Monotonic Chunkwise Attention

单调的分块注意

钟正秋和科林·拉斐尔谷歌大脑∗∗

加利福尼亚州山景城，94043，美国

{Chungcheng，craffel}@谷歌网站

摘要

Sequence-to-sequence models with soft attention have been successfully applied to a wide variety of problems, but their decoding process incurs a quadratic time and space cost and is inapplicable to real-time sequence transduction. To address these issues, we propose Monotonic Chunkwise Attention (MoChA), which adaptively splits the input sequence into small chunks over which soft attention is computed. We show that models utilizing MoChA can be trained efficiently with standard backpropagation while allowing online and linear-time decoding at test time. When applied to online speech recognition, we obtain state-of-theart results and match the performance of a model using an offline soft attention mechanism. In document summarization experiments where we do not expect monotonic alignments, we show significantly improved performance compared to a baseline monotonic attention-based model.

带软注意的序列间模型已经成功地应用于各种各样的问题，但是它们的译码过程需要二次的时间和空间开销，不适用于实时的序列转换。为了解决这些问题，我们提出了单调分块注意（MoChA），它自适应地将输入序列分割成小块，在这些小块上计算软注意。我们表明，使用MoChA的模型可以通过标准反向传播有效地训练，同时允许在测试时在线和线性时间解码。当应用于在线语音识别时，我们使用离线软注意机制来获得最新结果并匹配模型的性能。在不期望单调对齐的文档摘要实验中，与基于基线单调注意的模型相比，我们显示出显著的改进性能。

2017.12.14 arxiv

# 1简介

具有软注意机制的序列到序列模型（Sutskever et al.，2014；Cho et al.，2014）（Bahdanau et al.，2015）已成功应用于大量的序列转导问题（Luong et al.，2015；Xu et al.，2015；Chorowski et al.，2015；Wang et al.，2017；See et al.，2017）。在他们最熟悉的形式中，这些模型使用编码器递归神经网络（RNN）处理输入序列，以产生一系列隐藏状态，称为内存。解码器RNN然后自回归地产生输出序列。在每个输出时间步，解码器直接受到注意机制的调节，该机制允许解码器返回到编码器的隐藏状态序列中的条目。使用编码器的隐藏状态作为内存，使模型能够桥接长输入-输出时间滞后（Raffel&Ellis，2015），这与缺乏注意机制的序列到序列模型相比具有明显优势（Bahdanau et al.，2015）。此外，可视化模型在每个输出时间步处理的输入中的位置会产生一个输入-输出对齐，从而对模型的行为提供有价值的见解。

正如最初定义的那样，软注意在每个输出时间步检查内存的每个条目，有效地允许模型对任何任意输入序列条目设置条件。这种灵活性有一个明显的代价，即使用软注意机制的解码具有二次时间和空间代价O（TU），其中和分别是输入和输出序列长度。这就排除了它在非常长的序列上的使用，例如，总结非常长的文档。此外，由于软注意考虑到在每个输出时间步处理内存中的每个条目的可能性，因此它必须等到处理完输入序列之后才能产生输出。这使得它不适用于实时序列转换问题。Raffel et al.（2017）最近指出，当输入-输出对齐是单调的，即输入和输出序列中元素之间的对应关系不涉及重新排序时，这些问题可以得到缓解。这种特性存在于各种实际问题中，例如语音识别和合成，其中输入和输出共享自然的时间顺序（例如，参见图2）。在其他设置中，对齐只涉及局部重新排序，例如某些语言对的机器翻译（Birch et al.，2008）。*T型U型*

基于这一观察，Raffel et al.（2017）引入了一种注意机制，该机制明确实施了硬单调的输入-输出对齐，允许在线和线性时间解码。



∗

平等贡献。

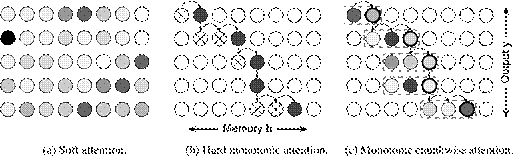


图1：本文讨论的注意机制示意图。每个节点表示模型在给定输出时间步（纵轴）处理给定内存条目（横轴）的可能性。（a） 在软注意中，模型在每个输出时间步为每个记忆条目分配一个概率（用每个节点的灰度阴影表示）。上下文向量被计算为内存的加权平均值，由这些概率加权。（b） 在测试时，单调注意从左到右检查记忆条目，选择是转到下一个记忆条目（用××表示）还是停止并参加（用黑色表示）。上下文向量被硬分配给所关注的内存条目。在下一个输出时间步中，它将从停止的位置重新开始。（c） MoChA利用一种硬单调的注意机制来选择它所关注的区块的端点（显示为带有粗体边框的节点）。块边界（这里，窗口大小为3）显示为虚线。然后，该模型对区块执行软注意（注意权重显示为灰色阴影），并计算上下文向量作为区块的加权平均值。

然而，与软注意相比，硬单调性约束也限制了模型的表达能力（软注意可以诱导任意的软对齐）。事实上，实验表明，利用这种单调注意机制的序列间模型的性能落后于标准的软注意。

在本文中，我们的目的是通过引入一种新的注意机制来弥补这一差距，该机制保留了硬单调注意的在线和线性时间优势，同时允许软对齐。我们的方法，我们称之为“单调的分块注意”（MoChA），允许模型在硬单调注意机制选择参与的记忆之前，对小块记忆进行软注意。它还有一个训练程序，允许直接应用于现有的序列到序列模型，并使用标准反向传播进行训练。实验结果表明，MoChA有效地缩小了在线语音识别中单调注意和软注意之间的差距，并在文档摘要（一项不显示单调对齐的任务）中提供了20%的相对改进。这些好处只引起参数数量和计算成本的适度增加。我们也提供了相关工作的讨论和未来的研究思路使用我们提出的机制。

# 2.摩卡咖啡

为了发展我们提出的注意机制，我们将首先回顾序列到序列的框架和最常见的软注意形式。因为MoChA可以被认为是单调注意的推广，所以我们重新推导了这种方法，并指出了它的一些缺点。从那里，我们展示了软注意块可以直接添加到硬单调注意，给我们的摩卡注意机制。我们还展示了如何根据机制的预期输出有效地训练MoChA，这允许我们使用标准的反向传播。

