Google Research新成果，让表格理解和检索更上一层楼！

原创 舞风小兔 夕小瑶的卖萌屋 2021-09-28 12:05

文 | 舞风小兔
编 | 智商掉了一地

> 如何更好地理解自然语言查询问题与表格信息？Google Research给出了一个改进版Transformer，一起来看看吧！

表格以结构化方式存储信息，广泛地存在于web世界中。表格最为常见的一种用法就是人们查询其中的信息。在很多情况下，我们可能**只能够用自然语言描述出心中的查询条件**，那么，自然语言处理技术是否能**理解我们的问题**，**理解表格信息**，帮助我们**自动地从表格中检索答案**呢？那就让Transformer这个已然横扫各项自然语言处理任务的明星模型试试吧。

然而很快会发现，在Transformer 最为常见的使用方式中，模型总是被用来处理**长度为512个token的单层序列**。当面对行数非常多的“表格”这种半结构化数据时，**数据的结构**以及多行带来的**超长文本**，都在挑战经典Transformer对数据建模的能力和计算效率。

本文要介绍的模型 MATE 就是在上述问题的挑战之下，提出的一个Transformer改进模型。专门用来以更快的处理速度学习含有非常多行的表格型数据。

论文标题

MATE: Multi-view Attention for Table Transformer Efficiency

论文链接

https://arxiv.org/abs/2109.04312

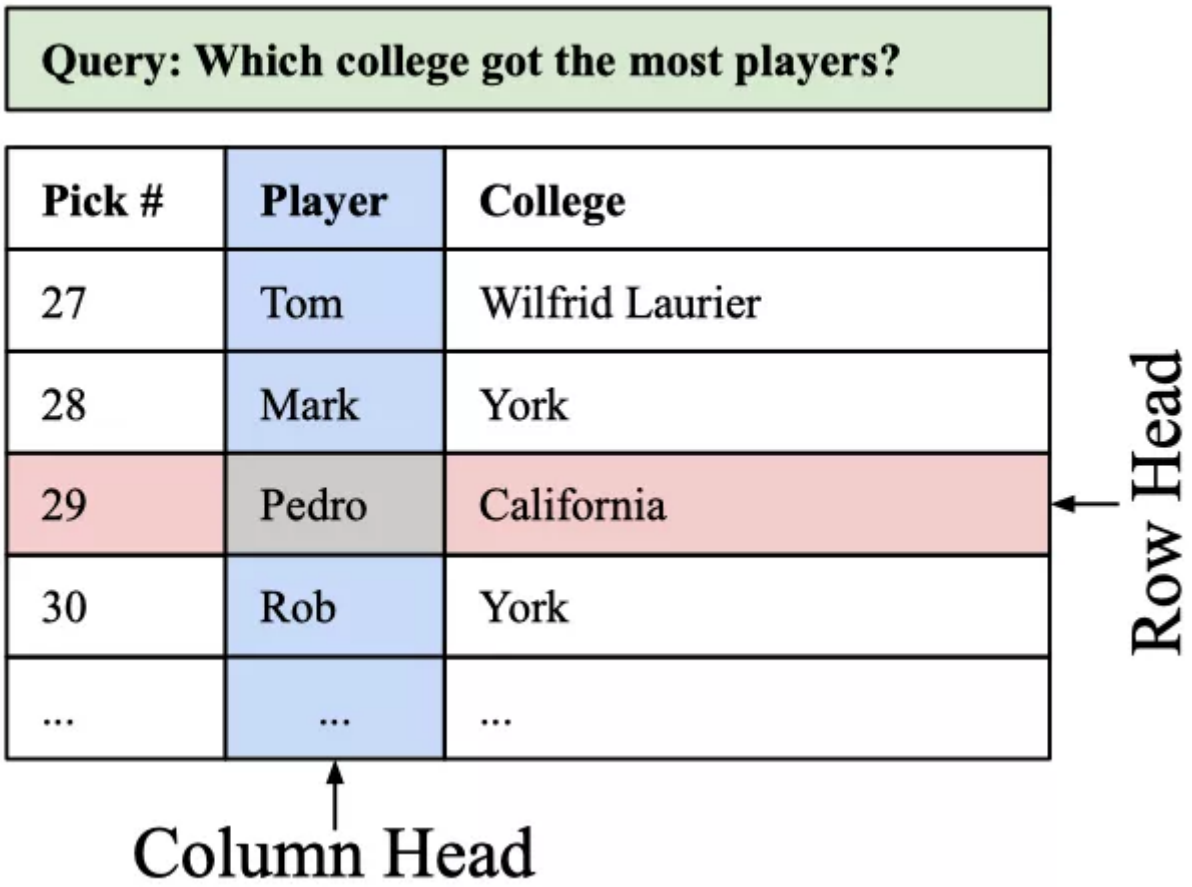
1 为什么需要 MATE

关系型表格（relational tables）是一种 **半结构化文本** (semi-structured text)，广泛存在于Web数据之中。在WebTables[1]项目统计的Web数据中，超过20%的关系型表格含有20甚至更

多行。每一行的每个单元格都有可能包含文本片段，如果将所有单元格中的文本拼接在一起，将会构成一个超长文本。

> *半结构化文本：是指具有结构的文本，但是这个结构并不反应某种已知的 data schema。例如组织成html，树，或者变长序列的文本，是常见的半结构化文本*。

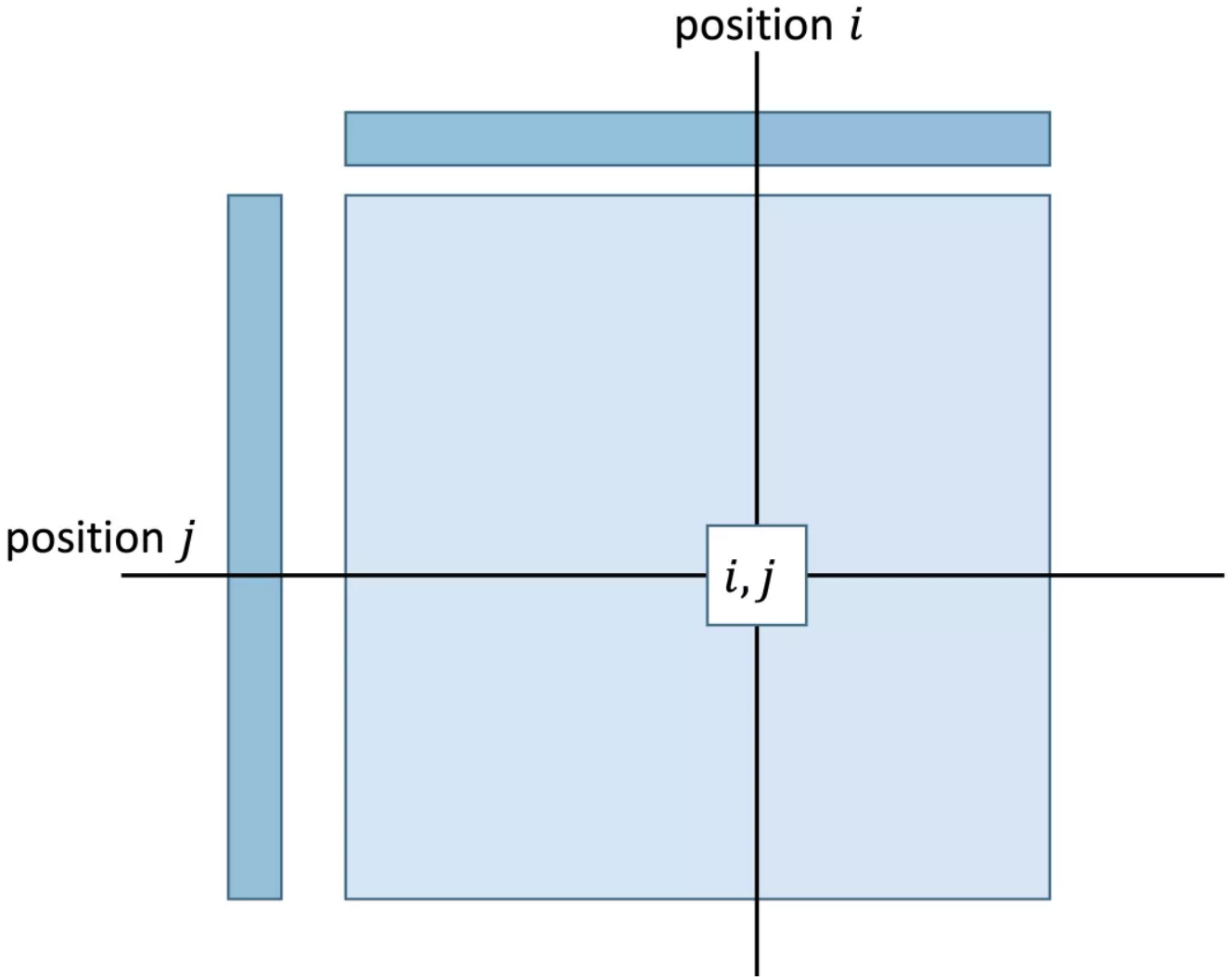
我们先通过图1这个表格问答任务，直观地理解一下MATE模型学习表格数据的设计动机。在表格问答任务中，给定问题的答案蕴含在表格某个单元格之内。当人们在处理信息时，会通过确定行和列，在行列相交处得到问题的答案。MATE就是通过对Transformer中注意力机制的改建，来模拟这一决策过程。



