[预训练语言模型专题] XLNet：公平一战！多项任务效果超越BERT

原创 管扬 朴素人工智能

来自专辑

预训练语言模型

本文为预训练语言模型专题的第**14**篇。

快速传送门

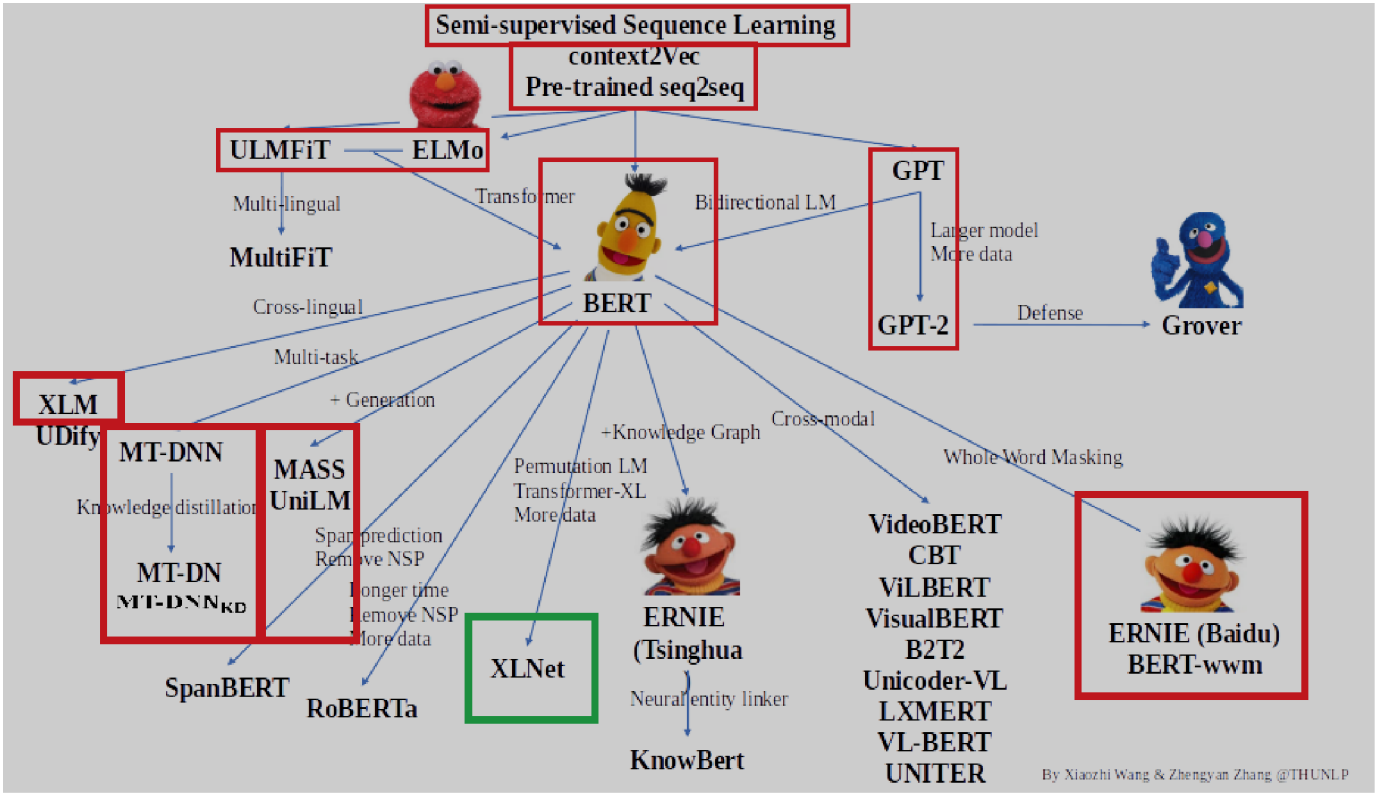
1-4:[萌芽时代]、[风起云涌]、[文本分类通用技巧] 、 [GPT家族]

5-8:[BERT来临]、[浅析BERT代码]、[ERNIE合集]、[MT-DNN(KD)]