## 2.1序列间模型

序列到序列模型是将输入序列＝{x1，…，xT}转换为输出序列（可能具有不同的模态）＝{y1，…，yU}的模型。通常，输入序列首先由编码器递归神经网络（RNN）转换为隐藏状态序列={h1，…，hT}：**x y高**

*hj公司*=EncoderRNN（x，hjj−1）(1)

解码器RNN然后自回归地更新其隐藏状态，并且输出层（通常使用softmax非线性）产生输出序列：

|  |  |
| --- | --- |
| *硅*=DecoderRNN（yi−1，si−1，ci） | (2) |
| *彝语*=输出（si，ci） | (3) |

其中是解码器的状态，是作为编码器隐藏状态序列的函数计算的“上下文”向量。请注意，这是解码器访问有关输入序列的信息的唯一通道。*硅ci公司***小时***ci公司*

在最初提出的序列到序列框架（Sutskever等人，2014）中，上下文向量被简单地设置为最终编码器隐藏状态，即=hT。随后发现，这种方法在转换长序列时表现出性能下降（Bahdanau等人，2015）。取而代之的是，使用一种注意机制已经成为标准，这种机制将隐藏的状态序列视为一个（软）可寻址的内存，其条目用于计算上下文向量。在下面的小节中，我们将讨论三种这样的计算方法；否则，序列到序列框架将保持不变。*ci公司ci公司ci公司*

## 2.2标准软注意

目前，最常用的注意机制最初是在（Bahdanau等人，2015）中提出的。在每个输出时间步，此方法按以下步骤进行：首先，为每个内存条目生成一个未规范化的标量“能量”值：*我艾未未，j*

*艾未未，j*=能量（h，sji−1）(4)

一个常见的能源选择（·）是

                                                 能量（hj，si−1）：=v>tanh（Whhj+Wssi−1+b）（5）

其中∈R×dim（h），∈R×dim（s−1），∈Rand∈稀有可学习参数，是能量函数的隐维数。其次，这些能量标量在内存中使用softmax函数进行归一化，以产生权重值：*W型小时d级日本W型s码d级我b类d级五d级d级αi，j*

=softmax（ei，：）j（6）

最后，将上下文向量计算为的简单加权平均值，加权方式为：**小时***αi，*:

*T型*

*c级我*=Xα*i、 日本小时日本*(7)

*日本*=1

我们在图1a中看到了这种软注意机制。

注意，为了计算任何输出时间步，我们需要计算∈{1，…，T}的所有编码器隐藏状态。这意味着这种形式的注意不适用于在线/实时序列转换问题，因为它需要在产生任何输出之前观察整个输入序列。此外，生成每个上下文向量需要计算能量标量项和权重值。虽然这些操作通常可以并行化，但这仍然导致解码在时间和空间上具有O（TU）开销。*ci公司我hj公司日本ci公司T型*

## 2.3单调注意

为了用软注意解决上述问题，Raffel等人（2017）提出了一种硬单调注意机制，其注意过程可以描述为：在输出时间步，注意机制从它在前一个输出时间步关注的记忆索引开始检查记忆条目，称为。然后它计算=ti−1，ti−1+1，。。。并将这些能量值传递给logistic sigmoid函数（·），以产生“选择概率”。然后，从由参数化的Bernoulli随机变量中抽取离散的出席/不出席决策。到目前为止，我们总共*我钛*−1*艾未未，j日本*σ*皮，j齐，j皮，j*

|  |  |
| --- | --- |
| *艾未未，j*=单调能量（si−1，hj） | (8) |
| *第i、 日本*=σ（e）*i、 日本* | (9) |
| *齐，j*∼伯努利（pi，j） | (10) |

对于某些情况，一旦=1，模型就会停止并设置=j和=ht。这一过程如图1b所示。请注意，由于这种注意机制只在内存上进行单次传递，因此它具有O（max（T，U））（线性）代价。此外，为了处理存储器条目，编码器RNN只需要处理输入序列条目，这允许它用于在线序列转换。最后，请注意，如果∈{0,1}（一个鼓励的条件，如下所述），那么贪婪的赋值=hts相当于在可能的对齐路径上边缘化。*齐，j日本钛ci公司我hj公司十*1*，…，xj皮，jci公司我*

由于这个注意过程涉及到抽样和硬分配，所以利用硬单调注意的模型不能用反向传播进行训练。为了弥补这一点，Raffel et al.（2017）提出了通过计算注意力过程诱发的记忆概率分布来进行与期望值相关的训练。此分布采用以下形式：*ci公司*

（11） 然后将上下文向量计算为如等式（7）中所示的存储器的加权和。方程（11）可以通过观察（1−pi，j−1）αi，j−1/pi，j−1是在当前输出时间步（）处理内存条目−1的概率来解释（通过乘以（1−pi，j−1）并除以）。的添加表示模型在上一个输出时间步处理条目的额外可能性，最后将其全部乘以反映模型在当前输出时间步选择内存项的概率。请注意，这种递推关系不能在内存索引之间并行化（与softmax不同），但幸运的是，代入=αi，j/pi，j产生一阶线性差分方程=（1−pi，j−1）qi，j−1+αi−1，j，其具有以下解（Kelley&Peterson，2001）：*ci公司日本αi，j*−1*日本皮，j*−1*αi*−1，j*日本皮，j日本我日本齐，j齐，j*

*气，*:=cumprod（1−pi，：）cumsum（12） cumprod（1−pi，：）

这里是cumprod和cumsum。由于累积和和积可以并行计算（Ladner&Fischer，1980），因此使用这种方法仍然可以有效地训练模型。

注意，训练不再是在线或线性时间，但建议的解决方案是使用这种“软”单调注意进行训练，并在测试时使用硬单调注意过程。为了鼓励离散性，Raffel et al.（2017）使用了将零均值、单位方差高斯噪声添加到logistic sigmoid函数的激活中的常见方法，这使得模型学习有效地产生二进制。如果是二进制的，=1（pi，j>0.5），那么在实践中，为了简单的阈值化，避免在测试时采样。另外，观察到从softmax非线性到logistic sigmoid的切换由于饱和和偏移敏感而导致优化问题。为了缓解这种情况，使用了一个稍微修改的能量函数：*皮，j皮，j齐，j*

