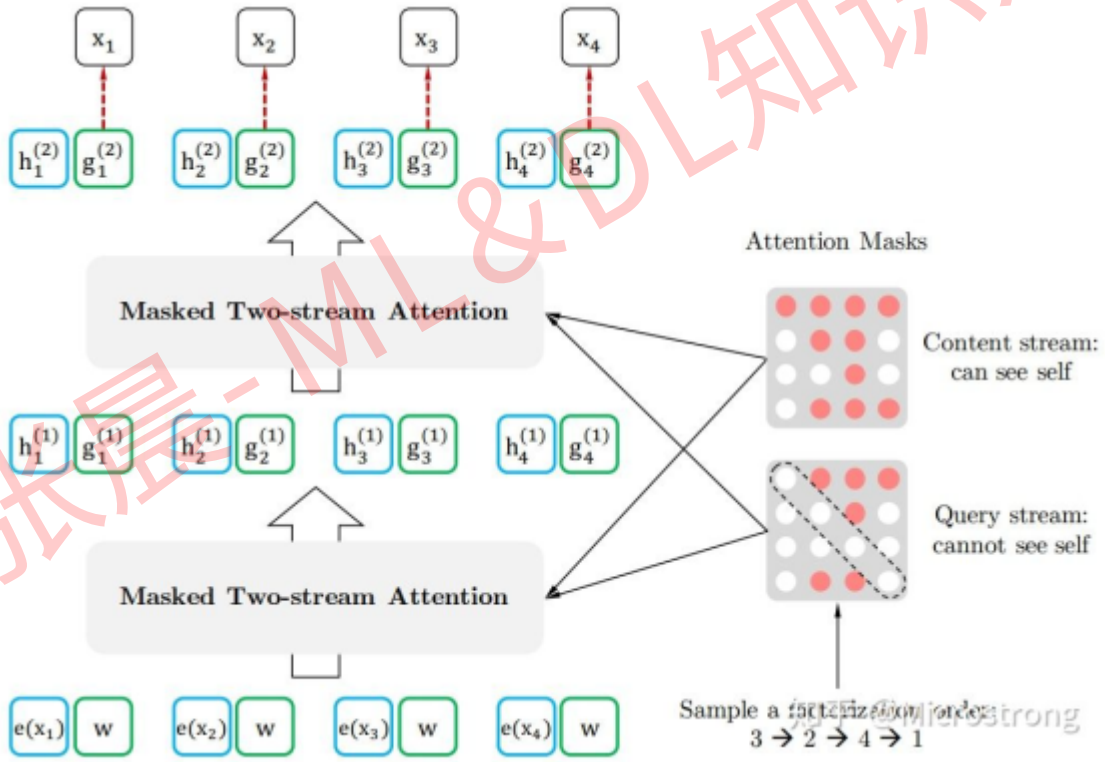
8. XLNET

1．模型结构

BERT本身很有效，但它本身也存在一些问题，比如不能用于生成、以及训练数据和测试数据的不一致(Discrepancy)。比BERT更强大的预训练模型-XLNet，它为了达到真正的双向学习，采用了Permutation语言模型、以及使用了双流自注意力机制，并结合了Transformer-XL的相对位置编码。



2．AR与AE模型

AR（Autoregressive Language Modeling）：指的是，依据前面（或后面）出现的tokens来预测当前时刻的token，代表有 ELMO， GPT等。

- 优点：对生成模型友好，天然符合生成式任务的生成过程。这也是为什么 GPT 能够编故事的原因。
- 缺点：它只能利用单向语义而不能同时利用上下文信息。 ELMO 通过双向都做AR 模型，

然后进行拼接，但从结果来看，效果并不是太好。

AE（Autoencoding Language Modeling）：通过上下文信息来预测被mask的token，代表有 BERT , Word2Vec(CBOW) 。

