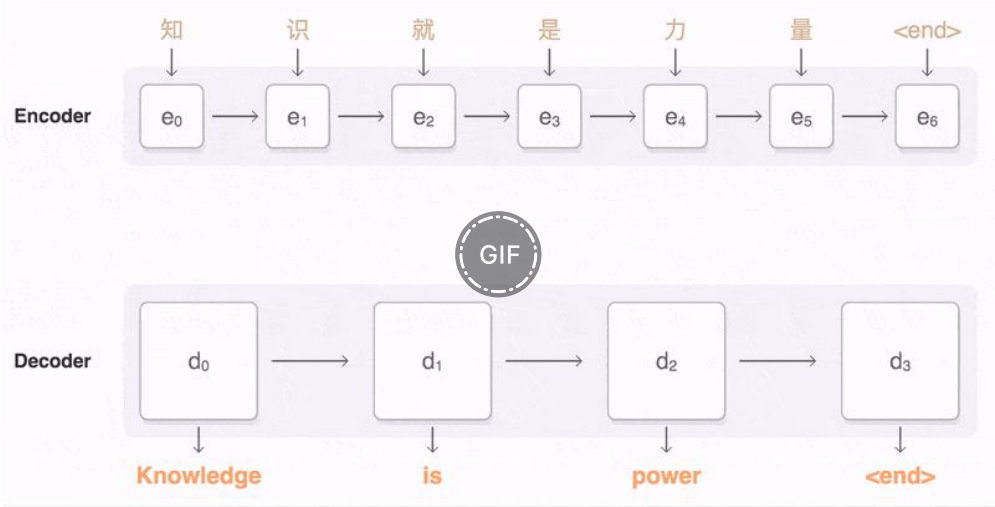
Attention 就是为了解决信息过长导致信息丢失的问题，Attention 名为注意力机制，何为注意力机制。

就像我们平时读文章一样，我们的视线不可能注意到所有的文字，随着眼睛的移动，我们的注意力[illegible]，一篇文章的标

题，小标题，我们都会第一时间注意到，这就是注意力机制。

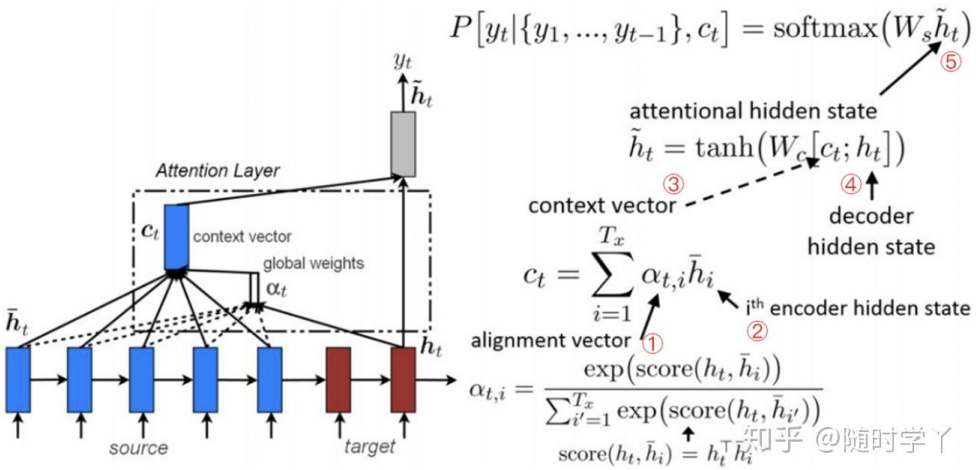
在 Attention 模型中，我们翻译当前词，会寻找于源语句中相对应的几个词语，然后结合之前已经翻译的序列来翻译下一个词。

> 当我们翻译 Knowledge 时，我们只需要将注意力集中在 "知识" 上面，翻译 “is” 的时候，只需要将注意力集中在 "就是" 上面，翻译 “力量” 时将注意力集中在 "power" 上面。
> 这样，当 Decoder 在预测目标翻译的时候，就可以看到 Encoder 的所有信息，而不仅局限于模型的定长隐向量，并且不会丢失重要信息。

