基于transformer-XL的XLNet是目前在BERT基础上改动较大的后起之秀。在超长文本的场景下，XLNet相比其他Bert系列的模型会有更好的性能(recurrent机制使其可捕获更长的上下文依赖关系)以及更快的训练与推理速度(memory单元中缓存了之前(一个或多个)段的隐状态信息，避免了重复计算)，且在一般长度的文本场景中会有更完整的语义信息(PLM考虑了被mask的token间的联系)。以上分析都是基于paper中的理论。实际上，当有足够多的数据时，Bert系列的各个版本在大部分场景(超长文本的场景比较少，显现不出XLNet的优势)下的效果差别不大。

为了更好的理解XLNet，首先对其采用的特征抽取器transformer-XL进行解读，然后再逐步介绍XLNet内部的各个组件。

# Transformer-XL

**背景**

在对语言建模时，针对如何提升编码器捕获长距离依赖关系能力的问题，有几种比较有效的编码器。LSTM为了建模长距离依赖，利用门控机制和梯度裁剪，有paper验证目前可编码的最长平均距离在200左右。Transformer利用self-attention机制，允许词之间直接建立联系，能更好地捕获长距离依赖，其编码能力超过了LSTM，但局限于固定长度的上下文。

**概览**

Transformer编码固定长度的上下文。具体地，将一个长文本序列截断为几个固定长度的片段(segment)，然后分别编码每个片段，片段间没有任何的信息交互(如BERT的预训练模型中序列长度的极限为512)。如下图所示

图表, 散点图

描述已自动生成

上述编码策略有如下弊端：

* 对超过固定长度的依赖关系无法建模编码
* 对长文本的分割破坏了语义边界，导致上下文碎片化(context fragmentation)

为了克服上述弊端，有效建模长距离依赖关系，就有了transformer-XL (XL = eXtra Long)，其与传统(Vanilla)的Transformer相比，有如下两个特点：

* 片段级的递归机制(Segment-Level Recurrence with State Reuse)，引入memory模块(cache之前一个或多个segment的隐状态信息)，循环建模片段间的联系
* 使超长距离依赖关系的编码成为可能
* 使得片段之间产生交互，解决了上下文碎片化问题
* 相对位置编码(Relative Positional Encodings)，代替绝对位置编码
* 避免了memory中缓存的片段的位置信息与当前片段中的位置信息相互混淆

具体示意图如下：

图表, 散点图, 漏斗图

描述已自动生成

**Segment-Level Recurrence with State Reuse**

为了解决transformer模型使用固定上下文的限制，transformer-xl引入了循环机制。具体地，在训练阶段，当模型处理下一个新段(segment)时，前一段序列中的隐状态信息被固定并缓存，作为扩展上下文重用，如图2a所示。尽管梯度只保持在一个独立的段中，但额外的输入信息(扩展上下文)允许网络利用历史信息，从而能够对长期依赖关系进行建模并避免上下文碎片化(context fragmentation)。

记两个长为L的连续段分别为和。

令第τ个segment中第n层生成的隐状态序列为，其中d表示隐层维度。则segment 中第n层的隐状态序列可按如下规则生成：

文本

描述已自动生成

其中SG(⋅)表示梯度不回传(stop-gradient，fixed and cached)，表示第τ+1个segment的第n−1层扩展上下文后的隐状态序列，其由第τ个segment的第n−1层隐状态与第τ+1个segment的第n−1层隐状态在长度(L)的维度进行拼接后得到。为模型待学习的参数。

与标准的transformer相比，关键区别在于与以扩展上下文的隐状态序列为条件，其中缓存了上一个segment的隐状态序列。图2a中用绿色路径表示依赖的上下文来自memory单元中缓存的隐状态信息。

