从粗粒度到细粒度的神经机器翻译系统推断加速方法研究

张裕浩[1], 许诺[1], 李垠桥[1], 肖桐[1,2*], 朱靖波[1,2]
（1.东北大学自然语言处理实验室，辽宁 沈阳 110819；
2.沈阳雅译网络技术有限公司，辽宁 沈阳 110004）

摘要：近年来，Transformer 系统通过多层注意力网络的引入有效提升了翻译模型的译文质量，但与此同时大量注意力操作的使用也导致模型整体的推断效率相对较低。针对该问题，本文提出从粗粒度到细粒度(coarse-to-fine)方法，根据注意力权重中信息量差异对信息表示进行细粒度压缩，最终达到加速推断的目的。实验发现，在 NIST 中-英以及 WMT 英-德翻译任务上，该方法均能在保证模型性能的前提下，分别获得 12.2%与 11.4%的推断速度提升。同时，本文进一步分析了注意力操作在不同表示粒度下的信息量差异，对 coarse-to-fine 方法的合理性提供支持。

关键词：神经机器翻译；模型加速；coarse-to-fine
中图分类号：TP391　　**文献标识码：**A

1 引言

机器翻译（Machine translation）是一种利用计算机软件自动地对自然语言进行翻译的技术，它的发展从最开始基于规则的方式逐渐发展到目前备受关注的神经机器翻译系统。2017 年，Vaswani 等人提出 Transformer 模型[1]，使用注意力机制的方法对语言信息进行提取[2]，进一步提升了翻译系统的译文质量，这种方法很好地缓解了翻译过程中长距离依赖等问题，同时能使模型的并行训练过程更加友好。凭借其优势特点，这种方法已经得到众多研究人员的广泛认可，并在多个机器翻译评测任务以及实际应用中得到使用。

虽然利用 Transformer 模型可以获得高质量的译文结果，但是对于一些实时性很高的任务而言，其推断速度依旧无法令人满意。研究人员注意到，在 Transformer 模型的结构中存在大量的注意力操作，其占整体推断时间的比例高达 63.99%，因此对注意力操作进行加速可以有效降低模型的推断耗时。从该角度出发，Zhang 等人、Xiao 等人分别提出使用更加轻量级的平均注意力网络（AAN）[3]以及共享注意力网络（SAN）[4]对传统的注意力模型进行替代，最终达到加速推断的目的。而 Gu 等人使用非自回归解码器降低了翻译过程中前向依赖的问题，提升了推断过程中的并行效率从而提升推断速度[5]。

本文研究工作主要从第一个角度出发，旨在通过提升注意力操作的执行效率来达到加速推断的目的。我们在实验中发现，在 Transformer 系统中不同层的注意力权重所包含的信息量各有高低，因此简单地使用同样大小的空间对注意力权重进行提取将从一定程度上造成了信息空间的冗余，浪费不必要的计算资源。面对该情况，本文进一步地提出了从粗粒度到细粒度的 coarse-to-fine 推断加速方法，利用既有模型中每层注意力权重的信息量来对注意力机制中语言信息的表示空间大小进行动态调整，使得在保证译文质量不发生明显变化的情况下，加快模型的推断速度。

基金项目：国家自然科学基金面上项目（61876035）；国家自然科学基金重点项目（61732005；61432013）；国家重点研发计划（2019QY1801）；网络文化与数字传播北京市重点实验室开放课题
***通信作者：**肖桐（xiaotong@mail.neu.edu.cn）

Coarse-to-fine 方法具有简单易实现的特点，我们利用该方法在 NIST 中-英以及 WMT 英-德任务上进行实验，发现在性能没有明显损失的条件下，模型推断速度分别提升了 12.2% 和 11.4%以上。此外，我们在对 Transformer 系统的信息量进行分析的过程中发现了一个有趣的现象，对于自注意力操作而言，其注意力权重中包含的信息量的大小一般是随模型层数的升高而增大的，而对于编码-解码注意力操作而言，其信息量的变化趋势恰好与之相反。

