①

Transformer 完全符合逻辑结构

承上文：我提出了用逻辑结构加速 Transformer 的方法，但详细分析之后，竟然发现 Transformer 跟逻辑结构是完全重合的！这结果令人失望，但也加深了我们对 Transformer 和逻辑结构的认识。

以下 讲述我思考 "logic Transformer" 的细节：

逻辑有「双层结构」：命题由概念原子构成，命题是可交换的，而概念不可交换。

逻辑系统的 **状态** 是一堆命题的集合，也就是 "sequence of sequences" 的结构。

命题内部层次 (sub-propositional level)

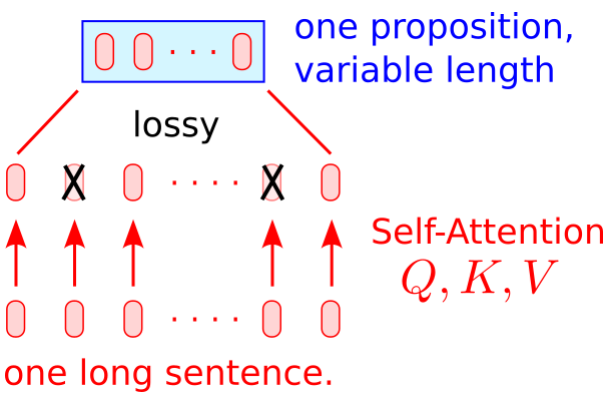
第一层 "lossy Self-Attention" 将一个 长句子 分拆成 n 个命题，命题可以由数量不同的 概念表示。

例如，句子「拜登 2 月 在白宫 就中美贸易问题发言」可以分拆为：

- 「拜登在白宫」
- 「拜登发言」
- 「发言关于中美贸易问题」
- 「发言发生在 2 月」

等 几个 **较短的** 命题。n 的数目 视乎 句子复杂的程度而定。

我初时提出，每个 短命题 可以用一个 "lossy Self-Attention" 产生：（输出的 tokens 数目比输入少）



(1)

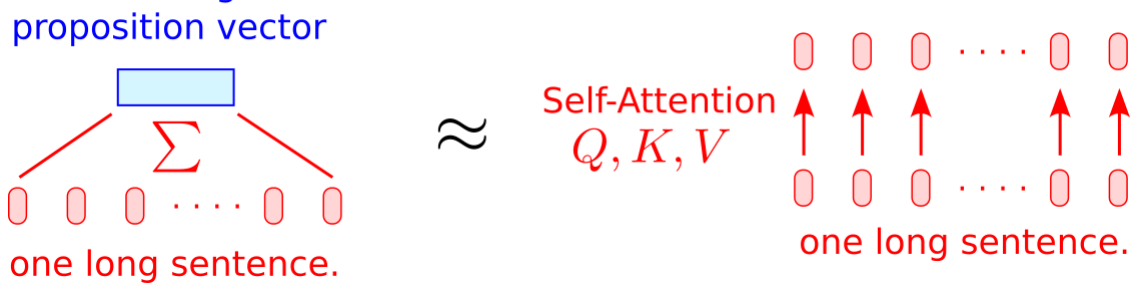
但这里出了问题：lossy 的意思是 将某些 输入的 tokens 省略掉，但这个动作是 **不可微**的，因为它改变了 向量的维数，而维数是 拓扑不变量，意思是说：任何（可逆的）连续映射不能改变空间的维数，而可微函数只是连续映射的一个特例。除非放弃可逆性，那就表示 两个不同的自然语言句子 有可能映射到同一个命题。

我们需要的映射是：sequence → shorter sequence. 例如，由 10 个 tokens 变成 3 个 tokens：

$$\overbrace{\text{tok} \times \text{tok} \times \ldots \times \text{tok}}^{\text{10 times}} \to \text{tok} \times \text{tok} \times \text{tok} \tag{2}$$

这两边的维数肯定是不同的，所以 injectiveness 必须放弃。也就是说，两个不同的长序列 有可能映射到同一个短序列[1]。

有个简单的解决办法是：将自然语言句子映射到 一个长度**固定**的向量，它的内容是几个意思的**叠加**（图左）：



(3)