- 优点：能够很好的编码上下文语义信息（即考虑句子的双向信息）， 在自然语言理解相关的下游任务上表现突出。
- 缺点：由于训练中采用了 [MASK] 标记，导致预训练与微调阶段不一致的问题。BERT独立性假设问题，即没有对被遮掩（Mask）的 token 之间的关系进行学习。 此外对于生成式问题， AE 模型也显得捉襟见肘。

3．BERT的缺点

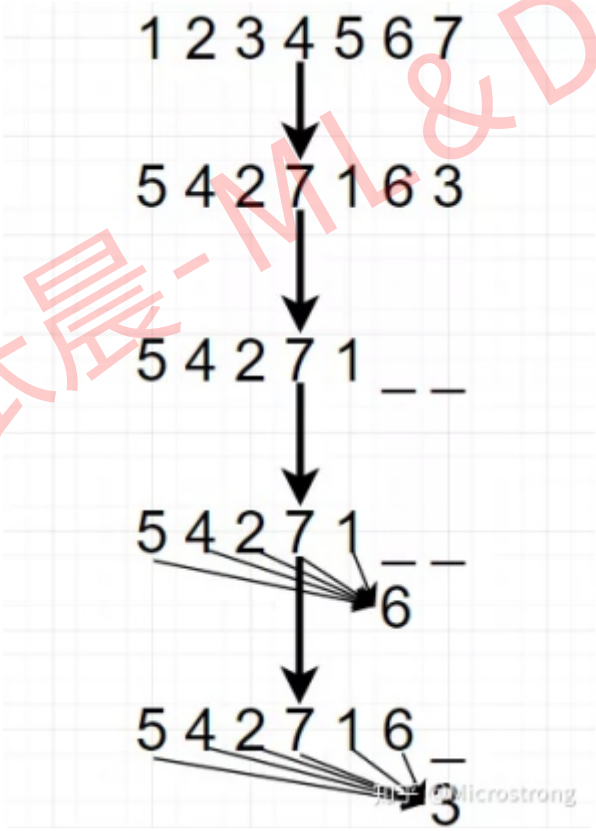
- 由于训练中采用了 [MASK] 标记，导致预训练与微调阶段不一致的问题
- BERT独立性假设问题，认为被（Mask）的 token 之间是独立的

4．XLNET的基本思想

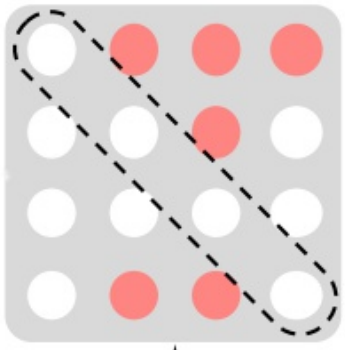
使模型融合AR及AE的优点。如何设计模型，才能使其**看上去仍然是从左到右的输入和预测模式，但是内部已经关注到了上下文的信息。**

5．置换语言模型（Permutation Language Model）

通过随机取一句话的一种排列，然后将末尾一定量的词给“遮掩”（和 BERT 里的直接替换 [MASK] 有些不同）掉，最后用 AR 的方式来按照这种排列依次预测被“遮掩”掉的词。