2. Transformer 模型及注意力机制

Transformer 模型是一种基于序列到序列的网络架构，主要包括编码器与解码器两个部分，其中编码器用于对源语信息进行抽象，然后由解码器根据源语信息以及已经翻译出的译文片段对下一个候选译文词汇进行预测。二者内部均由若干同构子层堆叠而成，编码器子层内部主要包括自注意力结构以及前馈神经网络结构，解码器相较编码器而言，每层还增设了编码解码注意力模块，用于对跨语言信息进行有效建模。Transformer 模型的结构图如图 1 所示。

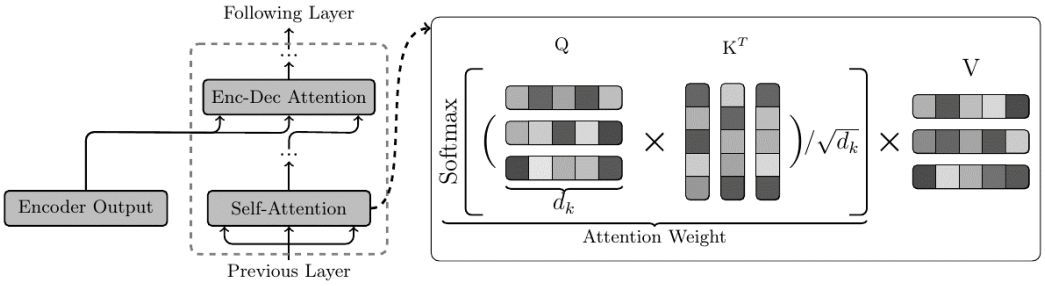


图 1 Transformer 结构及注意力机制

注意力机制在 Transformer 系统中被大量使用，该方法基于语言片段之间的关联程度对语义信息进行加权整合，最终获得更加精确的表示信息。其输入包括三个语言表示：Q($\in R^{l_q * d_k}$)、K($\in R^{l_k * d_k}$)以及 V($\in R^{l_q * d_v}$)，即 Q，K，V 都是二维矩阵，其中l_q，l_k为当前所处理的序列长度，d_k和d_v分别为信息表示 K 和 V 的维度，计算方式如式（1）所示。

$$Att = Softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V \quad (1)$$

其中注意力操作通过 Q 和 K 计算出语言片段间的注意力权重，然后以该权重从 V 中对信息进行提取，得到所需的上下文表示。在 Transformer 模型的自注意力操作中，Q、K、V 均由前一子层的输出信息变换而成，而对于编码解码注意力，其中的 Q 则对应于编码器端顶层上下文向量，K 和 V 来自于解码器端自注意力子层的输出。

3 基于粗粒度到细粒度的注意力计算

对于 Transformer 系统的推断过程而言，由于其推断过程中自回归的特性，解码端的时间占比远高于编码端的时间消耗，因此本文主要针对系统中解码端进行加速。

3.1 注意力权重信息量

注意力机制主要包括两个阶段，注意力权重的计算以及信息融合的过程。注意力权重的计算过程将捕获当前状态下最关注语言片段的位置，从公式（1）中可知，其计算复杂度高达$l_q * l_k * d_k$，在模型推断过程中占用较多时间。此外，根据 Wang 等人的工作，模型内部每一层中注意力机制所关注的内容类别各有侧重，因此不同层表示向量中所囊括的信息量也各有不同[6]。这里我们采用信息熵作为衡量不同层注意力权重中所包含信息量大小的指标。

信息论中，信息熵（Information entropy）常常被用来衡量事件中所包含信息的期望。由事件概率分布和每个事件信息量构成了一个随机变量，这个随机变量的期望便是分布产生的信息量的平均值（即熵）。对于一个受限数量的随机变量 X，其信息熵为：