>

单调能量（（13）

其中是可学习的标量，如式（5）所示。（Raffel等人，2017）附录G中提供了对这些修改的进一步讨论。*g、 右v、 Ws、Wh、b*

## 2.4单调的分块注意

尽管硬单调注意提供了在线和线性时间解码，但是它对模型施加了两个重要的限制：第一，解码器在每个输出时间步只能处理内存中的一个条目，第二，输入-输出对齐必须是严格单调的。这些约束与标准的软注意形成对比，后者允许潜在的算法1 MoChA解码过程（测试时间）。在培训期间，第4-19行被替换为eqs。（20） 到（26），并在时间步−1处用接地真值输出替换。*彝语*−1*我*



1： 输入：内存的长度，块大小**小时***T型w型*

2： 状态：=~0，t0=1，=1，=StartOfSequence*s码*0 *我是的*0

3： 而6=内序do*彝语*−1*// 生成输出令牌，直到生成序列结束令牌*

4： for=ti−1待办事项*日本T型// 开始检查内存条目hj公司从我们离开的地方从左到右*

5： =单调能量（si−1，hj）*艾未未，j// 计算注意力能量hj公司*

6： =（ei，j）*皮，j*σ*// 计算选择概率hj公司*

7： 如果≥0.5，则为0.5*皮，j// 如果皮，j大于，我们停止扫描内存*

8： =j−w+1*五// 设置区块开始位置*

9： for=v做softmax*k公司日本// 按块计算超过一个尺寸的能量-w型前块日本*

10： =能量（si−1，hk）*用户界面，k*

11： 结束

*计算*softmax软件*-区块加权平均*

13： =j*钛// 还记得我们为下一个输出时间步停在哪里吗*

14： 中断*// 停止扫描内存*

15： 结束if

16： 结束

17： 如果0.5，∀j∈{ti−1，ti−1+1，…，T}，那么*皮，j<*

18: = ~0*ci公司// 如果我们不停地扫描整个内存，设置ci公司零向量*

19： 结束if

20： =DecoderRNN（si−1，yi−1，ci）*硅// 基于新的上下文向量更新输出RNN状态*

21:=输出（si，ci）softmax*彝语// 使用输出层*

22:=i+1*我*

23:结束时



|  |
| --- |
|  |
|  |  |

任意平滑的输入输出对齐。实验表明，在中测试的所有任务中，性能都有所下降（Raffel等人，2017）。我们的假设是，这种退化源于前面提到的硬单调注意所施加的限制。

为了解决这些问题，我们提出了一种新的注意机制，我们称之为MoChA，用于单调的分块注意。我们的想法的核心是允许注意机制在硬单调注意机制决定停止之前，在记忆的小“块”上执行软注意。这有助于在输入输出对齐中实现某种程度的软性，同时保留在线解码和线性时间复杂度的优点。

在测试时，我们遵循第2.3节的硬单调注意过程来确定（硬单调注意机制决定在输出时间步停止扫描内存的位置）。但是，我们不设置=ht，而是允许模型在内存条目的长度窗口上执行软注意，包括：*钛我ci公司我w型钛*

|  |  |
| --- | --- |
| *五*=钛-w+1 | (14) |
| *用户界面，k*=能量（si−1，hk），k∈{v，v+1，…，ti} | (15) |

(16)

其中，能量（·）是一个类似于式（5）的能量函数，它不同于单调能量（·）函数。摩卡的注意力过程如图1c所示。注意，摩卡允许非单调排列；特别是，它允许记忆条目的重新排序。在块上包含软注意只会增加运行时的复杂度，而且解码仍然可以在线进行。此外，使用摩卡只会引起参数总数的适度增加（对应于添加第二注意能量函数ChunkEnergy（·））。例如，在第3.1节描述的语音识别实验中，模型参数的总数只增加了大约1%。最后，我们指出设置=1可以恢复单调的注意力。为了完整性，我们在算法1中完整地展示了MoChA的解码算法。*高压，…，高压我w型w型*

在训练过程中，我们以与单调注意相似的方式进行，即基于MoChA的诱导概率分布（我们表示）使用期望值训练模型。这可以计算为*ci公司βi，j*

!

(17)

“总和”反映了单调注意可能停止扫描记忆的位置，以贡献概率，总和内的项表示块上的softmax概率分布，由单调注意概率缩放。由于嵌套求和，以这种方式计算每一个都是昂贵的。幸运的是，有一种有效的方法来并行计算∈{1，…，T}：首先，对于序列={x1，…，xT}，我们定义*k公司βi，jαi，kβi，jβi，j日本***十**

*不*+f−1级

                                                                MovingSum（x）n:=（18）*、b、f*十*xm公司*

*米*=n−（b−1）

例如，通过卷积长度为1s的-（f+b−1）序列并适当截断，可以有效地计算该函数。现在，我们可以高效地计算**十***βi，*:

*βi，*:=exp（ui，：）移动sum(19)

移动sum（exp（ui，：），w，1）

|  |  |
| --- | --- |
| 将所有这些放在一起会产生以下算法，以便在训练期间进行计算：*ci公司* |  |
| *艾未未，j*=单调能量（si−1，hj） | (20) |

(21)

(22)

*αi，*:=π，：cumprod（1−π，：）cumsum（23）cumprod（1−pi，：）

*乌伊，j*=能量（si−1，hj）(24)

*βi，*:=exp（ui，：）移动sum(25)

移动sum（exp（ui，：），w，1）

*T型*

*c级我*=Xβ*i、 日本小时日本*(26)