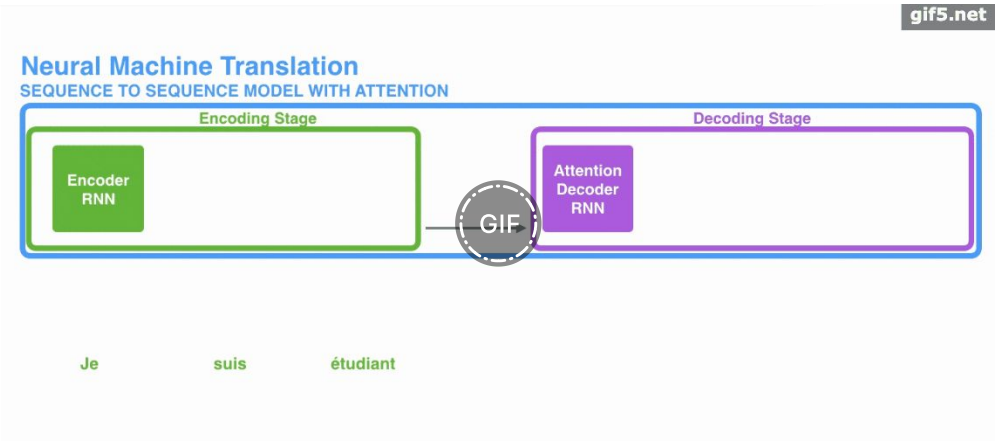


以上是对注意力机制的直观理解，那么 Attention 如何准确将注意力放在关注的地方呢?

> ① 对 RNN 的输出计算注意程度，通过计算最终时刻的向量与任意 i 时刻向量的权重，通过 softmax 计算出得到注意力偏向分数，如果对某一个序列特别注意，那么计算的偏向分数将会比较大。
> ② 计算 Encoder 中每个时刻的隐向量
> ③ 将各个时刻对于最后输出的注意力分数进行加权，计算出每个时刻 i 向量应该赋予多少注意力
> ④ decoder 每个时刻都会将 ③ 部分的注意力权重输入到 Decoder 中，此时 Decoder 中的输入有：经过注意力加权的隐藏层向量，Encoder 的输出向量，以及 Decoder 上一时刻的隐向量
> ⑤ Decoder 通过不断迭代，Decoder 可以输出最终翻译的序列。



引入 Attention 的 Encoder-Decoder 框架下，完成机器翻译任务的大致流程如下：



Transformer 中的 Encoder-Decoder

我们知道，Transformer 中的 Attention 是 Self-Attention (自注意力机制)，而且是 Multi-Head Attention (多头注意力机制)。