通过这种递归机制应用于语料库的两个连续片段，就会在隐状态下创建segment级的递归。因此，可利用的有效上下文可能远不止两个segment。然而，在与之间的循环依赖关系不同于传统的RNN-LMs中的同层递归，这里在每个segment中向上移动一层。因此最大可能的依赖长度随着层数和segment的长度线性增长，即O(N×L)。如图2b中的阴影区域所示，**作者们在设计时限制了跨segment时不同层间的token最多可依赖的token个数为segment的长度，理论上也可以不限制**。

除了获得超长的上下文并解决了碎片问题，递归方案的另一个好处是评估速度明显加快，其可重复使用先前segment的表示，不必像图1中那样从头开始计算。

最后，作者们指出递归方案不必仅局限于前面介绍的细节。从理论上讲，我们可以缓存GPU显存允许的尽可能多的先前segment，并在处理当前segment时将所有的先前的segment重用为额外的上下文。因此，我们可以缓存一个预定义的长为M的之前的隐状态序列，其(可能)跨越多个segment，作为记忆单元。作者们在实际实验时，在训练期间令M为segment的长度，评估阶段为M的倍数。

**Relative Positional Encodings**

recurrent机制使得先前的绝对位置编码方案不再适用，因为在多个segment中会出现多个同样的位置信息。 为此，作者们提出一种新的相对位置编码形式。其不仅与绝对位置一一对应，而且具有更好的泛化性。

首先，在标准的transformer中，在同一个segment下的query向量和key向量的**注意力得分可分**解为：

图示

低可信度描述已自动生成

其中E表示词向量组成的矩阵，它是内容的承载者。U表示绝对位置向量组成的矩阵，它是绝对位置的承载者。W主要用于attention机制QK的转换。(a)表示纯基于内容间的寻址，(b)和(c)分别是i位置的内容和位置信息分别与j位置的位置和内容信息进行的寻址，(d)表示纯基于位置间的寻址。

基于仅依赖相对位置信息的思想，transformer-XL将上述**query与key的得分**改为如下形式：

图示

描述已自动生成

主要改进点如下：

* 将(b)和(d)项的绝对位置信息替换成相对位置信息。这在本质上反映了一个先验：只有相对距离才是主要的关注点。其中R是像transformer中那种不可学习的正弦编码矩阵。该种编码使得在一定长度的memory上训练的模型可以自动推广到评估期间要长好几倍的memory上。
* 将(c)和(d)项的分别替换成可学习的向量和。将替换成可学习的向量，表明对于所有的query位置对应的query(位置)向量是相同的。 即无论query位置如何，对不同词的注意偏差都保持一致。
* 将转换key的权重矩阵分成基于内容的key的权重矩阵和基于位置的key的权重矩阵。

改进后的四部分的含义解读如下：

* (a)表示基于内容的寻址，即没有考虑位置编码的原始分数
* (b)表示内容相关的位置偏差，即相对于当前内容的位置偏差
* (c)表示全局的内容偏置，从内容层面衡量键的重要性
* (d)表示全局的位置偏置，从相对位置层面衡量键的重要性

综合上述的递归机制和相对位置编码，就得到了最终的transformer-XL的架构。下面总结一个只有单attention head的N层transformer-XL的完整计算过程。对于n=1,⋯,N有

文本, 信件

描述已自动生成

其中表示段τ中所有词的embedding组成的序列。此外，直接按注意力得分A中的计算，其复杂度为O()。实际上i-j的取值范围为0∼L，可先计算好L个向量，然后在计算A时直接用就好，此时可将复杂度降为O(L)。