$$H(X) = \sum_i P(x_i) I(x_i) = -\sum_i P(x_i)\, ln\, P(x_i) \tag{2}$$

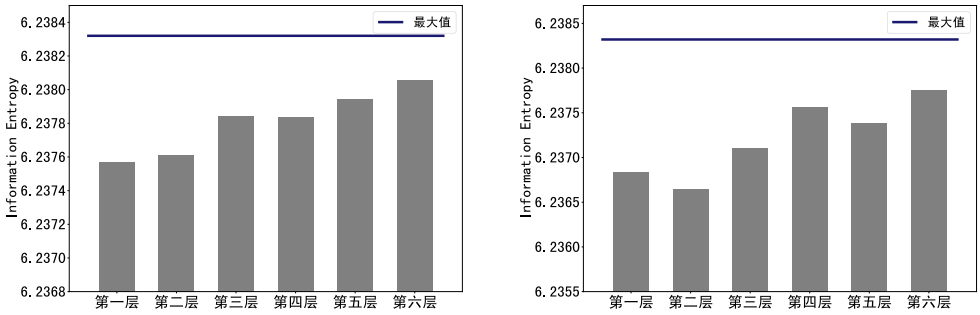
其中，$P(\cdot)$为变量的概率质量函数，$I(\cdot)$是信息量。

对于注意力权重中的概率质量函数，根据公式（1）中注意力计算，其计算公式如下所示。

$$P(S_t) = Softmax(\frac{(h_q * W_q)(h_k * W_k)^T}{\sqrt{d_k}}) \tag{3}$$

其中S_t为当前时刻 t 下注意力操作中候选语言片段被选中的事件，h_q和h_k分别为注意力子层的输入，其通过相应的同维度变换矩阵W_q和W_k（二者均$\in R^{d_k * d_k}$）得到公式（1）中的 Q 和 K，通过公式（2）的计算得到 t 时刻下注意力权重的信息熵$H(S_t)$。由于不同时刻之间的注意力权重相互独立，因此对模型某一层的权重信息量衡量可直接以信息熵的均值作为衡量标准。

我们对模型中不同层注意力权重的信息量进行了分析，发现其中每一层的信息量存在明显差异。如图 2 所示，对于编码器中的自注意力操作而言，信息熵整体呈现随层数递增的趋势。这种情况下，普通的 Transformer 模型使用相同维度的 Q 和 K 对每一层的注意力权重进行计算将导致信息冗余的现象存在，即其计算粒度非常的粗，在推断过程中产生不必要的计算。图 2（a）中，从数值上看虽然信息量最高和最低层的差别非常小，这是因为所有位置的权重是归一的，而句子长度较长的话，每个位置的权重值都会比较低，这也导致了即使各层之间注意力分布存在着显著不同但是计算出的信息量数值上差距不大。



(a)中英　　(b)英德

图 2 编码自注意力各层信息量及最大信息量

3.2 基于粗粒度到细粒度的注意力权重计算

面对注意力权重计算过程中存在的冗余信息问题，我们提出使用粗粒度到细粒度的 coarse-to-fine 的方法对注意力权重计算过程中的输入表示 Q 和 K 进行信息压缩，通过降低其信息表示的粒度来达到推断加速的目的。具体来说，我们将对模型注意力权重计算过程中信息表示维度大小进行降低，通过降维后的$W_q{}'$和$W_k{}'$（二者均$\epsilon R^{d_k * d_k{}'}$）使粗粒度的信息表

示 Q 和 K 变为细粒度的Q'（$\in R^{l_q*d_k'}$）和K'（$\in R^{l_k*d_k'}$）。基于 coarse-to-fine 的注意力权重计算过程如图 3 所示。

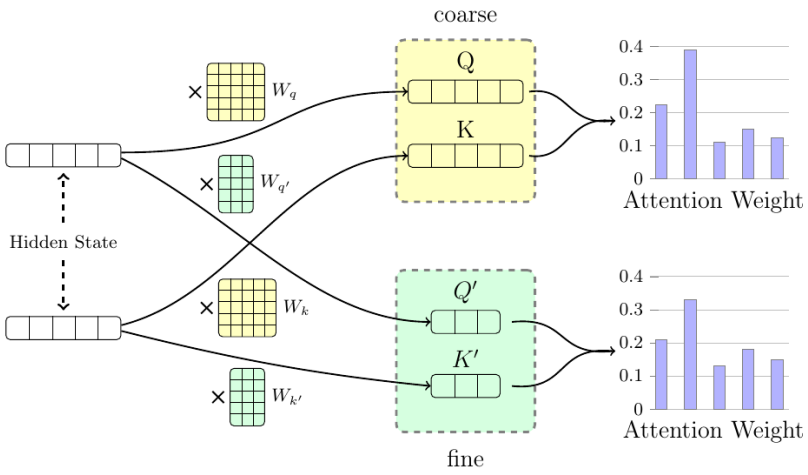


图 3 标准方法与 coarse-to-fine 方法注意力权重计算比较

算法 1：基于 coarse-to-fine 的表示压缩模型训练方法

输入：*layers*，*model*

train a *model*
for *i* = 1 → *layers* ：
    calculate information entropy H_i on dev set
analyze a policy $\{d_k'\}$ on given *model*
train a new model use $\{d_k'\}$
return *model*

