

《The road to AGI》

YKY 甄景贤

December 20, 2019

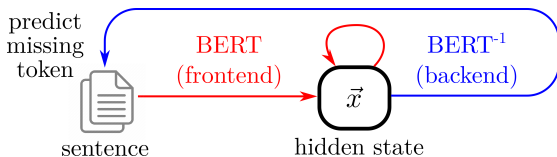
Table of contents

- 1 BERT 的革命性意义
- 2 从 BERT 过渡到 AGI
- 3 「集」结构 带来麻烦
- 4 Attention 是什么？
- 5 Attention 在逻辑中的用处
- 6 要设计一种新的 attention

多谢 支持 😊

BERT 的革命性意义

- BERT 利用平常的文本 induce 出知识，而这 representation 具有 通用性 (universality) :



换句话说：隐状态的 representation 压缩了句子的意思，而它可以应用在别的场景下

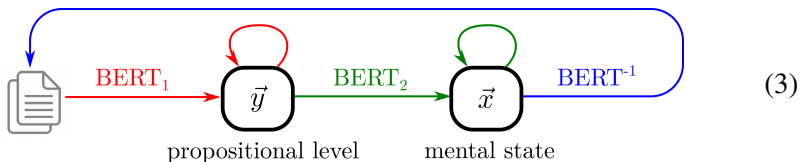
- This implies that human-level AI can be *induced* from existing corpora, 而不需要重复像人类婴儿成长的学习阶段
- Such corpora can include items such as images, movies with dialogues / subtitles
- 这种训练方法是较早的另一篇论文提出，它并不属于 BERT 的内部结构

从 BERT 过渡到 AGI

- 词语 组成 句子，类比於 逻辑中，概念 组成 逻辑命题
- 抽象地说，逻辑语言 可以看成是一种有 2 个运算的 代数结构，可以看成是 加法 \wedge 和 乘法 \cdot ，其中 乘法 是不可交换的，但加法 可交换
- 例如 两个命题：

$$\text{我} \cdot \text{爱} \cdot \text{妳} \wedge \text{妳} \cdot \text{爱} \cdot \text{我} \quad (2)$$

- 这种逻辑结构 可以用 两层 的 BERT 模型 处理：

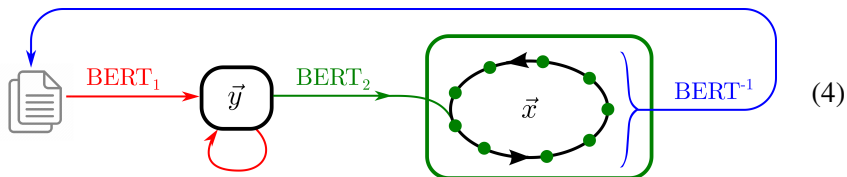


(第一层似乎可以纳入到第二层，简化整个模型)

- 我发现 最难处理的问题，是在第 2 层 的状态 \vec{x} . 它是一个 逻辑命题 的 集合，集合中元素有可交换性，亦即 **permutation invariance**. 这看似简单的特性，其实带来很大的麻烦

「集」结构带来麻烦

- Word2Vec 也是革命性的；由 Word2Vec 演变成 Sentence2Vec 则比较容易，基本上只是向量的延长 (concatenation)；逻辑命题 类似於 sentence
- 假设 全体逻辑命题的空间是 \mathbb{P} ，则 命题集合 的空间是 $2^{\mathbb{P}}$ ，非常庞大
- 如果限制 状态 $\vec{x} = \text{working memory}$ 只有 10 个命题， \vec{x} 的空间是 \mathbb{P}^{10} / \sim 其中 \sim 是对称群 \mathfrak{S}_{10} 的等价关系。换句话说 $2^{\mathbb{P}} \cong \coprod_{n=0}^{\infty} \mathbb{P}^n / \mathfrak{S}_n$
- $\mathbb{P}^n / \mathfrak{S}_n$ 虽然是 \mathbb{P}^n 的商空间，但 \mathfrak{S}_n -不变性 很难用神经网络实现
- 现时 比较可行的办法，是将 状态 \vec{x} 实现成一个时间上的「轮盘」，每个●表示一个命题：



- 有趣的是，如果用「轮盘」方法，BERT 的 注意力机制 有特殊意义....

Attention 是什么？

- 注意力 最初起源於 Seq2seq，后来 BERT 引入 self-attention
- 在 Seq2seq 中，编码器 (encoder) 由下式给出，它将输入的词语 x_i 转化成一连串的隐状态 h_i ：

$$h_t = \text{RNN}_{\text{encode}}(x_t, h_{t-1}) \quad (5)$$

- 这些 h_i 可以综合成单一个隐状态 $c = q(h_1, \dots, h_n)$.
- 这个 c 被「寄予厚望」，它浓缩了整个句子的意义
- 解码器 的结构类似，它的隐状态是 s_t ，输出 y_t ：

$$s_t = \text{RNN}_{\text{decode}}(y_t, s_{t-1}, c_t) \quad (6)$$

- 注意最后的 c_t 依赖时间，它是隐状态 h_j 的 加权平均：

$$c_i = \sum_j \alpha_{ij} h_j \quad (7)$$

- 其中 α_{ij} 量度 输入 / 输出 的隐状态之间的 相似度，取其最大值：

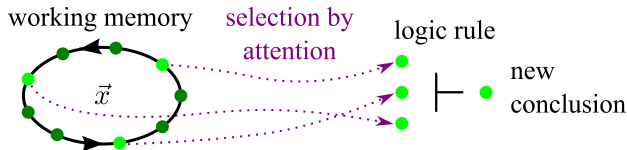
$$\alpha_{ij} = \text{softmax}\{\langle s_i, h_j \rangle\} \quad (8)$$

换句话说， α_{ij} 选择 最接近 h_j 的 s_i

Attention 在逻辑中的用处

我这样理解 attention:

- 例如，翻译时，输入 / 输出句子中「动词」的位置可以是不同的
- 当 解码器 需要一个「动词」时，它的隐状态 s_t 含有「动词」的意思
- Attention 机制 找出最接近「动词」的 编码器的隐状态（可以 ≥ 1 个） $\sum h_j$ ，交给 解码器，这是一种 **information retrieval**
- 例如，将 M 件东西 映射 到 N 件东西，可以有 N^M 个 mappings，这是非常庞大的空间。但如果这些物件有 **类别**，而 同类只映射到同类，则可以用 attention 简化 mappings
- 所以 attention 是一种 inductive bias，它大大地缩小 mapping 空间
- 在逻辑的场景下，需要的 mapping 是 $f : \text{命题集合} \rightarrow \text{命题}$



要设计一种新的 attention

- 逻辑 attention 和 传统 attention 要求略有不同，这是关键的一步
- 不是「同类映射到同类」，而是要在庞大的 logic rules 空间中找到适用 (applicable) 的 rules
- 隐状态 s_t 代表 “search state”，注意力的目的是 选择 s_t 所需要的那些命题，交给 解码器
- 注意：逻辑 attention 从 M 个命题中选择 N 个命题， $M > N$. 这是 inductive bias. 而 Symmetric NN 的做法，只是要求 M 个命题的 置换不变性，所以它浪费了资源在很多 “don't care” 的命题上
- 换句话说，selection 所带来的 bias 如果足够强，似乎不需要 symmetric. 很巧合地，再次应验了 “attention is all you need” 这句话

多谢收看 😊