**结论**

**transformer-XL是一种强大的语言模型。其有较低的困惑度，与RNN和Transformer相比，可建模更长的依赖关系，在评估过程中实现了实质性的加速，并能够生成连贯的文章。**

XLNet: Generalized Autoregressive Pretraining for Language Understanding：

https://arxiv.org/pdf/1906.08237.pdf

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context：

https://arxiv.org/pdf/1901.02860.pdf

https://blog.csdn.net/linchuhai/article/details/100032082

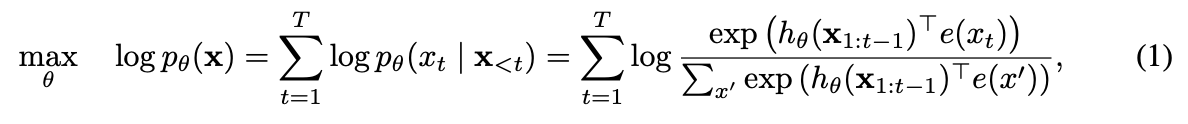
https://copyfuture.com/blogs-details/20210328224453170Y

# XLNet

与**基于自回归**的语言建模的预训练方法(GPT)相比，像BERT那种基于**降噪自编码**的语言建模的预训练方法可建模双向上下文信息。但是BERT通过使用mask破坏了输入，造成了**预训练与微调间的不一致，且忽略了mask间的依赖关系**。 基于自回归和自编码语言模型的优缺点，提出一种**广义的自回归**预训练方法XLNet。其主要**由排序语言模型(PLM)、双流自注意机制、(部分预测)Partial Prediction、相对segment编码以及transformer-XL中(基于memory单元)的segment循环机制和相对位置编码构成**。 整体设计很有说服力，理论性强。

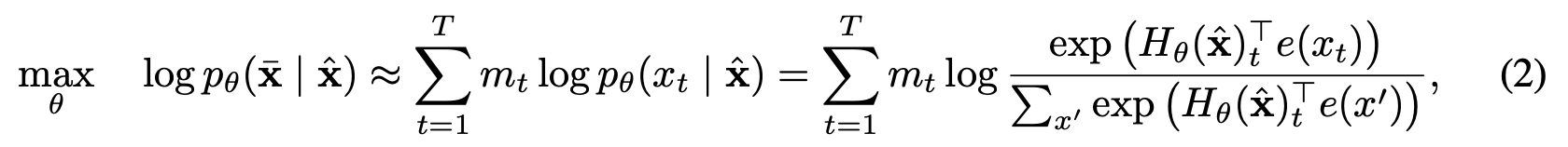
**AR与AE**

作者们认为，当前预训练最主要的两个目标可分为两类，一类是类似GPT的AR(AutoRegression，自回归) 方式。即根据前面所有信息预测后一个token，不断重复(自回归)。形式化的描述为：



其中为给定长度为T的文本序列，是由神经模型(如RNN或Transformer)产生的上下文表示，e(x)表示x的embedding。

另一类是像BERT的AE(AutoEncoder，自编码器)方式，做法类似DAE(Denoising AutoEncoder，去噪自编码器)，即把输入破坏掉一部分，然后还原。BERT中的具体做法是随机将一些token替换成mask(一部分是MASK,一部分是来自词表，一部分保持不变)，然后预测被mask掉的token。形式化的描述为：



其中表示x经过mask后的文本序列，即带噪输入。表示给定的文本序列x中所有被mask的token的集合，表明token 被mask，表示将一个长为T的文本序列x映射成一个隐向量序列*。*