论文中 Permutation 具体的实现方式是通过直接对 Transformer 的 **Attention Mask** 进行操作。



Query stream:
cannot see self

Sample a factorization order:
3 → 2 → 4 → 1

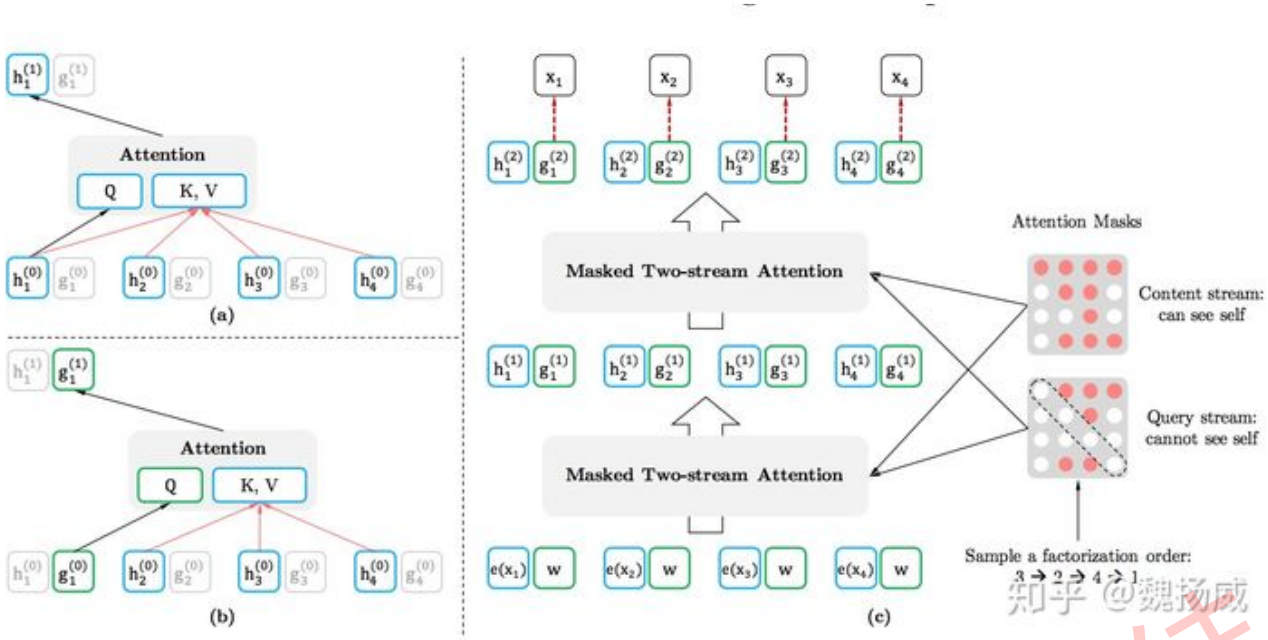
比如说序号依次为 1234 的句子，先随机取一种排列3241。于是根据这个排列我们就做出类似上图的 Attention Mask。先看第1行，因为在新的排列方式中 1 在最后一个，根据从左到右 AR 方式，1 就能看到 234 全部，于是第一行的 234 位置是红色的（没有遮盖掉，会用到），以此类推。第2行，因为 2 在新排列是第二个，只能看到 3，于是 3 位置是红色。第 3 行，因为 3 在第一个，看不到其他位置，所以全部遮盖掉...

6．双流自注意力机制（Two-Stream Self-Attention）

为了在不引入当前word信息的前提下把位置信息引入，也为了在下层计算时，当前的内容信息不缺失，引入了双流自注意力机制机制。content stream用来记录content信息，即历史content+当前位置的content信息，用于保留当前的内容信息；另外一个query stream用来记录历史content+当前位置信息，用于计算当前位置的正常输出。最后一层预测的时候使用query stream中的隐变量g来进行预测。

$$h_{z_t}^{(m)} = Attention(Q = h_{z_t}^{(m-1)}, KV = h_{z \le t}^{(m-1)}; \theta)$$

$$g_{z_t}^{(m)} = Attention(Q = g_{z_t}^{(m-1)}, KV = h_{z < t}^{(m-1)}; \theta)$$



7．部分预测

XLNet还使用了部分预测（Partial Prediction）的方法。因为LM是从第一个Token预测到最后一个Token，在预测的起始阶段，上文信息很少而不足以支持Token的预测，这样可能会对分布产生误导，从而使得模型收敛变慢。为此，XLNet只预测后面一部分的Token，而把前面的所有Token都当作上下文。具体来说，对长度为 T 的句子，我们选取一个超参数 K ，使得后面 $1/K$ 的Token用来预测，前面的 $1-1/K$ 的Token用作上下文。注意， K 越大，上下文越多，模型预测得就越精确。

