Transformer-XL

本文发表在ACL2019上.

论文想要解决的问题：如何赋予编码器捕获长距离依赖的能力。目前在自然语言处理领域，Transformer的编码能力超越了RNN，但是对长距离依赖的建模能力仍然不足。在基于LSTM的模型中，为了建模长距离依赖，提出了门控机制和梯度裁剪，目前可以编码的最长距离在200左右。在基于Transformer的模型中，允许词之间直接建立联系【self-attention】，能够更好地捕获长期依赖关系，但是还是有限制。

Motivation

Transformer编码固定长度的上下文，即将一个长的文本序列截断为几百个字符的固定长度片段(segment)，然后分别编码每个片段[1]，片段之间没有任何的信息交互。比如BERT，序列长度的极限一般在512。**动机总结如下：**

- Transformer**无法建模超过固定长度的依赖关系**，对长文本编码效果差。
- Transformer把要处理的文本分割成等长的片段，通常不考虑句子（语义）边界，导致**上下文碎片化(context fragmentation)**。通俗来讲，一个完整的句子在分割后，一半在前面的片段，一半在后面的片段。

文章围绕如何建模长距离依赖，提出Transformer-XL【XL是extra long的意思】：

- 提出**片段级递归机制(segment-level recurrence mechanism)**，引入一个**记忆(memory)**模块（类似于cache或cell），循环用来建模片段之间的联系。
  - 使得长距离依赖的建模成为可能；
  - 使得片段之间产生交互，解决上下文碎片化问题。
- 提出**相对位置编码机制(relative position embedding scheme)**，代替绝对位置编码。
  - 在memory的循环计算过程中，避免时序混淆【见model部分】，位置编码可重用。

小结一下，片段级递归机制为了解决编码长距离依赖和上下文碎片化，相对位置编码机制为了实现片段级递归机制而提出，解决可能出现的时序混淆问题。

Model

Vanilla Transformer

> 每个segment分别编码，相互之间不产生任何交互。

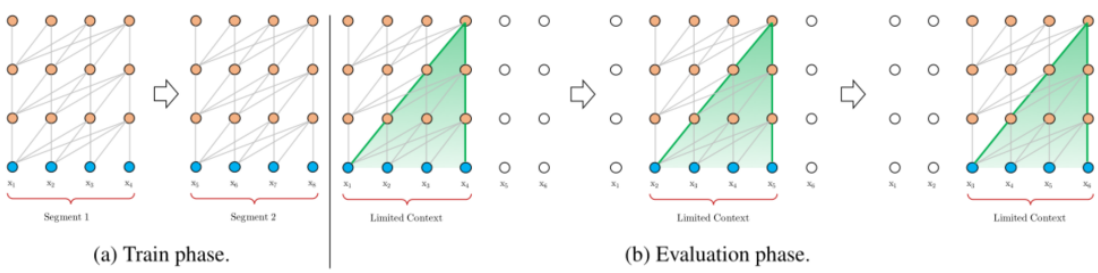
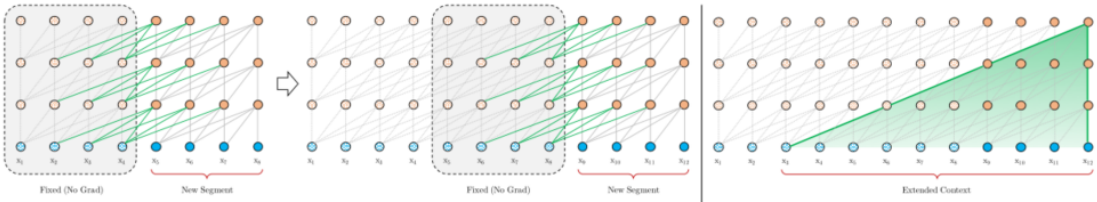


Figure 1: Illustration of the vanilla model with a segment length 4.

segment-level recurrence mechanism

> 为了解决长距离依赖，文章引入一个memory状态。
>
> 在训练过程中，每个片段的表示为最后的隐层状态，表示片段的序号，表示片段的长度，表示隐层维度。
>
> 在计算片段的表示时，用memory缓存片段层的隐层状态，用来更新，这样就给下一个片段同了上文，长距离依赖也通过memory保存了下来。并且，最大可能的依赖长度线性增长，达到 N*L 。



(a) Training phase. (b) Evaluation phase.

Figure 2: Illustration of the Transformer-XL model with a segment length 4.

relative position embedding scheme

在实现片段级递归时遇到一个问题：如果采用绝对位置编码，不同片段的位置编码是一样的，这很显然是不对的。公式如下：

$$\mathbf{h}_{\tau+1} = f\left(\mathbf{h}_{\tau}, \mathbf{E}_{\mathbf{s}_{\tau+1}} + \mathbf{U}_{1:L}\right)$$

$$\mathbf{h}_{\tau} = f\left(\mathbf{h}_{\tau-1}, \mathbf{E}_{\mathbf{s}_{\tau}} + \mathbf{U}_{1:L}\right)$$

$\mathbf{E}_{\mathbf{s}_{\tau}}$ 表示片段 $\mathbf{s}_{\tau}$ 的词向量， $\mathbf{U}_{1:L}$ 表示绝对位置向量，可以看出，两个片段之间所用的位置向量是一样的。如果一个词出现在两个片段中 $x_{\tau,j}$ 、 $x_{\tau+1,j}$ ，按照绝对位置编码方式，它们的表示向量是一样的，难以区分。

因此，本文引入相对位置编码机制，计算self-attention公式如下：

$$\mathbf{A}^{\mathrm{rl}}_{i,j} = \underbrace{\mathbf{E}^{\top}_{x_i}\mathbf{W}^{\top}_{q}\mathbf{W}_{k,E}\mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}^{\top}_{x_i}\mathbf{W}^{\top}_{q}\mathbf{W}_{k,R}\mathbf{R}_{i-j}}_{(b)} + \underbrace{u^{\top}\mathbf{W}_{k,E}\mathbf{E}_{x_j}}_{(c)} + \underbrace{v^{\top}\mathbf{W}_{k,R}\mathbf{R}_{i-j}}_{(d)}$$

1. 引入相对位置编码，用的是Transformer里用的sinusoid encoding matrix，不需要学。
2. u 和 v 是需要学习的参数，这是这部分的关键。在计算self-attention时，由于query所有位置对应的query向量是一样的，因此不管的query位置如何，对不同单词的attention偏差应保持相同。
3. $\mathbf{W}_{k,E}$ 、 $\mathbf{W}_{k,R}$ 也是需要学习的参数，分别产生基于内容的key向量和基于位置的key向量。

最后再经过Masked-Softmax、Layer Normalization、Positionwise-Feed-Forward得到最终预测用的，详细的过程看论文[1]提供的**补充材料B**。