从下面几个方面说明两种预训练目标的优缺点：

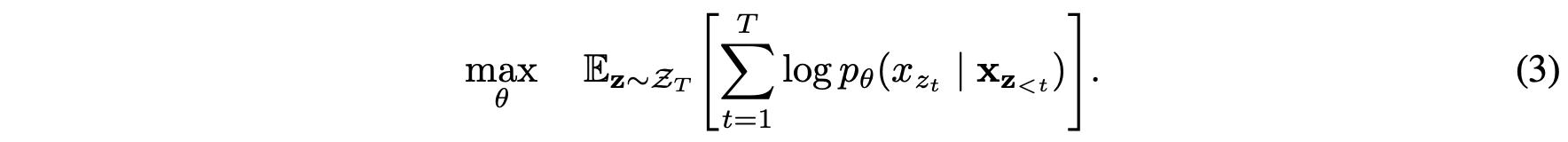
* 独立假设。BERT基于所有被mask的token间是相互独立的假设对联合条件概率进行因式分解的。式(2)中的≈强调了这里有独立假设导致等号不成立。而式(1)中则没有这种假设。
* 输入噪声。式(2)引入了mask，使得预训练与下游任务不一致。而式(1)中则没有这种输入噪声。(**此处虽然在原始输入中没有直接引入噪声，但是在内部处理的时候会用到掩码矩阵进行token的预测**)
* 上下文的依赖。式(1)只能依赖当前位置左边的token，而(2)可同时依赖左右两边的token。 **这使得bert在NLU方面的性能要好于GPT**

**排列语言模型(Permutation Language Model)**

通过上面的比较可知，AR和AE两种方式在语言建模时各有各的优缺点。作者们提出的排列语言模型，既避免了两者的缺点又兼具了两者的优点。其是一种**广义的AR**方式，既保留了AR模型的优点，同时允许模型捕获双向上下文。

具体地，对于长度为T的文本序列x，共有T!种不同的排列顺序，每种不同的排列顺序执行一个有效的自回归式的因子分解。**如果模型参数在所有的排列顺序共享，那么模型可学习到双向信息。**

形式化地，令表示长度为T的文本序列所有可能排列的集合。和分别表示某个排列中的第t个元素和前t−1个元素。则排列语言模型的目标函数为：



对于一个文本序列x，每次采样一个因子分解顺序(factorization order)的排列z并根据得到的排列后的文本序列计算似然。**由于训练期间所有的排列共享一个模型参数**θ**，因此可看到文本序列**x**中所有可能的元素。** 即PLM具有捕获双向上下文的能力。此外，由于目标函数基于AR框架，因此PLM不存在独立假设和预训练与微调不一致的弊端。

**备注：** PLM只对因子分解顺序进行排列，而不是对初始输入序列进行排序。具体地，**保持原始输入序列的顺序不变，使用与原输入序列对应的位置编码，并依靠适当的attention mask实现因子分解顺序的排列。** 这保证了预训练与微调的一致性(输入的都是具有自然顺序的文本序列)。

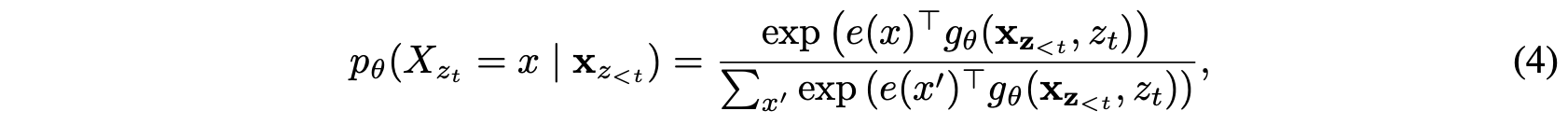
为了说明总体情况，给出如下图示。展示了在给定相同输入的文本序列x但对应不同的因子分解顺序下，预测token 。左边的mem表示transformer-XL中的memory单元。