9-12:[Transformer]、[Transformer-XL]、[UniLM]、[Mass-Bart]

13：[跨语种模型]

感谢清华大学自然语言处理实验室对**预训练语言模型架构**的梳理，我们将沿此脉络前行，探索预训练语言模型的前沿技术，红框中为已介绍的文章，绿框中为本期介绍的XLNet，欢迎大家留言讨论交流。



XLNet: Generalized Autoregressive Pretraining for Language Understanding（2019）

XLNet是由卡内基梅隆大学和Google AI Brain Team的作者共同发表的。文章摘要中提到，他们认为BERT由于在预训练时加入了[MASK]的token，导致pretrain 和 finetune 在训练数据上有差异，以致于降低了模型的效果。所以它们提出了XLNet，一种通用的自回归预训练模型，解决了这个问题，在包括问答，推断，极性分析，文档排序等20个自然语言任务上超过BERT，甚至在某些任务上超出了不少。

无监督的预训练在自然语言处理中已经取得了巨大的成功。典型的做法就是在大规模无标注数据上预训练，然后在下游任务上finetune得到模型或表示。在这个前提思想下，不同的预训练语言模型训练目标纷纷被提出。其中，autoregressive (AR) 自回归语言模型 和 autoencoding (AE) 自编码语言模型是最成功的两个。

- 自回归语言模型（AR）

  代表模型为**ELMo和GPT**，他们的语言模型任务是已知一段文本序列去建模后向或前向文本的概率分布。比如已知前t个文本的序列，来获得t位置文本的条件概率分布。它由回归分析发展而来，而这里预测的是该文本自己，所以被称为自回归。由这种定义，自回归语言模型是仅能建模单向的文本的概率分布，无法有效地深层双向的context。但众所周知，双向信息对预训练语言模型是很重要的，这也是自回归语言模型的一大问题。

- 自编码语言模型（AE）

代表模型为**BERT**，自编码语言模型的目的就不是去直接地估计下一段文本的条件密度，而是从被掩盖或残缺的文本中来重新构建原始的文本。像BERT，原始文本中的一定比例token会被mask掉，然后训练模型从被mask掉的文本中重现原来的文本。因为并非用条件概率密度估计作为目标，BERT就可以同时利用上下文的信息来重构原始文本，带来了极大的收益。但是BERT的一个问题是在预训练的时候，输入文本因为任务的原因，含有不少[MASK] token，但是在finetune的时候是没有的，导致了pretrain-finetune discrepancy。同时，BERT的mask策略导致它假设每个被预测（掩蔽的）词 在给定未屏蔽的 词 的情况下彼此独立，这在很多时间是不成立的，掩盖的词之前常常也会有相关联系。

对于这两种训练模式的优缺点，本文提出了XLNet，一种通用的自回归训练方法，权衡了两者的优点，回避了他们各自的不足，主要的思路是两点：

1. 相比于AR模型任务中用前向或者后向的极大似然来建模。XLNet对**所有可能的分解顺序排列**进行最大对数似然的优化。通过这种操作，每个位置能看到的context都可以包含左边和右边的tokens。每个位置能够学到所有位置的上下文信息，以此实现双向建模。
2. 作为AR语言模型的一种，XLNet**不依赖于数据重建**，所以不会像BERT一样有预训练和finetune的差别。同时，自训练目标很自然地利用乘法原则，获得所预测token的联合概率，从而避免了BERT中的被屏蔽词独立假设。

同时，XLNet还做了一些额外的改进：

- 基于AR语言模型的最新进展，XLNet在预训练中融入了Transformer-xl的片段循环机制及相对位置编码，提升了任务地效果，特别是对比较长的文本。
- 直接将Transformer或者Transformer-xl应用于重排列的语言模型效果不好，目标也比较模糊，所以XLNET重新调整了Transformer(-XL)使模糊性消除。