▲MATE模型的设计动机

图1. MATE模型的设计动机：MATE模型的稀疏自注意力模型包含两种注意力头（Attention heads）：Row Head对齐到（attends to）的tokens所处的单元格都位于一行。同样地，Column Head对齐到（attends to）的tokens所处的单元格都位于自同一列。问题对齐到除此之外的所有其他token。

在这里，我们不妨先想一想，在作者想要解决的问题中，表格会包含非常多的行数，形成了一个超长文本序列，如果直接应用Transformer中的经典注意力机制，会遇到什么样的问题呢?



▲“注意力”概念示意图

图2是注意力机制工作过程的一个概念示意图：两个序列中分别处于位置i和位置j的token计算一个相似度得分，这个得分表示了位置i处的token对齐（attends to）到位置j处token的强度。很容易看到，**注意力机制的计算和序列长度呈平方复杂度关系，当序列长度增长到千以上这样的量级时，将会消耗大量的内存和计算时间**。

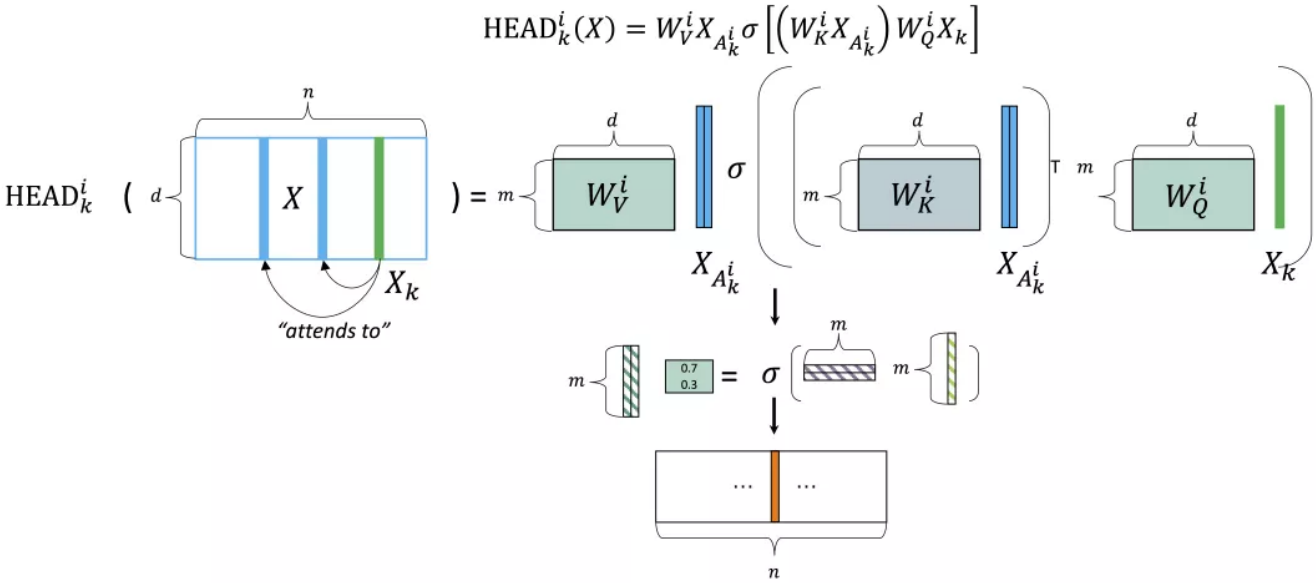
为了让注意力机制的计算在超长文本上更加可行和高效，MATE提出使用稀疏注意力模块，**目标是让模型的计算速度和内存消耗随序列长度增加而线性增加**，而不是像原始注意力机制那样，模型的计算速度和内存消耗随序列长度增加呈二次方增加。通过稀疏注意力机制，**MATE能够处理长达8000个token级别的输入数据**。