图示

描述已自动生成

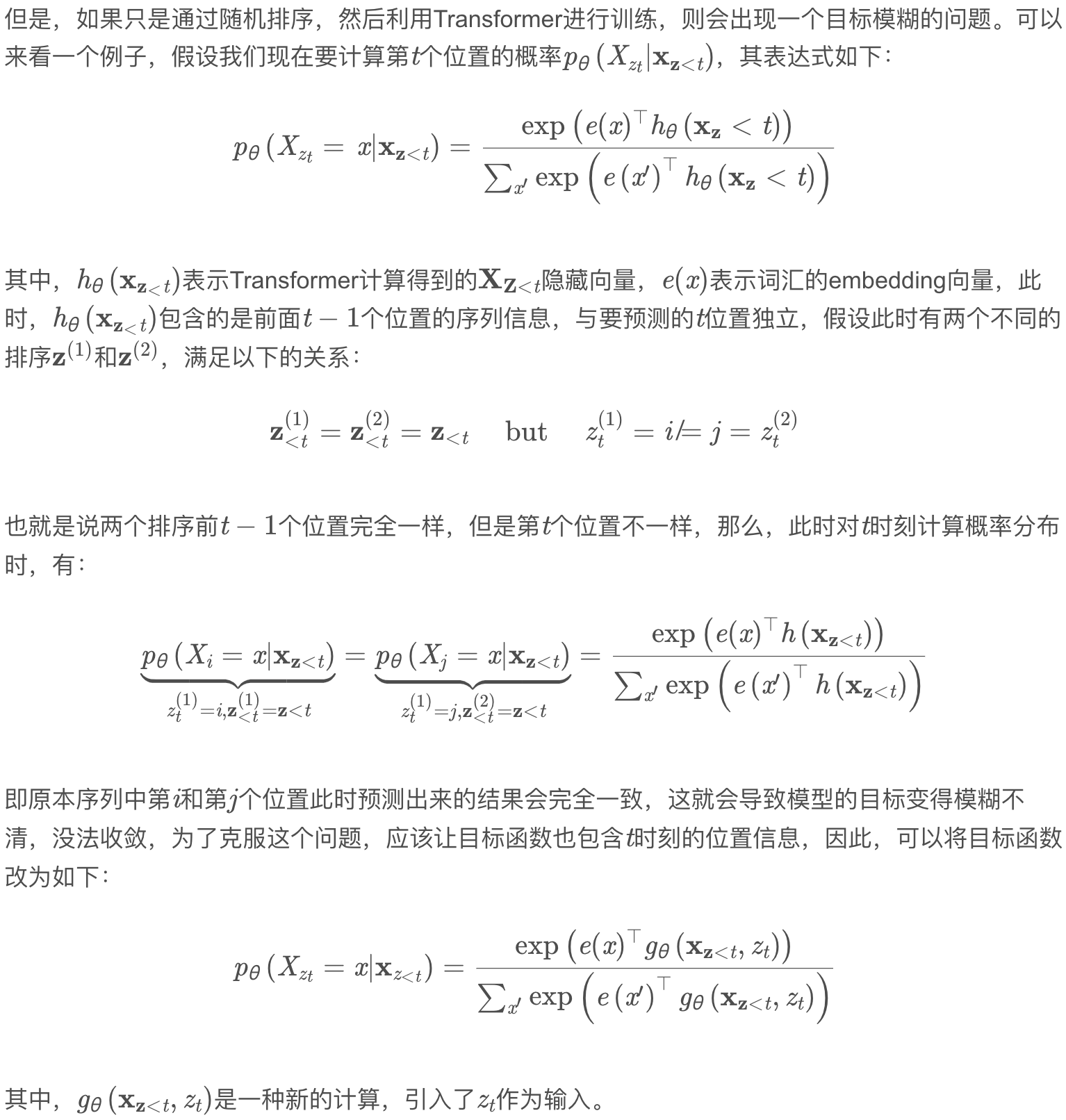
**Two-Stream Self-Attention for Target-Aware Representations**

虽然上述提出的PLM的目标函数具有很好的特性(充分利用双向上下文信息且不(显示地)引入外部噪声)，但用传统的transformer计算(3)式中的无法work。因为在未引入排列机制前，每个输入序列的顺序是确定的。**而引入排列机制后，同样的序列(目标token之前的序列)后要预测的token可能不同**，如果还是用(1)式那种经典的AR方式计算下一个token的分布情况，**会导致不同的token却有相同的分布**。 为了避免上述问题，PLM在预测下一个token的分布时将目标token的位置也考虑进来。即