*日本*=1

方程（20）到（23）反映了（不变的）单调注意概率分布的计算。（24）和（25）计算MoChA的概率分布，最后等式（26）计算上下文向量的期望值。总之，我们开发了一种新的注意机制，它允许在小块内存上计算软注意，这些内存的位置是自适应设置的。该机制具有高效的训练时间算法，在测试时具有在线和线性时间译码功能。我们试图用附录B中的综合基准来量化与软注意相比所产生的加速。*ci公司*

# 3个实验

为了测试MoChA，我们将其应用于两个典型的序列转换任务：在线语音识别和文档摘要。语音识别对于MoChA来说是一个很有前途的设置，因为它会导致自然单调的输入输出对齐，并且在现实世界中经常需要在线解码。另一方面，文档摘要并不表现出单调的一致性，我们主要将其作为测试模型局限性的一种方法。我们强调，在所有的实验中，我们采用了一个强基线序列来对标准的软注意模型进行排序，只改变了注意机制；所有的超参数、模型结构、训练方法等都保持完全相同。这使我们能够隔离切换到摩卡咖啡所造成的有效性能差异。当然，这可能是对摩卡最佳性能的人为低估，因为它可能受益于稍微不同的超参数设置。我们把最好的表现留给以后的工作。

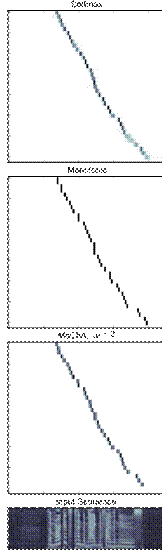


图2：语音识别任务的注意对齐图和语音特征序列。

具体来说，对于摩卡咖啡，我们使用公式（13）来表示单调能量和块能量函数。以下√



（Raffel et al.，2017），我们初始化=1/d（作为注意能量函数隐藏维度）并根据验证集性能调整初始值，使用=−4表示语音识别中的MoChA，使用=0表示摘要中的MoChA，使用=−1表示摘要中的单调注意基线。我们同样调整了块大小：对于语音识别，我们惊讶地发现所有∈{2,3,4,6,8}的性能都相当，因此选择了最小的值=2。总结一下，我们发现=8最有效。我们从经验上证明，即使这些小窗口的大小也能显著提高硬单调的注意力（=1），同时只会产生很小的计算代价。在所有的实验中，我们在验证集的最佳性能的训练步骤中报告测试集的度量。*克d级右右右右w型w型w型w型w型*

## 3.1在线语音识别

首先，我们将摩卡应用于它的自然环境，即我们期望大致单调对齐的领域：[1]华尔街日报（WSJ）语料库上的在线语音识别（Paul&Baker，1992）。这项任务的目标是在录音讲话中产生所说单词的序列。在此设置中，基于RNN的模型必须是单向的，以满足在线需求。我们使用（Raffel et al.，2017）的模型，该模型本身基于（Zhang et al.，2016）的模型。完整的模型和训练细节在附录A.1中提供，但是作为一个概括性的概述，网络将口语作为mel滤波器组谱图接收，该谱图被传递到由卷积层、卷积LSTM层和单向LSTM层组成的编码器。解码器是一个单一的单向LSTM，它通过MoChA或标准的软注意机制来处理编码器的状态序列。解码器在字符和字分隔符令牌上产生一系列分布。根据生成的单词分隔符标记，将模型输出的字符分割成单词后，根据单词错误率（WER）来衡量性能。我们报告的模型中没有一个集成了单独的语言模型。

我们在表1中显示了我们的实验结果，以及通过先前工作获得的结果。摩卡咖啡能够以很大的优势（20%的相对优势）击败最先进的咖啡。因为摩卡咖啡的表现和软注意基线非常接近，我们对这两种注意机制进行了8次重复试验，得出了最好的结果，

这些试验中单词错误率的平均值和标准差。我们发现基于摩卡咖啡的模型在试验中的方差稍高，这导致它的最佳WER较低，但与软注意相比，平均WER略高（尽管在未配对的学生t检验中，平均值=8的差异没有统计学意义）。据我们所知，这是第一次在线注意机制与标准（离线）软注意的表现相匹配。为了了解不同注意机制的行为，我们以图2中的WSJ验证集为例展示了注意对齐。正如预期的那样，所有注意机制的排列看起来大致相同。我们特别注意到，摩卡咖啡确实在利用这个机会，在每个长度为2的区块上产生一个柔和的注意力分布。*不*

由于我们在经验上发现=2的小值足以实现这些收益，我们进行了一些额外的实验，以确认它们确实可以归因于摩卡。首先是*w型*

|  |  |  |  |
| --- | --- | --- | --- |
| 先前结果 | | | WER公司 |
| （Raffel等人，2017年）（CTC基线） | | | 33.4% |
| （Luo等人，2016）（强化学习） | | | 27.0% |
| （Wang等人，2016）（CTC） | | | 22.7% |
| （Raffel等人，2017年）（单调关注） | | | 17.4% |
| 注意机制 | 最佳WER | 平均功率 | |
| 软注意（离线） | 14.2% | 14.6 ± 0.3% | |
| 摩卡，=2*w型* | 13.9% | 15.0 ± 0.6% | |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| 机制 | R-1型 | R-2型 |
| 软注意（离线） | 39.11 | 15.76 |
| 硬单调注意 | 31.14 | 11.16 |
| 摩卡，=8*w型* | 35.46 | 13.55 |

表2:CNN/每日文档摘要的ROUGE F分数

邮件数据集。软注意基线

表1：华尔街日报测试集的单词错误率。我们的结果（底部）反映了8个试验的统计数据。是我们对（见等人。，2017).