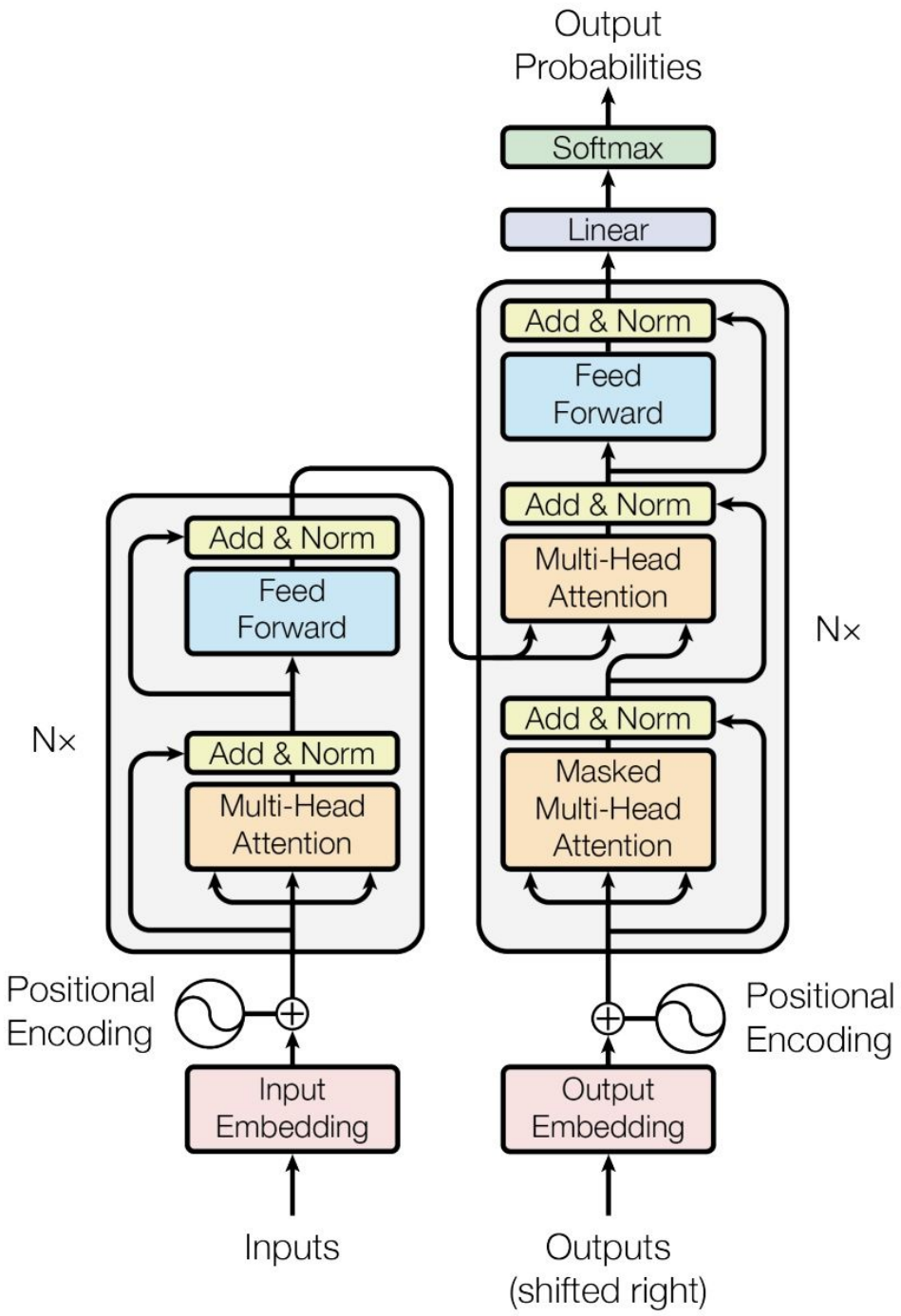
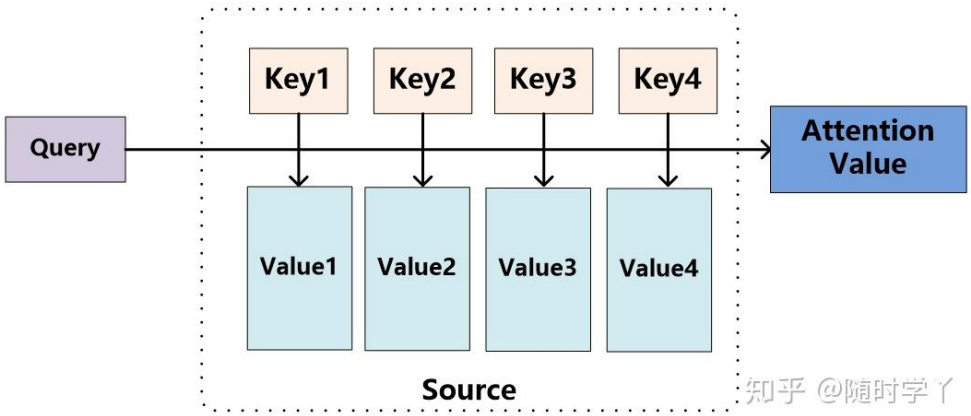


Figure 1: The Transformer - model architecture.