其中表示一种附加了目标位置作为输入的新型表示。

以下灰色部分为具体地为什么要引入g的相关举例：



**Two-Stream Self-Attention**

虽然解决了预测目标的歧义，但如何定义仍是个不小的问题。而在传统的transformer结构中有两个相互矛盾的要求：(1)在预测token 时，只能利用位置信息和上文信息，不能利用内容信息。 (2)在预测其他token时，如，其中j>t，又希望能将内容信息也编码进来，以提供完整的上下文信息。为了解决上述矛盾，提出**双流机制**(**用两种隐状态表示而不是像传统transformer中那样只有一种**)。

* Content representation *，*简写为*。*与传统transformer中的隐状态相似，其可编码的上(下)文和自身。图示说明如下：

图示

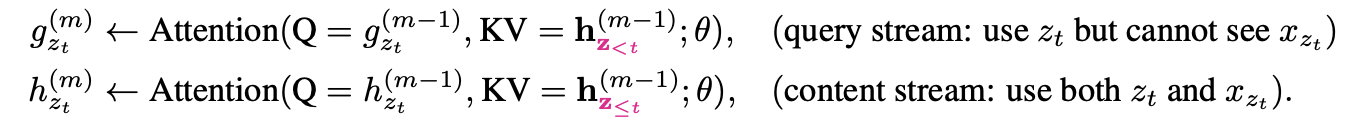
描述已自动生成

* Query representation，简写为。其可编码的上(下)文和的位置信息，不能编码的内容(content)。图示说明如下：

图示

描述已自动生成

在计算上，第一层的query stream被初始化为一个可学习的向量(*)*，而content stream设置为相应的词embedding()。对于每个自注意层m=1,⋯,M 通过使用一组共享的参数示意性地更新两个表示流，如下所示：



对应的图示说明如下图中的(b)和(c):

图示

描述已自动生成

其中，Q,K,V同Transformer中的query、key、value，content表示的更新规则与标准的self-attention完全相同，**在微调阶段，可简单地删除query stream，将content stream作为transformer(-XL)的输出**。作者在实际训练时采用的是Transformer-XL的结构。需要注意的是，这里两个stream其实参数都是共享的，只是两者的输入和mask稍微不一致，其实content stream本质只是其一个辅助计算作用，为了便于计算而已。在最后计算概率时，用最后一层的进行计算（式4）。

**Partial Prediction**

虽然式 (3)表示的PLM的目标函数有诸多好处，但是由于排列问题，该目标函数比较难优化，且在初步实验中会导致收敛缓慢。为了降低优化难度，作者们选择**仅预测**某个排列的序列的**最后几个token。**(有相对充分的context，可加快收敛，节省存储。**与transformer-XL中的half-loss有相似的意思，与BERT中只预测一个序列中的部分token类似**)

形式上，将某个因子分解顺序(factorization order)序列z分成两部分，一个非目标子序列和一个目标子序列，c为切分点。新的语言建模的目标函数为：

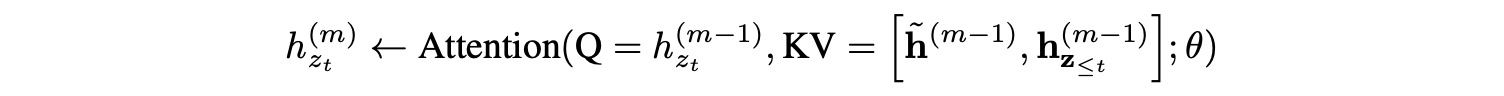
图表, 图示

描述已自动生成

关于选择多少个token作为预测目标，作者设置了一个超参数K，K等于序列总token个数除以需要预测的token个数，即。对于不需要预测的token，无需计算其query表示，从而节省了速度和内存。