使用第二个独立的注意力能量函数ChunkEnergy（·）会导致参数计数的适度增加——在我们的语音识别模型中约为1%。为了确保性能的提高不是由于这个参数的增加，我们还用一个具有双重隐藏维度的能量函数重新训练了单调注意基线（以自然的方式产生了参数数量的可比增加）。在8个试验中，与基线相比，表现上的差异（下降0.3%的WER）并不显著，与摩卡咖啡所获得的收益相比相形见绌。我们还用一半的注意力能量隐藏维度训练了=2摩卡模型（这同样可以协调参数差异），发现它没有显著削弱我们的收益，仅增加了0.2%的WER（在八次试验中不显著）。另外，MoChA的一个可能的好处是，当产生上下文向量时，注意机制可以访问更大的输入窗口。另一种方法是增加卷积前端的时间感受野，因此我们也用这种变化重新训练了单调注意基线。同样，在八次试验中，表现的差异（平均每小时增加0.3%）并不显著。这些额外的实验加强了使用MoChA进行在线语音识别的好处。*w型*

## 3.2文件综述

在证明了MoChA在语音识别舒适环境中的有效性之后，我们现在在没有单调输入/输出对齐的任务中测试它的限制。Raffel et al.（2017）在Gigaword数据集上进行了句子摘要实验，该数据集经常呈现单调对齐，并且涉及短序列（单词的句子长度序列）。与软注意基线相比，他们在硬单调注意的情况下只能取得轻微退化的表现。因此，我们转向了一项更为困难的任务，即由于缺乏单调一致性，硬单调注意力更难以集中：CNN/每日邮报语料库上的文档摘要（Nallapati et al.，2016）。虽然我们主要研究这个问题，因为它具有潜在的挑战性，但在线和线性时间注意力在现实世界场景中也可能是有益的，在这些场景中，非常长的文本体需要在创建时进行摘要（例如，在演讲时生成摘要）。

这个任务的目标是从一篇新闻文章中产生一系列“突出”的句子。作为基线模型，我们选择了的“指针生成器”网络（没有覆盖率惩罚）（参见等人，2017）。有关完整的模型体系结构和培训细节，请参阅附录A.2。作为一个简单的总结，输入字被转换成一个学习的嵌入，并被传递到模型的编码器中，该编码器由一个双向LSTM层组成。解码器是具有注意机制的单向LSTM，其状态被传递到softmax层，该层在词汇表上产生一系列分布。该模型增加了一个复制机制，该机制在使用softmax输出层的单词分布或在给定的输出时间步长上由注意分布加权的单词id分布之间进行线性插值。我们用标准的软注意（如在（见等人，2017年）中所用）、硬单调注意和=8的摩卡咖啡测试了这个模型。*w型*

结果见表2。我们发现，使用硬单调注意机制会显著降低性能（接近8个胭脂-1点），可能是因为这项任务需要很强的重新排序。然而，摩卡咖啡能够有效地将单调注意力和柔和注意力之间的差距缩小一半，尽管使用了=8的适度块大小。我们认为这是一个令人鼓舞的迹象，表明能够处理本地重新排序的好处。*w型*

# 4相关工作

与摩卡类似的模型是“神经传感器”（Jaitly et al.，2015），其中输入序列被预分割成大小相等的非重叠块，并在每个块上分别执行注意序列到序列的转换。完整的输出序列是通过边缘化从每个块生成的序列的可能的序列结束位置而产生的。虽然我们的模型也对组块进行了软注意，但是我们的组块的位置是由一个硬的单调注意机制自适应地设置的，而不是固定的，并且它避免了在序列标记的组块端边缘化。

Chorowski et al.（2015）提出了类似的想法，其中在每个输出时间步计算软注意的范围被限制在围绕前一个输出时间步的最大注意概率的记忆索引的固定大小的窗口中。虽然这也会对语块产生软注意，但我们的方法不同之处在于，语块边界是由一个独立的硬单调注意机制设置的。这种差异导致Chorowski et al.（2015）使用了150的非常大的块大小，这有效地防止了在在线设置中使用它，并且比我们只需要很小的值的方法产生了更高的计算成本。*w型*

一类相关的非注意序列转导模型可用于在线设置，包括连接主义时间分类（Graves等人，2006年）、RNN传感器（Graves，2012年）、节段间神经转导（Yu等人，2016年）和节段RNN（Kong等人，2015年）。这些模型与具有注意机制的序列到序列模型的区别在于，解码器不直接以输入序列为条件，而是通过动态程序进行解码。（Prabhavalkar等人，2017）对这类方法和基于注意的模型进行了详细的比较，结果表明，基于注意的模型在语音识别实验中表现最好。此外，Hori等人（2017）最近提出了一种联合训练具有CTC丢失和注意机制的语音识别模型。这种结合鼓励了模型学习单调排列，但Hori等人（2017）仍然使用标准的软注意机制，排除了模型在在线环境中的使用。

最后，我们注意到还有一些其他工作考虑了硬单调对齐，例如使用强化学习（Zaremba&Sutskever，2015；Luo et al.，2016；Lawson et al.，2017），通过使用单独计算的目标对齐（Aharoni&Goldberg，2016）或通过假设严格的对角线对齐（Luong et al.，2015）。我们怀疑这些方法通过增加分块注意力可能会带来类似的好处。

# 5结论

我们已经提出了MoChA，一种注意机制，它在输入序列的自适应定位块上执行软注意。MoChA允许在线和线性时间解码，同时也便于本地输入输出重新排序。实验结果表明，MoChA在在线语音识别任务中取得了最先进的性能，其性能明显优于基于文档摘要的硬单调注意模型。在未来的工作中，我们有兴趣将MoChA应用于（近似）单调排列的其他问题，例如语音合成（Wang et al.，2017）和形态变化（Aharoni&Goldberg，2016）。我们还想研究允许块大小自适应变化的方法。为了促进我们的工作，我们提供了一个MoChA online的示例实现*w型*

## 致谢

我们感谢萧颖、凯文·克拉克、雅各布·巴克曼、我们的匿名评论员以及Google Brain团队的成员对本文的有益评论。

# 参考文献

Martin Abadi、Paul Barham、陈建民、陈志峰、Andy Davis、Jeffrey Dean、Matthieu

Devin、Sanjay Ghemawat、Geoffrey Irving、Michael Isard、Manjunath Kudlur、Josh Levenberg、Rajat Monga、Sherry Moore、Derek G.Murray、Benoit Steiner、Paul Tucker、Vijay Vasudevan、Pete Warden、Martin Wicke、Yuan Yu和Xiaoqiang Zheng。TensorFlow：一个用于大规模机器学习的系统。操作系统设计与实现，2016。

Roee Aharoni和Yoav Goldberg。序列到序列的传递，具有很强的单调性。arXiv预印本附件十四：1611.01487, 2016.