Attention 机制

下图可以看到，Source 是由一系列 组成，此时给定 Target 中某个元素 Query，通过计算 Query 和 各个 Key 的相似性，得到每个 Key 对 Value 的权重系数，然后对 Value 进行加权求和，即得到最终 Attention 数值。



Self-Attention

在机器翻译中，一般输入 Source 和输出 Target 内容是不一样的，如英文翻译成中文，Source 是英文，Target 是中文，Attention 机制发生在 Target 元素 和 Source 中所有元素之间。而 Self-Attention 顾名思义，指的不是 Target 和 Source 之间的 Attention 机制，而是 Source 内部元素之间或者 Target 内部元素之间发生的 Attention 机制，也可以理解为 Target = Source 的特殊情况下的 Attention 机制。

Self-Attention 机制为什么要设置为 Source = Target，也即是 Key = Value = Query?

Self-Attention 究竟学习到了哪些规律或者抽出了哪些特征呢?

引入 Self-Attention 机制有哪些好处呢?

下图是可视化的表示了 Self-Attention 在同一个英语句子内单词间产生的联系。由此可见，Self-Attention 可以捕获同一个句子中单词之间的一些句法特征 (图 1：有一定距离的短语结构) 或语义特征 (图 2：its 指代的对象 Law)。

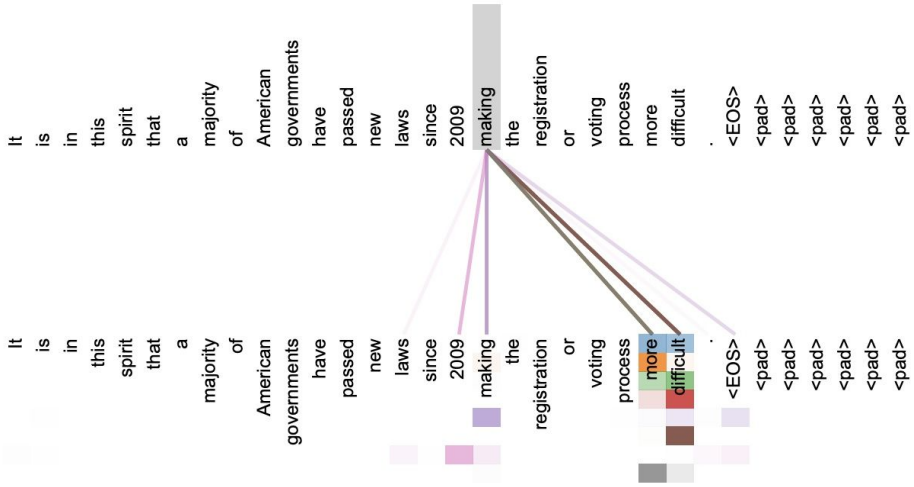


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