那么，我们来看看MATE是如何在表格数据上做到这一点的。

2 MATE 模型

MATE的核心稀疏注意力模块可以用位于图3中第一行的公式概括。其中$\mathbf{X}$是一个d（词向量/隐藏层的维度）行 n（序列的长度）列的矩阵，是Transformer层的输入张量；矩阵W_Q^i，W_K^i，

W_V^i 是三个映射矩阵，将输入的Query，Key， Values 分别映射到维度m；σ是softmax函数；HEAD_k^i是第i个注意力头在k位置的输出向量。

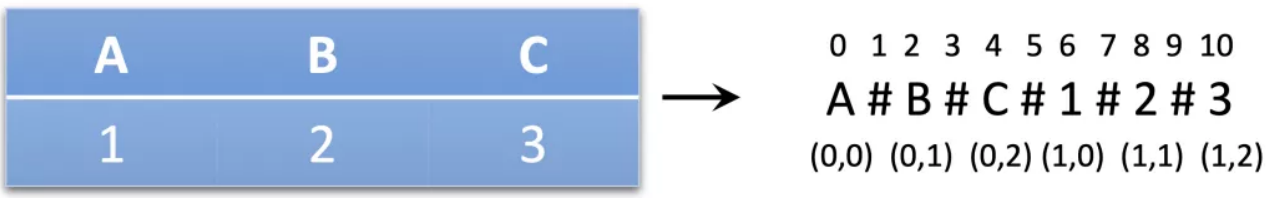


▲稀疏注意力机制的计算过程

相比图2中传统的稠密注意力，稀疏注意力**最重要的差别是引入了$A_k^i \subseteq \{1...n\}$这样一个称作注意力模式（attention pattern）的输入**。当指定了A_k^i时，**位置k处的token $\mathbf{X}_k$只会对齐到全序列中十分有限的几个位置**，以此来减少长序列情况下注意力模块的计算时间和对内存的消耗，也就是稀疏注意力这一名字的含义。

有了稀疏注意力机制之后，MATE又是如何处理有着行和列结构的表格型数据的呢？论文中的做法十分简单：

首先，结构化的表格构成输入数据时被拍平成一个序列，序列中token的序号k又进一步用两维的行序号r_k和列序号c_k表示。下图是这一个编号过程的概念示意图（*注意：图中没有完全精确地表示分隔词和填充如何构造这样的细节信息，只做编号的概念示意*）。



▲编号过程概念示意图

其次，MATE将多个注意力头划分为两组，前$h_r \geq 0$个注意力头是行注意力头，剩下h_c个为列注意力头。token的二维的位置编号会引导注意力头对输入数据进行排序分类，限制行注意力头只在同行的单元格之间计算注意力，列注意力头只在同列的单元格之间计算注意力。行注意力头和