Dzmitry Bahdanau、Kyunghyun Cho和Yoshua Bengio。联合学习对齐和翻译的神经机器翻译。2015年国际学习代表大会。

Alexandra Birch、Miles Osborne和Philipp Koehn。预测机器翻译的成功。自然语言处理经验方法会议论文集，第745-754页。计算语言学协会，2008年。

William Chan、Navdeep Jaitly、Quoc V.Le和Oriol Vinyals。听、听、拼：一种用于大词汇量会话语音识别的神经网络。2016年国际声学、语音和信号处理会议。

赵京贤、巴特·范梅里安布尔、切格拉尔·居尔、德兹米特里·巴达瑙、费提·布加勒斯、霍尔格施文克和约书亚·本吉奥。学习短语表示使用RNN编码器-解码器的统计机器翻译。在2014年自然语言处理经验方法会议上。

简·乔洛夫斯基和纳夫迪普·贾特利。在序列到序列模型中实现更好的解码和语言模型集成。arXiv预印本附件十四：1612.02695, 2017.

简·乔洛夫斯基、德米特里·巴达瑙、德米特里·塞尔杜克、赵京贤和本吉奥。基于注意的语音识别模型。2015年神经信息处理系统会议。

亚历克斯·格雷夫斯。用递归神经网络进行序列转导。arXiv预印本附件十四：1211.3711, 2012.

亚历克斯·格雷夫斯、圣地亚哥·费尔南德斯、福斯蒂诺·戈麦斯和于尔根·施密杜伯。连接主义时间分类：用递归神经网络标记未分段的序列数据。在2006年国际机器学习会议上。

Gaël Guennebaud，Benoıt Jacob，Philip Avery，Abraham Bachrach，Sebastien Barthelemy等，Eigen v3。http://eigen.tuxfamily.org, 2010.

Takaki Hori、Shinji Watanabe、Yu Zhang和William Chan。基于深度CNN编码器和RNN-LM的联合CTC注意端到端语音识别研究进展。Interspeech，2017年。

谢尔盖·伊夫和克里斯蒂安·塞格迪。批量标准化：通过减少内部协变量的变化来加速深层网络训练。在2015年国际机器学习会议上。

纳夫迪普·贾特利、大卫·苏西洛、库克·V·勒、奥里奥尔·维尼亚、伊利亚·萨茨凯和萨米·本吉奥。神经传感器。arXiv预印本附件十四：1511.04868, 2015.

沃尔特·G·凯利和艾伦·C·彼得森。差分方程：应用简介。学术出版社，2001年。

迪德里克·金马和吉米·巴。Adam：随机优化的一种方法。arXiv预印本附件十四：1412.6980, 2014.

孔令鹏、克里斯·戴尔和诺亚·A·史密斯。分段递归神经网络。arXiv预印本附件十四：1511.06018, 2015.

理查德·E·拉德纳和迈克尔·J·菲舍尔。并行前缀计算。ACM杂志（JACM），27（4）：831–8381980。

迪特里希·劳森、乔治·塔克、钟正秋、科林·拉斐尔、凯文·斯沃斯基和纳夫迪普·贾特利。用变分推理学习硬对齐。arXiv预印本附件十四：1705.05524, 2017.

罗玉萍、钟成秋、纳夫迪普·贾特利和伊利亚·萨茨凯。在线学习与持续的奖励政策梯度一致。arXiv预印本附件十四：1608.01281, 2016.

Minh Thang Luong、Hieu Pham和Christopher D.Manning。基于注意的神经机器翻译的有效方法。在自然语言处理经验方法会议上，2015年。

Ramesh Nallapati、Bowen Zhou、Cícero Nogueira dos Santos、圣格拉尔Gülçehre和Bing Xiang。使用序列到序列RNN的抽象文本摘要。在计算自然语言学习会议上，2016。

道格拉斯·B·保罗和珍妮特·M·贝克。基于《华尔街日报》的企业社会责任语料库设计。演讲与自然语言研讨会，1992年。

Rohit Prabhavalkar、Kanishka Rao、Tara Sainath、Bo Li、Leif Johnson和Navdeep Jaitly。用于语音识别的序列对序列模型的比较。Interspeech，2017年。

科林·拉斐尔和丹尼尔·P·W·埃利斯。注意前馈网络可以解决一些长期记忆问题。arXiv预印本附件十四：1512.08756, 2015.

Colin Raffel、Minh Thang Luong、Peter J.Liu、Ron J.Weiss和Douglas Eck。在线和线性注意通过强制单调排列。2017年国际机器学习大会。

Abigail See、Peter J.Liu和Christopher D.Manning。切中要害：使用指针生成器网络进行摘要。arXiv预印本附件十四：1704.04368, 2017.

伊利亚·萨茨凯、奥里奥尔·维尼亚和库克·V·勒。用神经网络进行序列间学习。神经信息处理系统进展，2014。

王冲、Yogatama、Adam Coates、Tony Han、Awni Hannun和Bo Xiao。单向递归神经网络的前向卷积层。2016年第四届学习表征国际会议研讨会扩展摘要。

王宇轩、RJ Skerry Ryan、Daisy Stanton、Wu Yonghui、Ron J.Weiss、Navdeep Jaitly、杨宗衡、萧颖、陈志峰、Samy Bengio、Quoc Le、Yannis Agimyrgiannakis、Rob Clark和Rif A.Sauros。Tacotron：走向端到端语音合成。arXiv预印本附件十四：1703.10135, 2017.

Kelvin Xu、Jimmy Ba、Ryan Kiros、Kyunghyun Cho、Aaron Courville、Ruslan Salakhudinov、Rich Zemel和Yoshua Bengio。展示、参与和讲述：视觉注意力的神经图像字幕生成。在2015年国际机器学习会议上。

雷宇，简买，和菲尔布伦松。在线段间神经传导。在自然语言处理经验方法会议上，2016。

Wojciech Zaremba和Ilya Sutskever。强化学习神经图灵机。arXiv预印本附件十四：1505.00521, 2015.