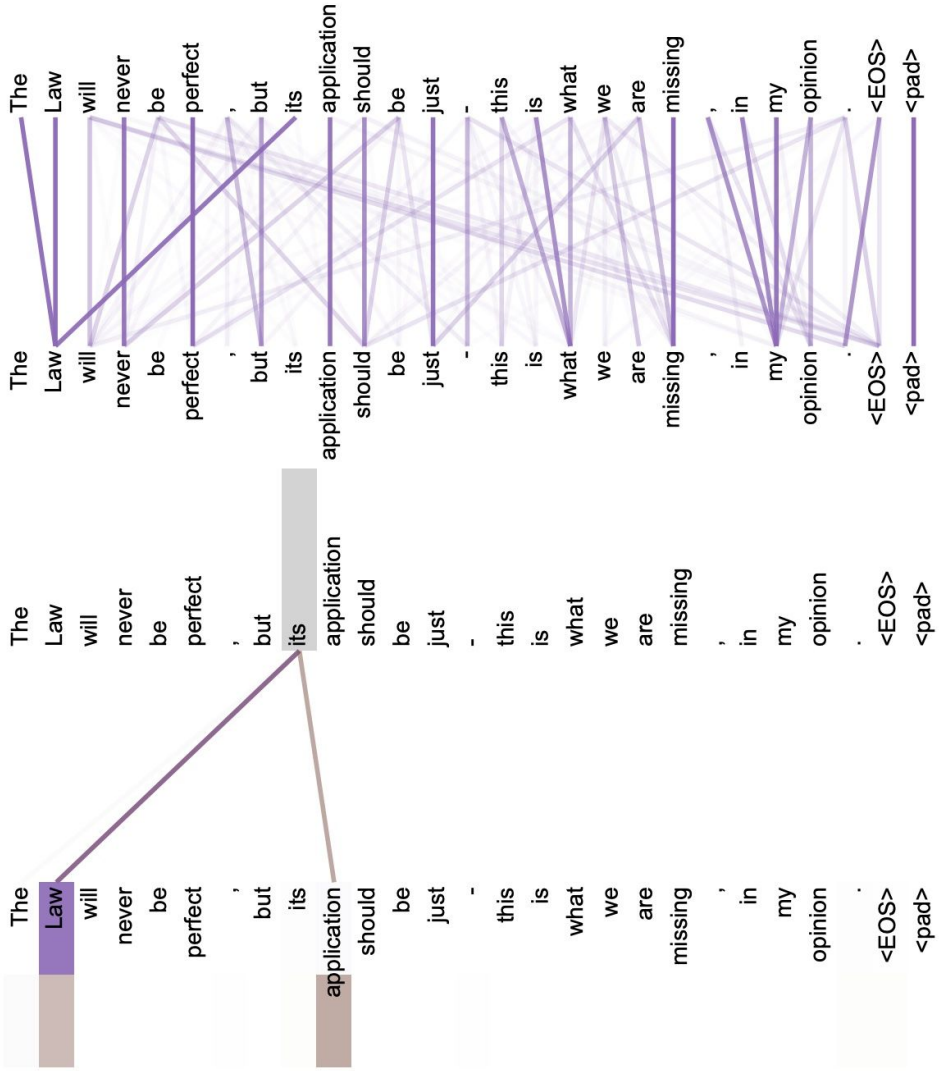


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

很明显，引入 Self-Attention 后会更容易捕获句子中长距离相互依赖特征，因为 Self-Attention 在计算过程中直接将句子任意两个单词的联系起来，此外，由于不依赖时间序列这一特性，Self-Attention 增加了计算的并行性。

Multi-Head Attention

而 Transformer 正是利用了 Self-Attention 的并行性，采用 Multi-Head Attention。因为 有好多人对 Attention 的看法不一样，那么我们就可以将这个任务给很多人一起做，最后争取大家共同的意见。

而将模型分为多个头，形成多个子空间，可以让模型去关注不同方向的信息，而模型真的会关注不同方面的特征吗?

在 Transformer 论文中，Multi-Head Attention 公式如下：

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{\text{model}}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

如果 Multi-Head 的作用是关注句子的不同方面，那么认为不同的头应该关注的 Token 不一样，但是有大量 paper 表名，Transformer 或 Bert 的特定层是有独特功能的，底层更偏向于关注语法，顶层更偏向于关注语义。既然同一层 Transformer 关注的方面是相同的，那么对于该方面而言，不同头的关注点应该也是一样的。但是我们发现，同一层中，总有那么一两个头独一无二，和其它头关注的 Token 不同。