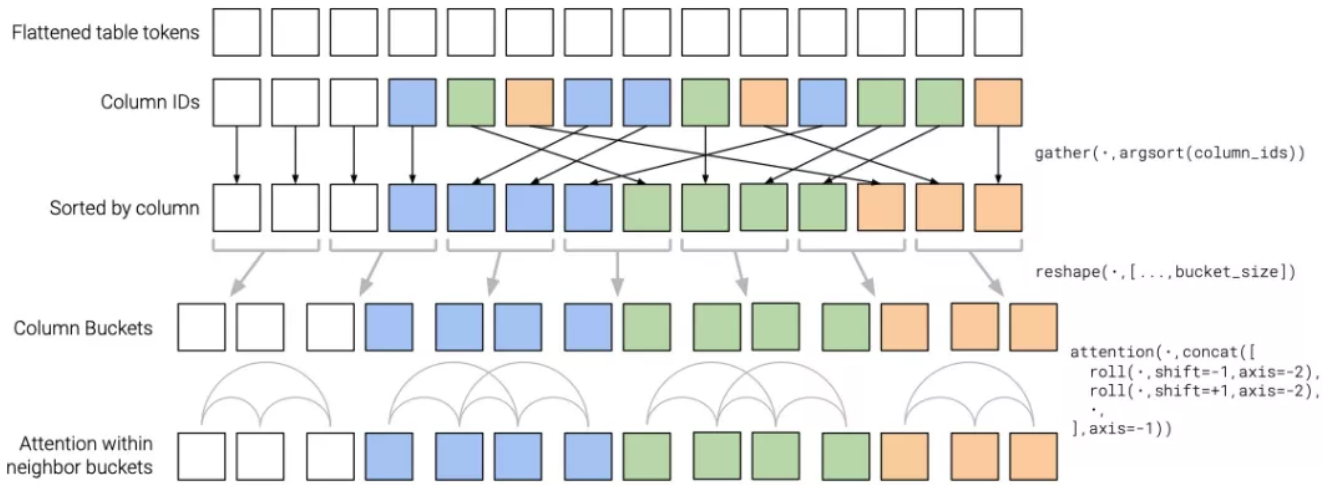
列注意力头的比例是一个超参数，作者在论文中的实验数据建议按照1：1比例划分行注意力头和列注意力头。

3 高效MATE的实现

有了上面的设计思想，一种最粗暴的实现MATE的方法是按照传统Transformer那样，通过masking消除不需要对齐位置的计算结果，但masking并不减少实际的计算，计算过程依然会按照稠密注意力计算方式，效率十分低下。另一方面，尽管MATE对注意力机制对齐的位置进行了一定的限制，能够改善模型的学习效果，但是并没有改善Transformer模型在计算速度方面的复杂度。

Transformer模型依然会消耗序列长度平方复杂度的内存和计算时间。为了提高模型的计算性能，受ETC[8]工作的启发，作者进一步提出了行/列注意力头的一种近似计算方法，具体来说分为2步：

1. 从输入序列中切分出为长度为 G 的global section：这部分token会与G中所有token逐一计算注意力分值；
2. 从输入序列中切分出 local section（local section的长度可以更长）：这部分token只会与global section计算注意力分值，或者与落在半径范围为R的token去计算注意力分值。



▲MATE 的高效实现

在进一步的注意力模块计算之前，列注意力头会根据token的列编号去排序输入token（对应了图 4中的第2行）。同理，行注意力头会根据token的行编号去排序token。当注意力头重排好输入数据之后，再将长度为G的global section 单独分为一组，计算稠密注意力；将剩下的输入进行分桶，每个桶内token长度相等（对应了图4中的最后一行）为R。每个桶内的token计算自注意力，两个相邻的桶之间也可以进行注意力机制的计算。