在可比较的实验设置上，XLNet在GLUE上多个自然语言理解数据集上，得到了比BERT更好的结果。

Objective: Permutation Language Modeling

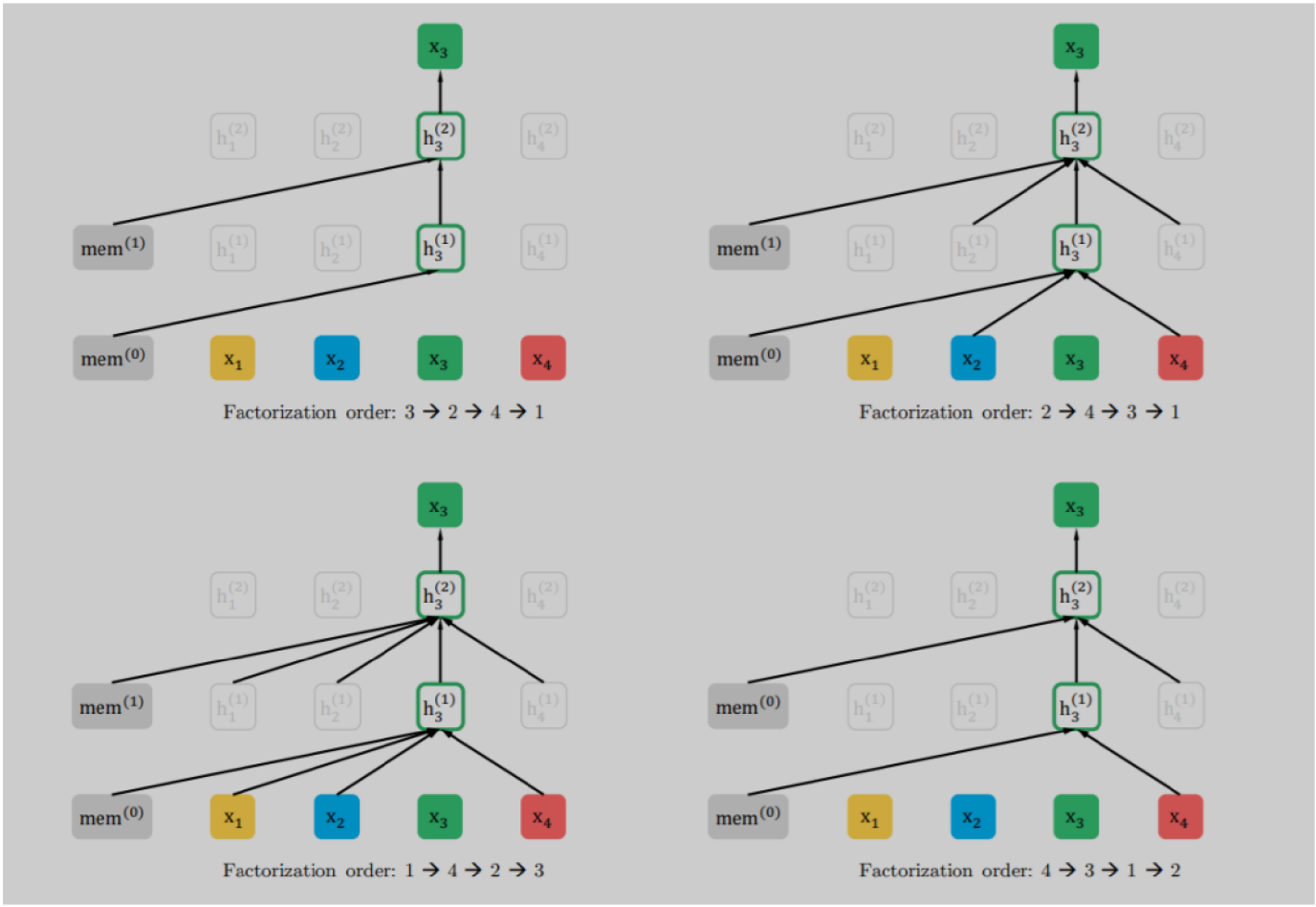
经过上面的比较，AR语言模型和BERT各有各的优点，如何取长补短得到更好的语言模型呢？XLNet借鉴和提出了一种混合排列的语言模型目标。比如说，对于一个长为T的序列x，它有T!种不同的排列方式。如果模型参数是所有顺序共享的，那模型就能从不同的方向接受序列的信息。