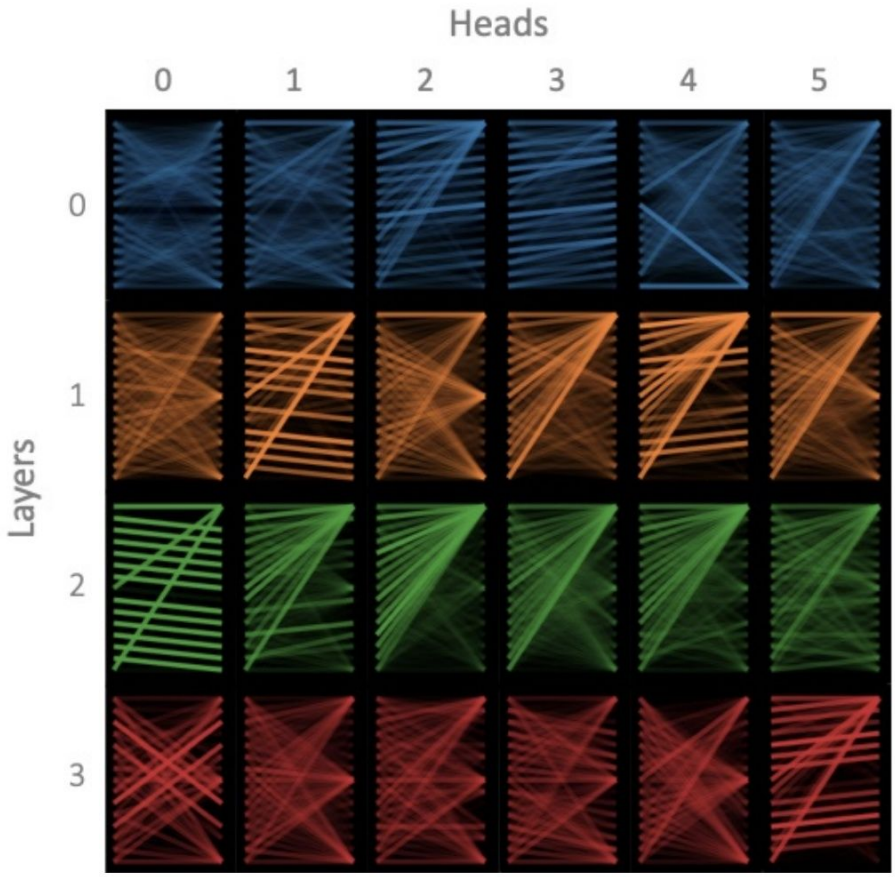


Figure 5: Model view of BERT, for same inputs as in Figure 2. Excludes layers 4–11 and heads 6–11.