8．位置编码

绝对位置编码：

Transformer使用的是绝对位置编码，如果我们继续使用absolute positing encoding的话，对于所有的sequence序列，只要这个字在序列中的位置一样的话，它的position encoding也会一样，这样的话，对于我们concat之后的输出，我们无法区别每个字的位置。

相对位置编码：

Transformer-XL 首先分析了position encoding在计算中的作用，然后根据这个结果将交互项转化为relative position encoding。

$$
\begin{aligned}
(QK^T)_{i,j} &= (E+P)_{i,\bullet} W^Q (W^K)^T (E+P)_{\bullet,j}^T \\
&= (E+P)_{i,\bullet} W^Q (W^K)^T (E^T+P^T)_{\bullet,j} \\
&= E_{i,\bullet} W^Q (W^K)^T (E^T+P^T)_{\bullet,j} + P_{i,\bullet} W^Q (W^K)^T (E^T+P^T)_{\bullet,j} \\
&= \underbrace{E_{i,\bullet} W^Q (W^K)^T E_{\bullet,j}^T}_{a)} + \underbrace{P_{i,\bullet} W^Q (W^K)^T P_{\bullet,j}^T}_{b)} + \underbrace{E_{i,\bullet} W^Q (W^K)^T P_{\bullet,j}^T}_{c)} + \underbrace{P_{i,\bullet} W^Q (W^K)^T E_{\bullet,j}^T}_{d)}
\end{aligned}
$$

- a) 这一项中没有包含 P 位置信息，代表的是在第 i 行的字应该对第 j 列的字提供多大的注意力。这是不管他们两个字的位置信息的。
- b) 这一项捕获的是模型的global attention，指的是一个字在position i 应该要对 position j 付出多大的注意力。例如两个字的位置越远，期望它们之间的注意力越小。
- c) 这一项捕获的是在row i的字对其他位置的关注信息，例如在position i是一个字"狗"， 应该要对j=i-1 这个位置特别注意，否则可能出现j=i-1是“热”， 出现是“热狗”的情况。
- d) 这个是c) 的逆向表示，指的是j的字要pay attention to 位置i的字。

为了通过输入形式 **[A, SEP, B, SEP, CLS]** 来处理句子对任务，于是需要加入标识 A 句和 B 句的段信息。BERT 里面很简单，直接准备两个向量，一个加到 A 句上，一个加到 B 句上。

但当这个遇上 Segment Recurrence Mechanism 时，和位置向量一样，也出问题了。万一出现了明明不是一句，但是相同了怎么办，于是我们就需要最后一块补丁，同样准备两个向量，**s+** 和 **s-** 分别表示在一句话内和不在一句话内。

具体实现是在计算 attention 的时候加入一项：

$$s_{ij} = \begin{cases} s_+ \text{ if } i, j \text{ in same segment} \\ s_- \text{ if } i, j \textbf{ not } \text{in same segment} \end{cases}$$
$$a_{ij} = (q_i + b)^T s_{ij}$$

当 i 和 j 位置在同一段里就用 s+，反之用 s-，在 attention 计算权重的时候加入额外项。

9．XLNET与BERT的区别

- Mask的位置，Bert更表面化一些，XLNet则把这个过程隐藏在了Transformer内部而已
- BERT是典型的AE模型，而XLNET是AE与AR的结合
- XLNET更快，因为只预测后面一部分的词语
- BERT认为mask的词是独立的，而XLNET认为词之间是有联系的

10．XLNET的改进

- 比BERT增加了训练集
- 引入了新的优化目标Permutation Language Modeling（PLM）
- 使用了双流自注意力机制（Two-Stream Self Attention, TSSA）和与之匹配的Mask技巧
- XLNet还使用了Transformer-XL作为Backbone，相对位置编码以及分段RNN机制

- [ ] https://zhuanlan.zhihu.com/p/81039057 <https://zhuanlan.zhihu.com/p/81039057>
- [ ] https://zhuanlan.zhihu.com/p/110204573 <https://zhuanlan.zhihu.com/p/110204573>