对于信息压缩后的表示维度，一种简单的方式为根据在模型在开发集上的性能对每一层设定一个相同大小的d_k'。但是由于统一的d_k'并没有将不同层信息量的差异考虑进来，因此我们提出一种基于信息熵 H 的压缩维度计算策略，根据注意力权重中的信息量对Q'和K'的所需的表示维度进行预估。具体来说，我们按照信息熵的大小将每一层的压缩比例映射到[a, 1]区间内，最终得到的细粒度表示维度如式（4）所示：

$$d_k' = f(H) = \left\lceil \left(a + \frac{H-Min}{(Max-Min)} * (1-a)\right) * d_k \right\rceil \quad (4)$$

其中$\lceil\cdot\rceil$代表向上取整函数，a 代表了映射比例的下界，用来控制表示空间压缩程度。Min 为所有层中信息量的最小值，而 Max 有两种选择方式，第一种是选取层中信息量最大值，第二种是选取注意力权重能表示最大值，其计算方式如式（5）所示。

$$Max = d_k * (1/d_k) * \ln(1/d_k) = \ln(1/d_k) \quad (5)$$

此外如果需要进行较大比例压缩的时，即当 a 趋近于 0 时，底层d_m可能也会趋近于 0，导致性能急剧下降，因此我们会根据 Max 与 Min 值进行适当的缩放，防止d_k'过小。还有一种情况是如果 Max 与 Min 差距过大时，可能会造成信息量最小的层过小，此时需要适当调高 a 的值。

为保证模型训练与推断过程的一致性，基于 coarse-to-fine 的注意力权重压缩模型训练

过程也需要对维度进行调整。首先，我们对原始模型进行初步训练，然后在开发集上根据公式（2）对模型中每一层注意力权重中所包含的信息熵H_i进行计算，使用公式（4）根据各层信息量的差异计算出压缩后的细粒度表示维度，得到每层各自的信息空间大小$\{d_k'\}$，最终根据该 coarse-to-fine 策略对模型进行重新训练，得到支持快速推断的模型。整个训练过程如算法 1。

4 实验

本文在中英任务上采用的训练数据为NIST 数据集，共 1897807 句，中文词数 208997117，英文词数 229858458，校验集从数据集中抽出 1664 句，测试集为 news-08，共 1357 句。在英德任务上数据均来自于 wmt14 数据集，训练数据 4528446 句，德语词数 534482778，英语词数 488459359，校验集 3000 句，测试集 3003 句。中文处理工具使用 NiuTrans toolkit[7]，其他语言使用 WMT 官方处理方法。生成词表时都是用了 BPE 方法，词表大小为 3.2 万。

Transformer 结构中使用了 6 层编码器和 6 成解码器，在进行多头注意力计算时，默认使用 8 个头，每个头中d_k及d_v为 64，前馈网络隐层大小为 2048，利用 4 卡 Titan X(Pascal) 训练 15 轮。测试推断时使用了 15 个检查点模型进行参数平均融合，同时使用了束搜索，束搜索大小 12，推断 batch 大小 16。我们在使用 coarse-to-fine 方法时，默认使用层中最大信息量作为 Max 值。

本文首先分别对比了编码与解码端的自注意力权重计算进行裁剪之后相较于基线的性能及速度，然后对比了编码解码注意力权重计算与基线及平均压缩的性能。之后分析了信息量变化以及权重预测的变化，最后进行了敏感实验及与束搜索方法的比较。

4.1 方法对速度及性能的影响

我们根据 coarse-to-fine 方法计算出的信息量来调整每一层的压缩比例，然后在中英和英德翻译任务上比较了同不进行压缩及每层相同压缩的质量和速度，如表 1 所示。表 2 中进一步列出了利用 coarse-to-fine 方法计算出的每一层进行压缩的比例。