张瑜、陈伟霆和贾泰利。用于端到端语音识别的非常深的卷积网络。arXiv预印本附件十四：1610.03022, 2016.

# 实验细节

在这个附录中，我们提供了关于第3节中进行的实验的细节。所有实验均采用TensorFlow（Abadi等人，2016年）。

## A、 1在线语音识别

总的来说，我们的模型遵循了（Raffel et al.，2017）的模型，但我们在此为后代重复这些细节。我们用80个系数的mel标度谱图以及delta和delta系数来表示语音。首先将特征序列分为两个卷积层，每层3×3个滤波器，每层32个滤波器，每层2×2步。每一次卷积之后都会进行批量标准化（Ioffe&Szegedy，2015），然后再进行重新线性化。卷积层的输出采用1×3滤波器馈入卷积LSTM层。接着是一个额外的3×3卷积层，带有32个滤波器，步幅为1×1。最后，编码器有三个额外的单向LSTM层，每个层的隐藏状态大小为256，紧跟其后的是一个密集层，输出为256维，具有批量归一化和ReLU非线性。

解码器是一个单一的单向LSTM层，隐藏状态大小为256。它的输入由先前输出符号的64维学习嵌入和注意机制产生的256维上下文向量组成。注意能量函数的隐维数为128。softmax输出层以注意上下文向量和解码器状态的级联作为输入。*d级*

使用Adam优化器（Kingma&Ba，2014）对网络进行训练，其值为=0.9，=0.999*β*1 *β*2 ，和。初始学习率在600000、800000和1000000步之后，0.001下降了10倍。请注意，Raffel等人（2017年）使用的学习率计划略有不同，但我们发现上述计划改善了软注意基线和摩卡咖啡的表现，但损害了硬单调注意的表现。因此，我们报告了来自（Raffel et al.，2017）的硬单调注意力表现，而不是重新运行该基线。输入被输入到网络中，使用标准的教师强迫，一批8个单词。局部标签平滑（Chorowski&Jaitly，2017）应用于目标输出，其权重为[−2，−1,1,2]处邻居的[0.015,0.035,0.035,0.015]。我们使用了梯度剪裁，当全局梯度向量的范数超过阈值时，就将其设置为1。我们在LSTM层参数和嵌入中加入了变权噪声，在20000个训练步骤后，标准偏差为0.075。我们还应用了L2权重衰减，系数为10−6。在测试时，我们使用了8个假设和3个剪枝阈值的秩剪枝波束搜索。

## A、 2文献综述

为了总结，我们重新实现了See等人（2017）的指针生成器。输入作为一个热向量提供，表示50000单词词汇表中的ID，映射到512维的学习嵌入。编码器由一个具有512个隐藏单元的双向LSTM层组成，解码器由一个具有1024个隐藏单元的单向LSTM层组成。我们的注意机制有一个隐藏的维度1024。输出字被嵌入到一个学习的1024维嵌入中，并在反馈到解码器之前与上下文向量连接。*d级*

对于训练，我们使用了Adam优化器，=0.9，=0.999*β*1 *β*2 ，和。我们的优化器的初始学习率为0.0005，从50000步开始连续衰减，每10000步学习率减半，直到达到0.00005。序列以64的批量输入模型。如See等人（2017）所述，我们将所有输入序列的最大长度截断为400个单词。梯度的全局范数被剪裁为永远不超过5。请注意，我们的模型中没有包含See等人（2017）中讨论的“覆盖率惩罚”。在评估过程中，我们使用了与语音识别实验相同的波束搜索，在8个假设下进行秩剪枝，剪枝阈值为3。

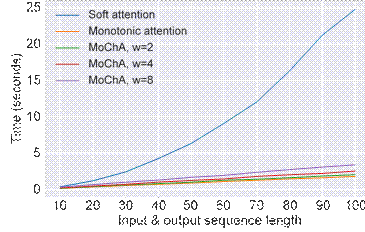


图3：综合基准上不同注意机制的速度。

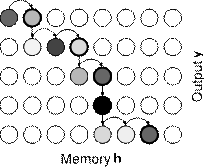
# B速度基准

为了了解使用摩卡代替标准软注意可能带来的加速，我们进行了一个简单的综合基准测试，类似于（Raffel et al.，2017），附录F中的基准测试。在这个测试中，我们单独实现了注意机制，并测量了其在不同输入/输出序列长度下的速度。这隔离了我们正在研究的网络部分的速度；在实践中，网络的其他部分（例如编码器RNN、解码器RNN等）可能支配运行完整模型的计算成本。因此，任何由此产生的加速都可以被视为现实世界中可能观察到的情况的上限。此外，我们使用特征库（GueNebAud等人，2010）在C++中对基准进行编码，以消除由特定模型框架引起的任何开销。

在这个合成设置中，注意力通过使用随机解码器状态的随机生成的编码器隐藏状态序列来执行。编码器和解码器的状态维数被设置为256。我们在{10,20,30，…，100}范围内同时改变输入和输出序列长度。我们测量了软注意、单调注意（即=1的摩卡咖啡）和={2,4,8}的摩卡咖啡的速度。一直以来，我们报告100次试验的平均值。*T型U型w型w型*

结果如图3所示。正如所料，软注意呈现出大致二次的时间复杂度，而摩卡的时间复杂度是线性的。这将导致随着时间的推移和时间的增加，加速系数增大。此外，摩卡咖啡的复杂度随时间线性增加。最后，请注意，对于=10和=8，摩卡和软注意的速度是相似的，因为块有效地跨越了整个内存。这证实了一种直觉，即摩卡咖啡的提速对于大型和相对小型的咖啡来说都是最引人注目的。*T型U型w型T、 U型w型T型U型w型*