**Incorporating Ideas from Transformer-XL**

由于上述分析的语言建模的目标函数适用于AR框架，因此作者们将最新的AR式的LM transformer-XL整合到预训练框架中。即将transformer-XL中的segment循环机制和相对位置编码集成到XLNet中。相对位置编码部分见transformer-XL中的介绍，不在赘述。segment循环部分主要利用memory中缓存的隐状态信息，从而可捕获更长的上下文依赖关系。不失一般性，假设一个长句s可以切分为两个子句，，和z分别表示[1…T]和[T+1…2T]的一种排序，基于排序，我们可以获得一个子句每一层m的content representations ，将其传递给下一个子句。对于content stream，其引入recurrent机制后的表示(content representations)如下：



**Modeling Multiple Segments**

许多下游任务具有多个输入段，如问答task中的问题和上下文段落。因此需要考虑如何在AR框架中预训练XLNet来建模多个segment。作者们仿照BERT，**在预训练阶段随机采样两个segment(要么来自相同的上下文要么不是)，并将其拼接为一个序列执行排序语言建模。在训练时，只重用属于同一个上下文的memory。** 具体地，模型的输入为 [A, SEP, B, SEP, CLS]，与BERT的输入相似。XLNet-Large在消融实验中发现NSP的task对性能提升不大。

**Relative Segment Encodings**

在结构上，不同于BERT在每个位置的token的embedding基础上增加绝对segment编码，作者们利用transformer-XL的相对编码思想来编码segment。

具体地，给定一个序列中的一对位置i和j，如果i和j在同一个segment中，那么segment编码，否则。其中与是每个attention head中可学习的模型参数。即在编码某个token的segment信息时，不考虑其来自哪个具体的segment，只考虑其与其他位置的token是否在同一个segment内。以上就是**对segment进行相对编码的思想**。当计算位置i与位置j的注意力权重时，会额外加入这一项。其中表示标准attention中的query向量，b是可学习的特定头部的偏差向量。

对segment进行相对编码，一方面提升了模型的泛化能力，另一方面使得对具有两个以上的输入segment进行微调成为可能，而使用绝对编码是不可能的。

**Discussion and Analysis**

**Comparison with BERT**

比较式(2)和式 (5)可知，BERT和XLNet都只预测一个序列中的部分token。对BERT来说，这是必须的，因为如果mask了所有的token，就不可能做出有意义的预测。此外，对BERT和XLNet而言，部分预测通过仅预测具有足够上下文的token来降低优化难度。然而**BERT中由于独立性假设无法对mask的token间的依赖关系进行建模，XLNet就没有这种缺陷。**

文本

描述已自动生成

**Comparison with Language Modeling**

标准的AR式的语言模型，只能建模单向的依赖，而XLNet通过排列机制，其可建模双向依赖，从而能够编码更全面的语义信息。

**Bridging the Gap Between Language Modeling and Pretraining**

语言建模是一个快速发展的研究领域。然而由于缺乏双向上下文的建模能力，语言建模和预训练间还存在着一定的差距。如果语言建模不能直接改善下游任务，那么语言建模是否有意义。XLNet通过泛化语言建模的形式弥补了两者的差距。因此，XLNet进一步证明了语言建模研究的重要性。此外，利用快速发展的语言模型进行预训练也成为可能。如XLNet集成了最新的语言模型transformer-XL。

**conclusions**

XLNet是一种广义的AR式的预训练方法，其利用PLM将AR和AE的优点结合起来。XLNet通过集成transformer-XL并设计双流注意机制，完美设配了AR式的目标函数。其在各项task中相比BERT都取得了一定的提升。

* XLNet利用随机排列的思想，使得模型可以考虑文本序列的双向信息，因此，比普遍的AR模型要更强。
* XLNet的目标函数采用的AR模型的目标函数，因此，不需要像BERT那样采用MASK字符，因此，避免了预训练和训练阶段的模型差异，也剔除了被MASK词汇之间互相独立的强假设。