比如说，Zt是序列的一种排序，那么这种排序下的模型目标是

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T}\log p_\theta\left(x_{z_t} \mid \mathbf{x}_{\mathbf{z}<t}\right)\right].$$

一旦我们采样出一种排序方式Zt，就可以形成一个似然的优化目标。因为所有的顺序都可能出现，所以模型能够学到序列两个方向上所有的信息。同时，因为是AR框架的训练目标，它自然地避免了独立性假设和预训练和finetune时数据的不一致性。

实际上，本文提出的这种训练目标仅仅是改变了因式的顺序，而非序列的顺序。换句话说，作者保持了原来的序列顺序和位置编码，只是在Transformers中使用了合适的attention mask达成了因式的顺序。这种方式是必要的，因为在finetune时，模型只会根据序列的自然顺序进行编码。



比如在上图中，要预测x3，我们分别用四种不同的分解顺序，当顺序为3-2-4-1时，由于3出现在最前面，所以不能看到其他的字符，于是只通过前面循环存储来进行预测。当顺序为2-4-3-1时，由于3出现在第三位，所以第2和4个字符也可以参与进行x3的预测。同理，当顺序为1-4-2-3时，第1，2，3个字符都可以加入进行预测。

Architecture: Two-Stream Self-Attention for Target-Aware Representations

这种语言模型方式很好，但是在标准的Transformer上使用可能效果不佳，文中附录中举了一个例子。如果有两种不同的排列zt1和zt2，满足

$$\mathbf{z}^{(1)}_{<t} = \mathbf{z}^{(2)}_{<t} = \mathbf{z}_{<t} \quad \text{but} \quad z^{(1)}_t = i \neq j = z^{(2)}_t.$$

根据原本的极大似然计算方式，下面两个式子的结果应该是一样的，实际上，根据所预测target不同，其真实的分布必然应该是不同的。

$$\underbrace{p_\theta(X_i = x \mid \mathbf{x}_{\mathbf{z}_{<t}})}_{z^{(1)}_t=i,\, \mathbf{z}^{(1)}_{<t}=\mathbf{z}_{<t}} = \underbrace{p_\theta(X_j = x \mid \mathbf{x}_{\mathbf{z}_{<t}})}_{z^{(1)}_t=j,\, \mathbf{z}^{(2)}_{<t}=\mathbf{z}_{<t}} = \frac{\exp\left(e(x)^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}{\sum_{x'} \exp\left(e(x')^\top h(\mathbf{x}_{\mathbf{z}_{<t}})\right)}.$$

因此，为了解决这个问题，作者把预测下个词概率的公式，改成和目标的位置有关了，其中g函数将目标的位置也作为输入来进行计算了。

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{z_{<t}}) = \frac{\exp\left(e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}{\sum_{x'} \exp\left(e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)},$$

但是，在不作为预测目标的token，我们是应该了解其token内容的，于是为了在引入zt位置的同时保证大于t时隐层能得到正确的结果，作者提出使用两组隐层g, h表示来代替一组隐层h。h为content表示，使用上下文以及所预测字符编码而成。而g为query表示，只能了解target的位置zt，而不应获得具体的内容。

$$g^{(m)}_{z_t} \leftarrow \text{Attention}(\text{Q} = g^{(m-1)}_{z_t}, \text{KV} = \mathbf{h}^{(m-1)}_{\mathbf{z}_{<t}}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t})$$
$$h^{(m)}_{z_t} \leftarrow \text{Attention}(\text{Q} = h^{(m-1)}_{z_t}, \text{KV} = \mathbf{h}^{(m-1)}_{\mathbf{z}_{\le t}}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}).$$

换句话说，一个token X 将服务两种角色。当它被用作内容来预测其他标记时，我们可以使用content表示(通过内容流注意力来学习)来表示X。但是如果我们想要预测X，我们应该只知道它的位置而不是它的内容。

