

# Transformer 的逻辑解释

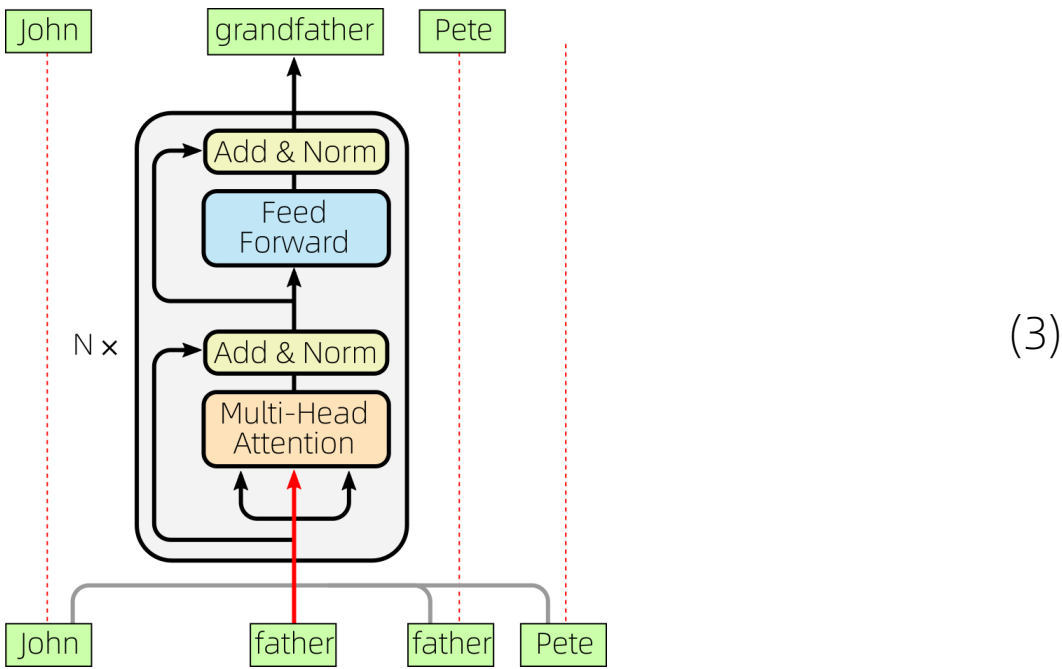
很多朋友问我, Transformer 是如何做逻辑推理? 在此尝试回答一下.  
考虑这个例子:

$$John's\ father's\ father\ is\ Pete \Rightarrow John's\ grandfather\ is\ Pete \tag{1}$$

省略一些多余的 tokens, 加入变量<sup>1</sup>:

$$\forall X, Y. X\ father\ father\ Y \Rightarrow X\ grandfather\ Y \tag{2}$$

考虑这样一个简单的  $N$ -层 Transformer, 它从左到右逐一处理 tokens, 红色表示担任 Query 的 token (注意左边和右边的 tokens 都有参与 attention, 亦即 bi-directional 处理):



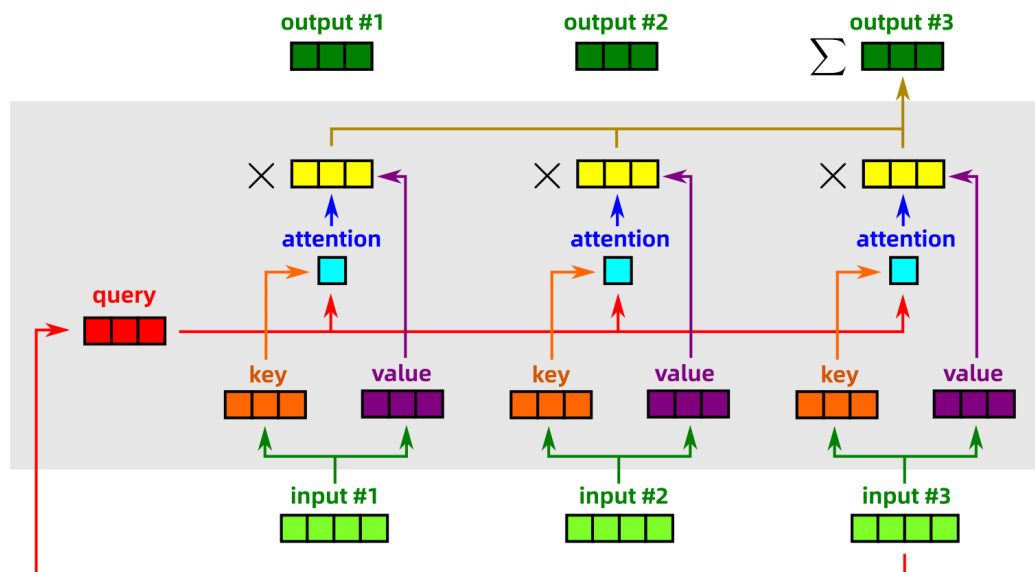
我们要回答两个问题:

- Query *father* 如何产生 *grandfather*?
- 第 4 输入位置的 *Pete* 如何出现在第 3 输出位置?