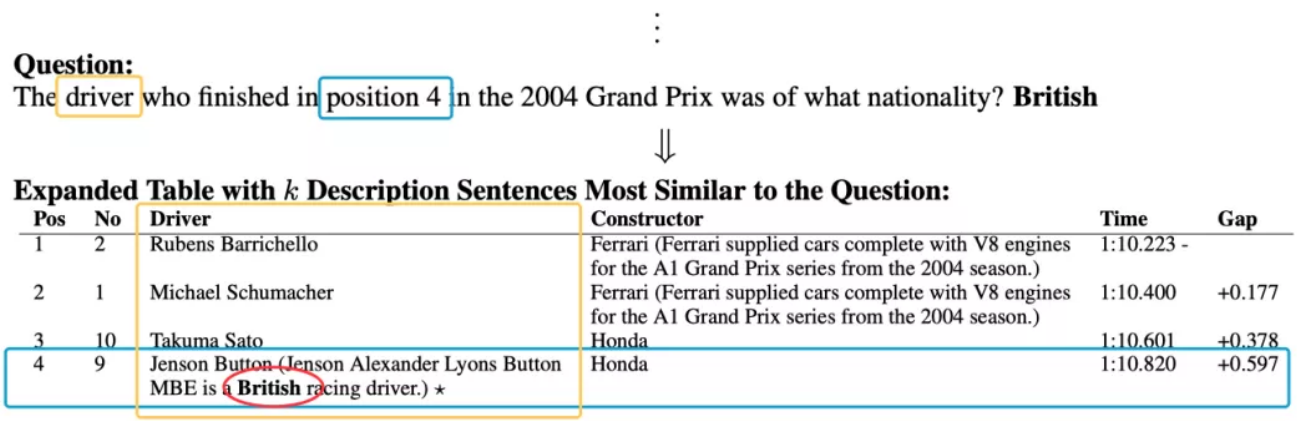
作者指出，当G的长度足以覆盖问题长度时，当R的长度足以覆盖单元格中文本的长度时，模型的计算效率会接近传统基于masked的Transformer。

4 PointR 结构

以上就是MATE模型的设计细节啦。有了这个基本算法单元，作者又进一步重点介绍了MATE如果应用于表格问答任务的处理流程设计。

4.1 含有长文本的表格问答任务

作者指出之前一些研究工作，在完成进行表格问答任务（table QA）任务时，使用的数据集token数目都被限制在512之内。而最近开源的一些数据集[3][4] 需要parse更长的半结构化文本。同样，我们先来看一个来自HybridQA[5]数据集的真实例子（图5）。HybridQA数据集中的表格抓取自维基百科。当某些单元格中的文本内容包含实体时，实体（entity）在维基百科中可能会有自己的超链接页面。数据集在构造时，对这些含有超链接的实体进行了展开处理。统计数字显示HybridQA数据集中，每个问题的表格平均有70个单元格和44个实体的链接，反应了这一数据集中文本的复杂程度。



▲结构化问答任务的一个示例

在表格问答任务中，模型接收的每个问题都会伴随一个表格，问题的答案可能跨越表格的某几个单元格，或者在某个单元格中跨越一个小的文本片段。HybridQA数据集本身并没有给出ground-truth答案如何抽取的标注信息，这就导致了数据集中大约会有50%有歧义的样本，这些样本中存在多个可能的成为回答的文本片段。如果希望保证90%的样本总是覆盖了潜在的答案，那么表格、问题和实体描述的总长度会增加到11000个token，这远远地超过了传统的Transformer能够处理的文本长度，也是MATE想要解决的问题。

为了将MATE应用到上面这样的超长文本结构化问答任务中，作者受到开放领域问答（open domain QA）pipeline设计的启发，进一步提出了一种两阶段处理框架：PointR。模型在**第一**

阶段指向（*Point*ing）正确的表格单元，第二阶段从找到的表格单元中读取（*R*eading）正确的答案。我们可以看到，这个处理过程再次十分自然地模拟了人处理表格信息的过程。

4.2 单元格选择（cell selection）阶段

单元格选择通过训练MATE模型，后接一个分类问题常用的交叉熵损失函数来完成，主要分为以下三步：

1. $S(t) = \mathrm{MLP}(\mathrm{MATE}(q, e)[t]$

   首先让MATE模型以“问题”和“拍平了的表格数据”这样一对输入通过自注意力机制学习表格文本的编码。之后，一个前馈网络（MLP）会对同一个单元格中的每个token t 进一步映射，得到单元格中每个token的logit得分。

2. $S(c) = \mathrm{avg}_{t \in c} S(t)$

   接着，对上一步计算出的单元格中每个token的logit得分求均值，输出单元格的整体logit得分。

3. $P(c) = \frac{\exp(\mathrm{S(c)})}{\sum_{c' \in x} \exp(S(c'))}$

   最后这一步是标准的softmax激活，之后就可以接上分类问题通用的交叉熵损失函数，这个交叉熵损失函数的指导下，模型去学习和预测一个单元格位置是否包含了答案。

在这里，作者讨论了单元格选择阶段会遇到一个十分现实的挑战。由于表格是输入信息中文本占比最多的一部分，如果MATE拍平整个表格作为输入数据，也就是展开表格中所有的单元格以及将单元格中的段落也进行展开，超长文本带来的计算量依然会让训练MATE模型去解决单元格选择问题变得十分不实用。