Incorporating Ideas from Transformer-XL

本文借鉴了不少Transformer-XL中的方法，而且以"XL"命名。这些方法主要包括相对位置编码和片段循环机制。有关内容可以参考我们关于Transformer-XL的推送。下图为XLNet中片段循环的公式。

$$h^{(m)}_{z_t} \leftarrow \text{Attention}(\text{Q} = h^{(m-1)}_{z_t}, \text{KV} = \left[\tilde{\mathbf{h}}^{(m-1)}, \mathbf{h}^{(m-1)}_{\mathbf{z}_{\le t}}\right]; \theta)$$

模型细节

最后，XLNet使用了很多数据来预训练，包括BooksCorpus，Engligh Wikipedia， Giga5，ClueWeb 2012-B，Common Crawl总共大概158G左右的文本，这个比BERT用的就多了不少。尺寸最大的XLNET是和BERT-large同一个大小，在512TPU v3 chips上训练了500k步，batchsize 8192， 大概训练了5.5天。同时，为了和BERT进行公平比较，作者也训练了XLNET在BooksCorpus，Engligh Wikipedia上的模型进行相同设置下的PK。结果如下：

Model	SQuAD1.1	SQuAD2.0	RACE	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
BERT-Large (Best of 3)	86.7/92.8	82.8/85.5	75.1	87.3	93.0	91.4	74.0	94.0	88.7	63.7	90.2
XLNet-Large-wikibooks	88.2/94.0	85.1/87.8	77.4	88.4	93.9	91.8	81.2	94.4	90.0	65.2	91.1

从结果中可以看出XLNet是好了不少的。但是，在Roberta出现以后，在不少结果上超过了XLNet初次发布的结果。让人们怀疑XLNet是否是靠着其使用了更多的数据而取得了优势，其本质上没有太大提升。

但是本文（那之后更新），又用与RoBERTa相同的设置和数据进行训练，做了一次公平的比较，结果如下图，总体来说，XLNet又超过了BERT和RoBERTa。

RACE	Accuracy	Middle	High	Model	NDCG@20	ERR@20
GPT [28]	59.0	62.9	57.4	DRMM [13]	24.3	13.8
BERT [25]	72.0	76.6	70.1	KNRM [8]	26.9	14.9
BERT+DCMN* [38]	74.1	79.5	71.8	Conv [8]	28.7	18.1
RoBERTa [21]	83.2	86.5	81.8	BERT†	30.53	18.67
XLNet	**85.4**	**88.6**	**84.0**	XLNet	**31.10**	**20.28**

再看在阅读理解数据集上结果，XLNet也领先于两者。

SQuAD2.0	EM	F1	SQuAD1.1	EM	F1
Dev set results (single model)					
BERT [10]	78.98	81.77	BERT† [10]	84.1	90.9
RoBERTa [21]	86.5	89.4	RoBERTa [21]	88.9	94.6
XLNet	**87.9**	**90.6**	XLNet	**89.7**	**95.1**
Test set results on leaderboard (single model, as of Dec 14, 2019)					
BERT [10]	80.005	83.061	BERT [10]	85.083	91.835
RoBERTa [21]	86.820	89.795	BERT* [10]	87.433	93.294
XLNet	**87.926**	**90.689**	XLNet	**89.898‡**	**95.080‡**

总的而言，XLNet使用了重排列的语言模型目标，结合了AR模型和AE模型的优点，同时XLNet继承了Transformer-XL的片段循环机制和相对编码特性，对文本长时序列进行了更好地处理，

在很多任务上超过了之前的模型。

未完待续

本期的论文就给大家分享到这里，感谢大家的阅读和支持，下期我们会给大家带来其他预训练语言模型的介绍，敬请大家期待！

欢迎关注朴素人工智能，这里有很多最新最热的论文阅读分享，有问题或建议可以在公众号下留言。

往期推荐

- 万字长文带你一览ICLR2020最新Transformers进展（上）
- [预训练语言模型专题] Transformer-XL 超长上下文注意力模型
- [预训练语言模型专题]跨语种语言模型
- [预训练语言模型专题] Huggingface简介及BERT代码浅析
- [预训练语言模型专题] 结合HuggingFace代码浅析Transformer