下面是两个 Self-Attention 执行同一个句子时候展现的不同的注意力，利用多头机制，明显学会了不同的任务下采取不一样的注意力。

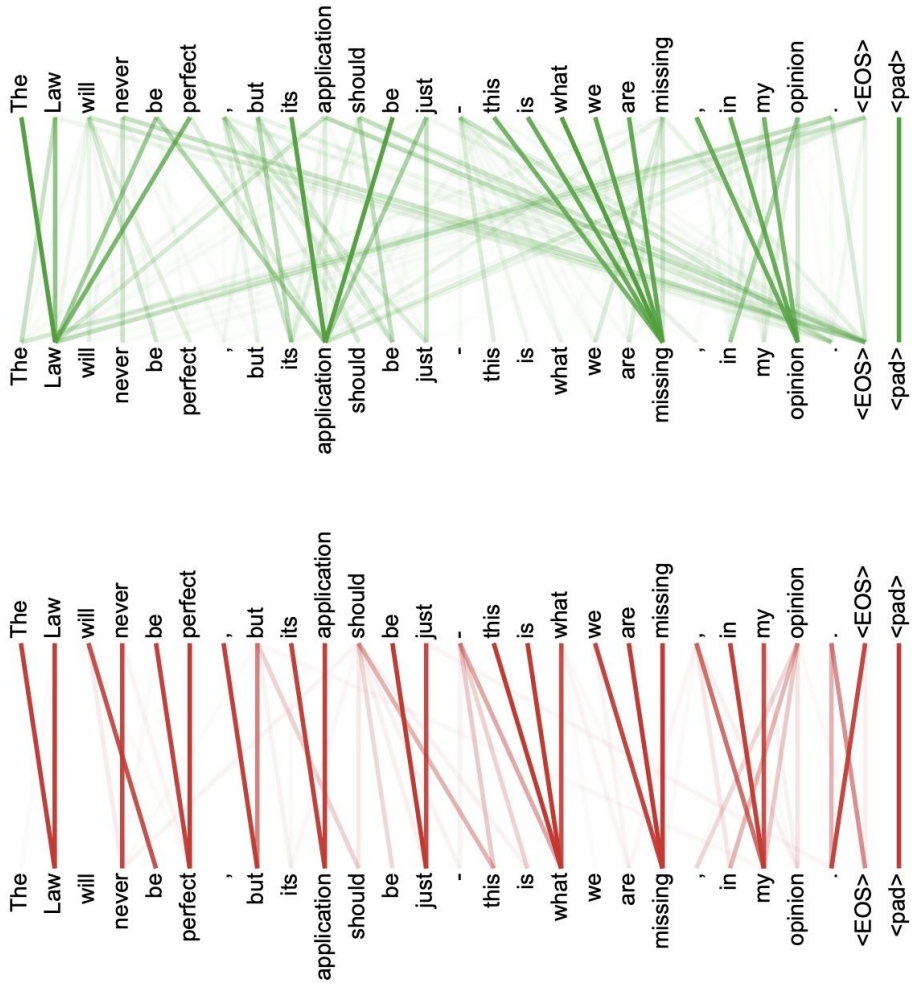


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

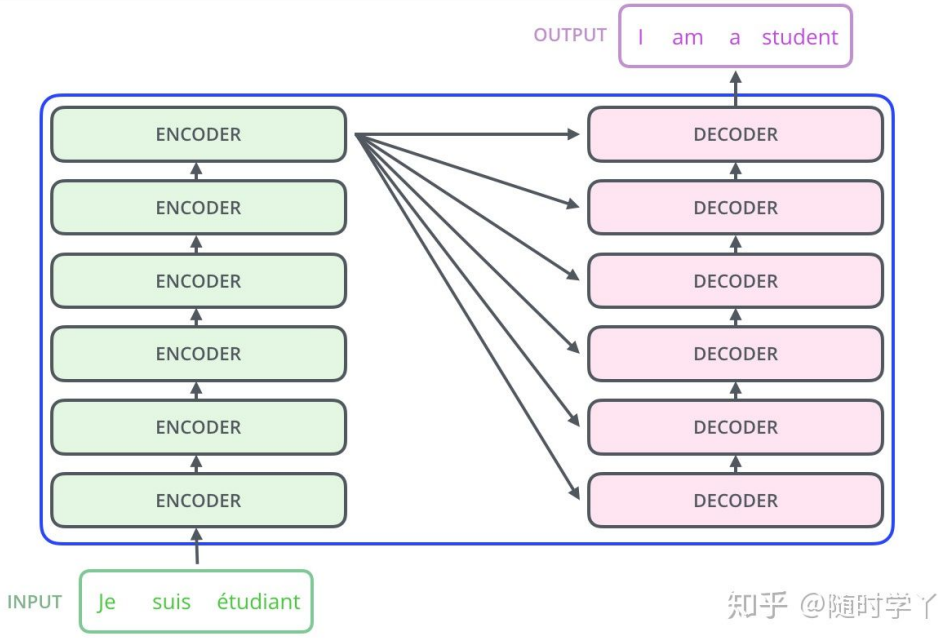
Transformer 中 Encoder 由 6 个相同的层组成，每个层包含 2 个部分：

- Multi-Head Self-Attention
- Position-Wise Feed-Forward Network (全连接层)

Decoder 也是由 6 个相同的层组成，每个层包含 3 个部分：

- Multi-Head Self-Attention
- Multi-Head Context-Attention
- Position-Wise Feed-Forward Network

上面每个部分都有残差连接 (redidual connection)，然后接一个 Layer Normalization。



以上就是 Transformer 中 Encoder 和 Decoder 的讲解。

发布于 2020-02-27

Transformer 机器翻译 自然语言处理

文章被以下专栏收录

深度学习
追逐前沿深度学习paper

推荐阅读

Transformer原理：自底向上解析

之前看了很多介绍Transformer的文章，但还是有种似是而非的感觉，直到今天看了李宏毅老师的Transformer视频才一下子恍然大悟，这里主要将自己的理解写下来，希望可以帮助大家少走些弯...

竹子 发表于竹林夜话

用放大镜看Transformer——总体和各个模块的结构到底是什...

摘要：Transformer是一种经典的基于神经网络的机器翻译模型，催生了BERT、GPT等大幅推进NLP领域发展的重要模型。本文首先对Transformer的发展史进行了简单回顾，然后概括地展示其总体结...

李鹏宇 发表于乐之

Transformer详解

一、Encoder-Decoder &a
Seq2Seq Encoder-
Decoder:Encoder-Decode
要是 NLP 领域里的概念。
值某种具体的算法，而是一
的统称。Encoder-Decode

自定义 发表于小白

还没有评论

▲ 赞同 49 ▼ 添加评论 分享 喜欢 收藏 申请转载 ···