为了进一步减少计算量，作者在这里又做了一些简单的数据策略设计：通过计算实体描述（entity description）对问题（query）的TF-IDF指标，只选择TF-IDF前k的实体描述进行展开。

- 当$k = 5$时，能让97%样本的表格部分含有的token数落在2048之内。
- 对剩下3%的超长样本按照能承担的计算资源，硬性截断。

好啦，至此单元格选择阶段完成，开始进入处理的下一个阶段的处理：从找到的单元格中读取答案。

4.3 答案读取（passage reading）阶段

作者在答案读取阶段并没有做太多新的设计，直接利用neural machine reading中最通用的实践。原理和工作过程与上一步的单元格选择十分类似。作者Fine-tune了预训练的BERT-uncased-large模型[6]。BERT模型以“上一步找到的单元格”和“问题”这样一对数据为输入，计算出单元格每个token的编码。由于答案在单元格中通常会跨越多个token，文章里将答案第一个token的表示和最后一个token的表示拼接作为答案的表示，最后通过Softmax激活后接交叉熵损失函数，训练阶段引导模型学习和预测单元格中的一个片段是否为答案。

至此，两阶段的PointR结构完成，是这篇论文在MATE结构基础上提出的一个微创新设计。

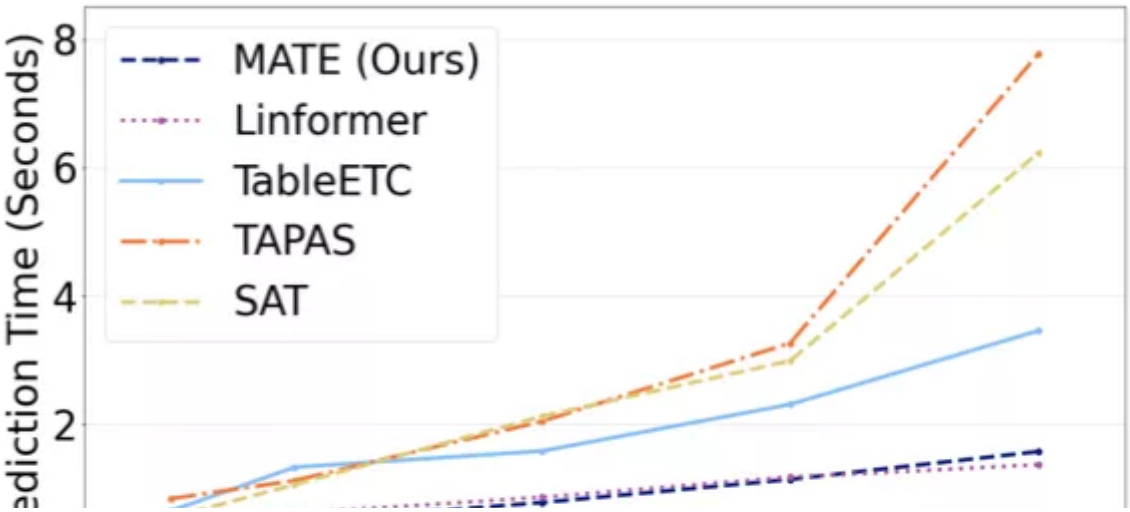
5 实验评估

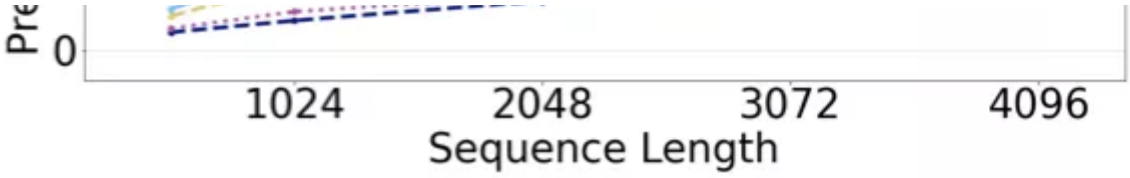
在实验评估中，作者主要关注两个方向：**表格型数据的学习效果以及模型速度**，与state-of-art方法相比，MATE都取得了显著的改善。学习效果方面，MATE将HybridQA数据集上的表格问答任务的最佳结果提高了19个点。