表 1 coarse-to-fine 方法与基线、平均压缩方法在不同任务比较

任务	模型	压缩比例 (%)	BLEU(%)	Δ BLEU(%)	速度 (token/sec)	Δ 速度(%)
Zh2En	基线	0	44.19	-	328.95	-
	coarse-to-fine	40.6	44.03	-0.16	374.53	12.2
	注意力平均		43.38	-0.81	371.75	11.5
En2De	基线	0	27.55	-	334.44	-
	coarse-to-fine	37.5	27.52	-0.03	377.36	11.4
	注意力平均		27.35	-0.2	369.0	9.4

可以从表中看出，无论是在中英任务还是英德任务上，基于 coarse-to-fine 方法在损失的性能以及带来的加速效果都优于平均压缩的方法，同时相较于基线质量都没有明显的质量变化。该方法通过压缩注意力权重计算维度，将推断速度在两个任务上分别提升了 12.2%和 11.4%。从表 2 中可以看到，使用细粒度压缩后英德编码注意力计算维度为压缩前的 6.25%，但是最后的性能只损失了 0.03，反映出 coarse-to-fine 方法是一种可靠、有效、且不会带来明

显损失的加速推断方法，其确实达到了减少冗余计算的目的。

表 2 coarse-to-fine 方法对注意力各部分压缩比例

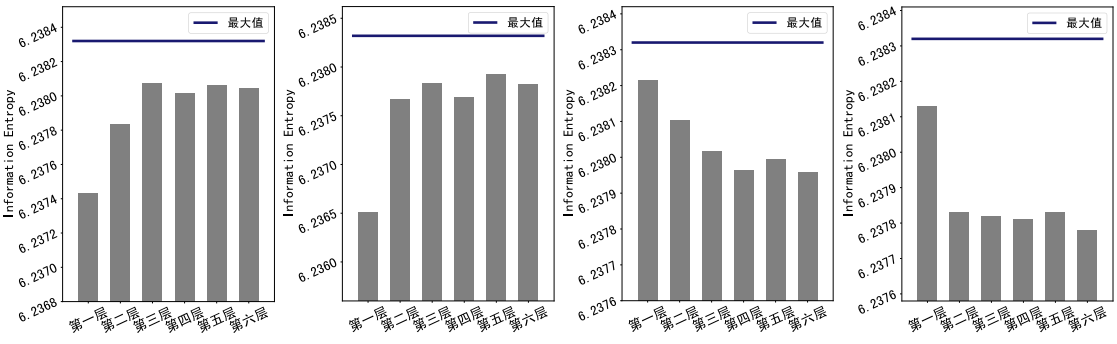
任务	类型	第 1 层(%)	第 2 层(%)	第 3 层(%)	第 4 层(%)	第 5 层(%)	第 6 层(%)
Zh2En	encode	50	50	25	25	25	0
	decode	80	50	25	25	25	25
	enc-dec	18.7	40.6	60.9	81.2	60.9	81.2
En2De	encode	75	81.25	56.25	31.25	37.5	18.75
	decode	93.75	18.75	12.5	18.75	6.25	12.5
	enc-dec	20	50	50	50	50	50

4.2 方法对解码端注意力行为的影响

我们首先对解码端自注意力信息量进行了分析，如图 4(a)与(b)，我们首先计算出训练好的模型解码端自注意力机制的各层信息量，可以看出在中英和英德任务上信息量都出现从底层到高层都出现了一个上升的趋势，但是和图 2 中编码端的自注意力权重信息不同的是，第二层相较于第一层的信息量出现了较大的提升，而第二层以上的信息量变化则小幅波动。我们推测是因为解码端除去第一层之外的隐层状态都通过编码解码注意力得到的了编码端的信息，所以导致了信息量的快速增长。同时每层的编码解码注意力对于编码端的信息也会有所选择，所以最顶层的解码信息量也不一定是最大的。如图 5(a)，其表示了词“weeks”在解码端自注意力权重，可以看到，在底层的注意力权重是比较尖锐的，因为其关注到的信息很有限，主要关注到了“few”这一个词，而到了顶层之后，其分布变得更加平滑，证明顶层关注到了更多的信息，比如“called”，“the”等，相较于底层，“few”词的权重也下降了很多，这也证明了顶层的信息量更大，需要更高维度进行计算表示。