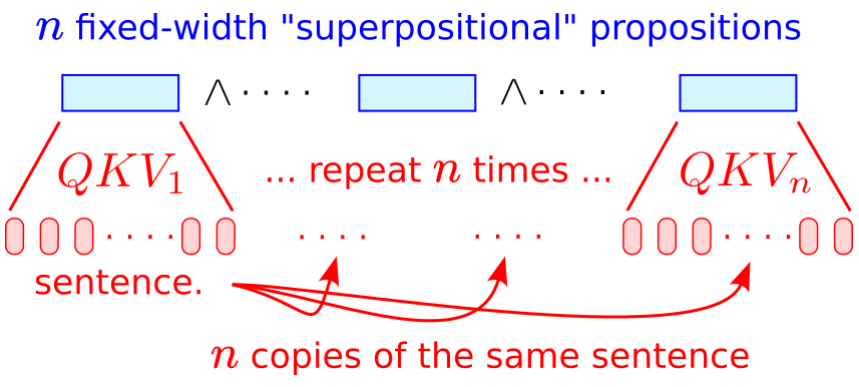
但传统 Transformer 的做法 就是这种做法的一个特例！Dot product similiarity 乘以每个 token 的 V 值，然后 **相加** 得到输出。这就是一种将概念叠加的逻辑命题。换句话说，输出的 ▯ 是一个 ▭。这很合理，因为在逻辑里，可交换的东西就是命题（若命题是真的，枚举它们的次序不重要），而 Self-Attention 的特点就是 equivariance.

但概念的叠加 丧失了 **不可交换性**：我爱你 = 你爱我，世界没有失恋。但似乎可以绕过这问题：例如「我爱你」可以分拆为三个子命题：

- 爱 = 动词
- 我 = 主语
- 你 = 宾语

这样分拆之后，命题内部不再需要次序（也就是可交换的）。这是一种特殊的逻辑语法，看来未尝不可。

将以上 (3) 的结构 横向重复 n 次：（简单起见，(Q, K, V) 简写为 QKV）



(4)

如果每个 QKV_i 都是一样的，那就是传统的 Transformer 结构；如果每个 QKV_i 不同，那就对应于 Multi-Head Attention.

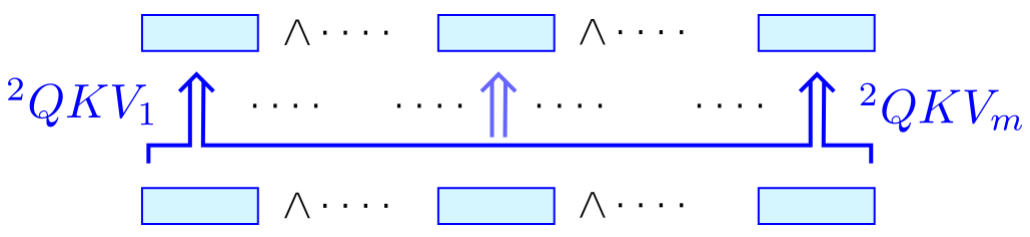
以前说过，每个 QKV_i 可以看成是一个 rule base，它们是独立运作的。Multi-Head Attention 相当于说，同一堆前提可以推出不同的结论，即

$$A \wedge B \wedge C \wedge \ldots \to X \text{ or } Y \tag{5}$$

这导致逻辑推论可能出现分歧，这种分歧似乎没有太大效益，因为我曾听说 Multi-Head Attention 的功效并不显著。在 Prolog 语言里也没有这种 "or" 语法，但 Prolog 作为编程语言还是 OK 的。

命题层次 (Propositional level)

第二层 我称为 "symmetric Self-Attention"，它比较简单，而且**不需要 位置编码**：



(6)

其实在 传统的多层 Transformer 里面，除了第一层也没有位置编码。这也表示，传统 Transformer 跟 逻辑结构 完全重合。

---

[1]这说法有点不准确：因为深度学习用了所谓 "embedding"，例如 Word2vec，这种嵌入的「维数」没有良好定义。在计算机上实践时，我们惯用 维数 = 512，但那是可以改变的。那么我们也可以改变 embedding 的维数，将多个 tokens 挤进较小的空间里，也是可以的，但似乎比较麻烦。厘清这种 embedding 的数学意义（它是不是有向量空间结构？微分流型结构？连续性跟离散的符号有何关系?），是一个很重要的问题，日后再谈....


2