Model	SQA ALL	SQA SEQ	WIKITQ	TABFACT
TAPAS	67.2 ±0.5	40.4 ±0.9	42.6 ±0.8	76.3 ±0.2
MATE	**71.6** ±0.1	**46.4** ±0.3	**42.8** ±0.8	**77.0** ±0.3
TAPAS + CS	71.0 ±0.4	44.8 ±0.8	46.6 ±0.3	81.0 ±0.1
MATE + CS	**71.7** ±0.4	**46.1** ±0.4	**51.5** ±0.2	**81.4** ±0.1

▲MATE与TAPAS在表格parsing任务上的学习效果对比（来自论文中的Table 4）

预测速度方面，在作者实验的云上64GB的虚拟机上，当序列长度增长到2048时，MATE比TaPas[7]快了近2倍。





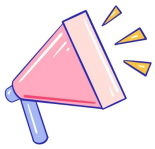
▲MATE与几种同类算法在预测速度上的对比（来自论文中的Figure 4）

关于作者给出的更多实验数据，有需要进一步了解可以参考原论文，这里将不再赘述。

6 小结

MATE 是google research 被 EMNLP 2021录用的一篇论文。文章介绍如何对经典作用于单层序列之上的Transformer模型进行改进使其在半结构化的表格数据上发挥作用。文章中有两个核心设计思想，能够为学习大规模结构化数据提供一些启示：

1. 将注意力头**针对结构化数据的结构特征进行分组**，不同分组关注不同的结构性输入，能够为模型的学习过程**引入更强的数据相关的结构性先验**。这是MATE在表格数据学习中与现有方法相比，能够取得更高学习性能的关键原因。MATE在学习时，将注意力头划分为行注意力头和列注意力头，行和列注意力头可以独立地配置所需的稀疏注意力策略，以及独立地去限制注意力机制关注的局部位置（locality），作者指出这是一个很大的数据策略空间，在具体应用中值得进一步依任务选择和调整。
2. MATE再次证明了，当Transformer应用长度超过千级别的超长序列时，使用稀疏注意力机制替代稠密注意力机制能够行之有效地改善学习和预测性能，MATE对稀疏注意力机制的优化，对有处理长序列需求的任务也是一个很好的参考。



参考文献

[1] Cafarella, Michael J., et al. "Uncovering the Relational Web". WebDB. 2008. (https://www.cs.columbia.edu/~ewu/files/papers/relweb-webdb08.pdf)

[3] Kardas, Marcin, et al. "Axcell: Automatic extraction of results from machine learning papers". arXiv preprint arXiv:2004.14356 (2020). (https://arxiv.org/pdf/2004.14356.pdf)

[4] Talmor, Alon, et al. "MultiModalQA: Complex Question Answering over Text, Tables and Images". arXiv preprint arXiv:2104.06039 (2021). (https://arxiv.org/pdf/2104.06039.pdf)

[5] Chen, Wenhu, et al. "Hybridqa: A dataset of multi-hop question answering over tabular and textual data". arXiv preprint arXiv:2004.07347 (2020). (https://arxiv.org/pdf/2004.07347.pdf)

[6] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". arXiv preprint arXiv:1810.04805 (2018). (https://arxiv.org/pdf/1902.10909.pdf)

[7] Herzig, Jonathan, et al. "TaPas: Weakly supervised table parsing via pre-training". arXiv preprint arXiv:2004.02349 (2020). (https://arxiv.org/pdf/2004.02349.pdf)

[8] Ainslie, Joshua, et al. "ETC: Encoding long and structured inputs in transformers". arXiv preprint arXiv:2004.08483 (2020). (https://arxiv.org/pdf/2004.08483.pdf)