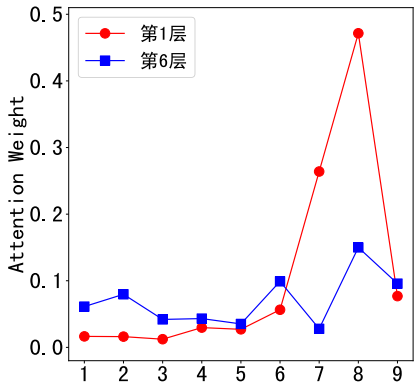
分析图 4 中的(c)与(d)编码解码注意力柱状图，我们容易得出从底层到顶层的注意力信息量呈现一个下降的趋势，第一层的信息量明显高于之后的信息量。我们可以从编码解码的作用来解释，对于解码器来说，底层的信息不足，无法直接关注到编码器中的对应信息，所以会把相关的信息一起传送到上层，而解码器的上层收集到的信息足够多之后便可以更多的关注到编码器对应的词。从图 5(b)中可以看出词“intensive”在底层的时候对于结束符和“呼吁”权重较高，但是到顶层之后对于“密集”的权重占了 70%，对于其他的词关注的权重都小于 10%，所以在顶层的时候关注的点非常集中，在编码解码注意力权重在底层的时候信息量比较多，反而需要使用高维度去计算表示。

同时对比每层能表示的最大的信息量，可以看出即便是信息量最大的层依然存在着一定的信息冗余问题。

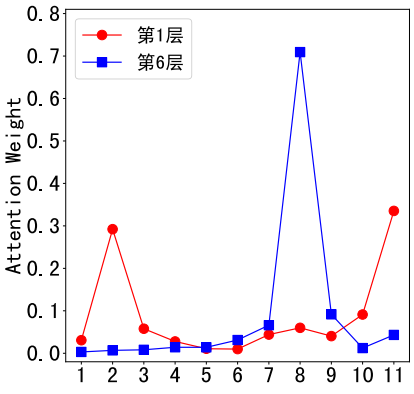


(a)中英解码端　(b)英德解码端　(c)中英编码解码　(d)英德编码解码

图 4 不同任务解码自注意力与编码解码注意力信息量比较



(a)词“weeks”解码自注意力权重



(b)词“intensive”编码解码注意力权重

图 5 句子“他 呼吁 在 未来 几 周 进行 密集 谈判 .”译成“He called for intensive negotiations over the next few weeks .”注意力权重输出

我们主要在中英任务上进一步探讨 coarse-to-fine 方法中压缩比例对性能和速度的影响，同时为了进一步验证方法的有效性，我们也和相同比例平均压缩每一层的方法进行了对比。

表 3 解码自注意力不同比例压缩前后比较

模型	压缩下界 a	压缩比例(%)	BLEU(%)	Δ BLEU(%)	速度 (token/sec)	Δ 速度(%)
基线	-	0	44.19	-	328.95	-
coarse-to-fine	0.5	18.7	44.16	-0.03	338.98	3
平均			43.74	-0.45	341.3	3.6
coarse-to-fine	0	37.5	44.23	+0.04	355.87	7.6
平均			43.94	-0.25	346.02	4.9

从表 3 中可以看出对于解码端的自注意力机制，压缩 30%的时候，性能基本不受影响，压缩比例提升一倍之后，速度提升了一倍多，但是性能反而相较于基线略有提升。我们分析其原因可能是因为压缩之后有些噪声信息恰好被去掉，没有继续传给下一层，从而使上层的预测更加准确，这也从侧面证实了基线系统每层表达的信息量不一定是最优的。同时我们注意到，不管压缩比例如何，coarse-to-fine 方法在性能的损失及速度提升均明显优于平均压缩的方法。

表 4 编码解码注意力不同比例压缩前后比较

模型	压缩下界 a	压缩比例(%)	BLEU(%)	Δ BLEU(%)	速度 (token/sec)	Δ 速度(%)
基线	-	0	44.19	-	328.95	-
coarse-to-fine	0.5	31.3	43.91	-0.28	337.84	2.6
平均			43.74	-0.45	334.45	1.6
coarse-to-fine	0	56.3	43.86	-0.33	346.02	4.9
平均			43.4	-0.79	340.14	3.2