<sup>1</sup>这里使用了 **relation algebra** 的表达方式, 这种方式更接近人类语言:  $aRb$  表示  $a$  和  $b$  之间有关系  $R$ . 关系之间可以 compose, 例如  $R \circ S$ . 如果用 **谓词逻辑** 表达, 会比较累赘:  $father(X, Y) \wedge father(Y, Z) \Rightarrow grandfather(X, Z)$ . 但我们不必太拘泥于逻辑形式的细节, 因为深度学习遵从「**后结构主义**」, 它只需要 抽取逻辑结构的某些特征, 将之变成 **inductive bias** 即可.

②

如果各位同学忘记了 **self-Attention** 机制是如何运作, 可以参考下图 重温一下 (图中第 3 个输入是 **Query**):



(4)

在我们的例子 (3) 里,  $\text{Query}(\text{father}_1)$  配上  $\text{Key}(\text{father}_2)$ , 所谓「配对」是指 dot product:  $\langle \text{Query}, \text{Key} \rangle$ .

重温一下 Attention 的公式:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{\langle Q, K \rangle}{\sqrt{d_k}} \right) V \quad (5)$$

Softmax 给出的是一组概率, 也就是注意力**权重**. 例如 给予  $\text{Value}(\text{father}_1)$  30% 的权重, 给  $\text{Value}(\text{father}_2)$  70%.

所以输出的向量是  $\text{Value}(\text{father}_1)$  和  $\text{Value}(\text{father}_2)$  的**线性组合**:

$$\boxed{\text{Output}} \quad \text{grandfather} = \alpha \text{Value}(\text{father}_1) + \beta \text{Value}(\text{father}_2) \quad (6)$$

留意上式中, *grandfather* 那些是 **词向量**, 即 vector embedding.

其实  $\text{Value}(\text{father}_1)$  和  $\text{Value}(\text{father}_2)$  是一样的, 如果不考虑 positional embedding.

目前为止, 以上的操作是 Transformer 可以轻易做到的, so far so good 😊

## “Copy” Mechanism

第二个问题是 将 *Pete* 从第 4 位置 “copy” 到第 3 位置.

Query 是第 3 位置的  $father_2$ .

用更简洁的符号表示, 并略去 softmax:

$$\forall \vec{x}. \quad \vec{x} = \langle Qf_2, Kf_1 \rangle Vf_1 + \langle Qf_2, Kf_2 \rangle Vf_2 + \langle Qf_2, K\vec{x} \rangle V\vec{x} \quad (7)$$

代入  $\vec{x} = 0$  可得 前两项 = 0, ie:

$$\forall \vec{x}. \quad \vec{x} = \langle Qf_2, K\vec{x} \rangle V\vec{x} \quad (8)$$

很明显  $\vec{x}$  需要是矩阵  $V$  的 eigen-vector. 但如果  $\vec{x}$  可取的值的数目  $>$  embedding dimension, 那么  $V$  必然是  $k \cdot I$  的形式. 那么  $\langle Qf_2, K\vec{x} \rangle = 1/k = \text{constant}$  表示  $K$  将  $\vec{x}$  投射到一个超平面上.

$\vec{x}$  的取值可能只是 embedding 空间的一个子集, 因此 约束 可能更宽松.

但如果上述条件满足, 那么 (7) 的前两项不会是 0 而是  $f_1 + f_2$ , 导致矛盾. 换句话说, copy 的条件无法满足. 但注意, 我们忽略了 softmax 的作用!

似乎满足 copy 的方法是: softmax 令某些 dot products 的值是 practically zero, 所以实际  $V\vec{x}$  那项的注意力接近 100%.

## Multiple Logic Rules

现在来考虑, 一层的 Transformer 能不能储存 多个逻辑 rules?

这是重要的, 因为要达到 人类 common sense 的知识, 估计需要的逻辑 rules 数量, 起码达到 百万或千万以上. BERT/GPT 的成功给了我们一个大约的估算.

考虑 这两条不同的 rules:

- $father\ sister \Rightarrow aunt$
- $church\ sister \Rightarrow nun$

这等于要约束:

- $aunt = \gamma V father + \delta V sister$
- $nun = \epsilon V church + \zeta V sister$

于是我们看到, 不同的 逻辑前提之下, 会产生不同的 Values 的线性组合, 这表达的能力是很高的.

而且 还不要忽略 MLP 层的 非线性作用.

④

Transformer 的逻辑解释

Multi-Head Attention