# C单调自适应分块注意（MATCHA）

在这篇文章中，我们考虑了一种注意机制，它关注单调注意机制设置的位置之前的小的、固定长度的块。与这项工作同时，我们还考虑了另一种在线线性时间注意机制，它将组块设置为介于和之间的记忆区域。我们称这种方法为MAtChA，用于单调自适应的分块注意。这种选择背后的动机是，在某些情况下，对所有序列中的所有位置使用固定的块大小可能不是最佳的。然而，正如我们将在下面讨论的，我们发现它并没有提高性能超过摩卡的任何任务，我们尝试，尽管培训时间-*钛钛*−1

|  |  |
| --- | --- |
| 图4:MAtChA的测试时间解码程序示意图。节点和水平轴和垂直轴的语义如图所示。1a到1c。MAtChA通过一种单调的注意机制，在由关注的位置设置的可变大小的块上执行软注意。 | 具有增加的内存和计算要求的算法。我们在这里为后人讨论和推导MAtChA，以防其他研究人员有兴趣追求类似的想法。  总的来说，MAtChA的测试时间解码过程（也是在线和线性时间）与算法1极为相似，只是没有将块开始位置设置为=j−w+1，而是将=ti−1设置为*五五五* |

(27)

该过程如图4所示。请注意，如果=ti−1，那么MAtChA必须将所有注意力分配给内存条目，因为必须将块的唯一条目的概率分配为1。*钛钛*

MAtChA在输出timestep时注意内存输入的总公式可以表示为*日本我*

!

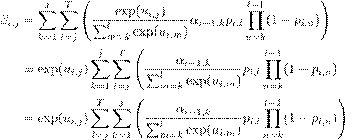
(28)

这个等式可以从左到右解释如下：首先，我们必须对单调注意力在前一个时间步∈{1，…，j}可能注意到的所有可能位置求和。第二，我们求和在当前输出时间步∈{j，…，T}可以参与的所有可能位置。第三，对于给定的输入/输出时间步组合，我们计算从到的区块上内存进入的softmax概率（在正文中，我们将由ChunkEnergy产生的注意力能量称为）。第四，我们乘以它表示单调注意机制在前一时间步注意记忆输入的概率。第五，我们乘以，单调注意机制在当前输出时间步选择记忆条目的概率。最后，我们将不选择任何内存项的概率乘以-1。使用公式（28）来计算以获得上下文向量的期望值允许利用MAtChA的模型通过反向传播进行训练。*k公司我日本k公司我乌伊，jαi*−1千*k公司皮，我我k公司我βi，jci公司*

注意，等式（28）包含多个嵌套的求和和和积，用于计算每一对。*i、 日本*

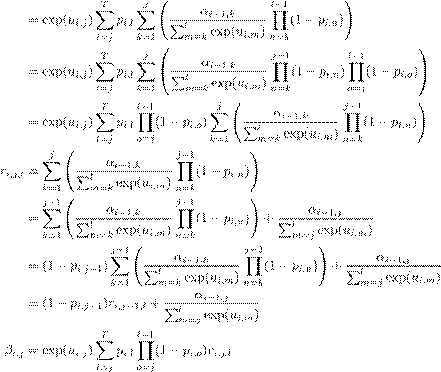
幸运的是，与单调的注意力和摩卡咖啡一样，有一个动态程序，允许完全并行计算，其可导出如下：*βi，*:

(29)

!

(30)

(31)

(32)

(33)

(34)

(35)

(36)

(37)

(38)

(39)

注意，式（38）的形式与式（11）相同；根据（Raffel et al.，2017）附录C.1的推导，它可以类似地表示为（可并行化的）累积和和和累积积运算。然而，式（38）和式（11）之间的一个显著区别是前者依赖于一个额外的指数变量。这是因为计算all和需要计算exp（ui，：）的所有可能子序列的和。幸运的是，这些子序列和也可以有效地计算；首先，定义*我李，j，l日本我*

                                                   全部部分（x）(40)

注意，对于一个长度序列，allpartialsum（x）产生一个形状为×T的矩阵**十***T型T型日本*

                                    所有部分（x）j，l=（x1+x2+。。。+xl）—（0+x1+。。。+xj−1）（41）

第一组括号中的和仅为的累计和的第个条目；第二组括号中的和为的排他累计和的第个条目。因此，通过计算累积和并进行适当的减法，可以有效地并行计算allpartialsum（x）的所有条目。结合以上和（Raffel等人，2017）附录的推导*我***十***日本十*

C、 1，我们有

                                                                         *αi*−1,:

*里，*:,:=cumprod（1−pi，：）cumsum

所有部件（exp（ui，：））cumprod（1−pi，：）

其中，作为一种轻微的滥用，我们使用的是“广播”[3]符号。以类似的方式，公式（39）中项（1−pi，o）的乘积涉及计算所有可能子序列的乘积。一个函数AllPartialProducts（·）可以类似地定义为等式（40），并用累积积和除法有效地计算。把它们放在一起，我们可以计算出所有的项*βi，j*

|  |  |
| --- | --- |
| 对于给定的输出时间步*我* |  |
| *T型*  *β我，*：=exp（u:）Xp：所有部分产品（1−p:）：，l:，l*我，我，我，右我，* | (43) |

*我*=j

虽然我们已经演示了一个计算MAtChA注意分布的并行过程，但是在所有可能的块开始和结束位置上的边缘化需要为每个输出时间步/内存项组合计算二次数量的项。即使在完全有效的并行化的情况下，结果也是一个需要O（UT）内存进行解码的算法（与训练标准软注意、单调注意或MoChA时所需的O（UT）内存相反）。这使得它处于一个明显的劣势，特别是对于大值的。实验上，我们曾希望这些缺点会被抹茶优越的经验表现所抵消，但不幸的是，我们发现它在我们尝试的任务中并没有表现得比摩卡更好。因此，我们决定不在正文中包括对MAtChA的讨论，并建议不要以目前的形式使用MAtChA。然而，我们有兴趣混合摩卡和抹茶在未来的工作中，试图收获他们的综合实力的好处。*T型*

[[1]](" \l "_ftnref1" \o ") 即使是非声调的话语（例如，“AAA”被转录为“AAA”），学习到的对齐方式仍然趋向于单调——参见例如（Chan et al.，2016）图6。

[[2]](" \l "_ftnref2" \o ")https://github.com/craffel/mocha

[[3]](" \l "_ftnref3" \o ") https://docs.scipy.org/doc/numpy-1.13.0/user/basics.broadcasting.html