我们对编码解码注意力机制也进行了类似实验，从表 4 中可以看到压缩更大的比例能获得更大加速，不过性能也会小幅下降。同时，coarse-to-fine 的方法在编码解码注意力其损失的性能与速度的提升也明显优于平均压缩的方法。

4.3 方法对编码端注意力影响

对于编码端的自注意力计算，我们根据信息量的大小，分别将每一层压缩的比例控制在 0.5-1 与 0-1 之间，对比不压缩和相同压缩比例且每层相同压缩比例的情况。由于在实际推断的时候会暂存编码器的输出，所以对于编码端来说，压缩之后对于推断速度提升只有 0.09%，但是质量有了明显下降。

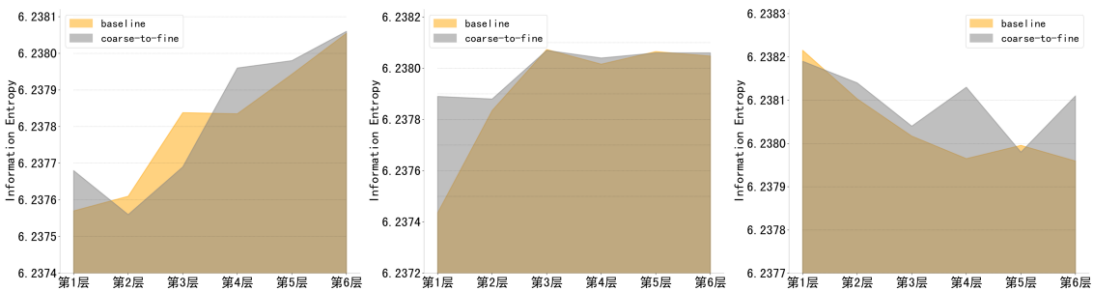
表 5 编码端注意力压缩实验比较

模型	压缩比例(%)	BLEU(%)	Δ BLEU(%)	速度 (token/sec)	Δ 速度(%)
基线	0	44.19	-	328.95	-
coarse-to-fine	31.25	43.72	-0.47	332.23	0.09
平均		43.58	-0.61	330.03	0.03

从表 5 中可以得出，将 coarse-to-fine 方法应用于编码器的自注意力权重计算，获得的加速效果与损失的性能是不成正比的，因此在进行整体压缩的时候对于解码端可以压缩得更少甚至不用压缩。不过 coarse-to-fine 方法的在速度与性能损失上依旧略优于平均的方法。

4.4 不同粒度信息量比较

本文将不进行压缩即粗粒度计算和使用 coarse-to-fine 方法得到的细粒度计算进行了比较，如图 6，可以看到压缩之后大部分层都没有发生信息量的剧烈变化，证明了 coarse-to-fine 方法压缩之后依然能够保证之前的信息不发生剧烈的变化。但是在压缩比例较大的层数时，还是出现了信息量较大的波动，过低的维度进行计算导致了无法舍去无用的信息，所以即使增加了信息量，也都是无用信息，并不会提升模型性能，不过由于模型对于信息的进一步提炼，最后无用信息也会在上层被舍去，因此最后也只带来了不到 0.2 个 BLEU 值的损失。这也从侧面反映了之前每一层都使用高维度来算出注意力权重是没有必要的，验证了 coarse-to-fine 方法的正确性，同时也解释了压缩的方法在加速计算的同时不会带来明显性能损失的原因。



(a)编码自注意力　(b)解码自注意力　(c)编码解码注意力

图 6 中英信息量压缩前后变化

不过，我们认为压缩之后的信息量过大或者过小可能都不好，过多相较于基线系统存在着更多噪音信息，而如果过少则没能捕捉到有用的信息，甚至可能没有将最关键的信息捕捉到。虽然按照每层最合适的信息量去调整压缩比例能达到最好的性能，然而实际上很难得到最优信息量的值，因此我们选择基线的信息量作为参考值。

4.5 不同粒度注意力权重比较

从 4.4 节中的实验可以看出，压缩前后实际上信息量变化并不太大，也就是说压缩之后依然可以表示出压缩前的信息，如果确实成立的话，那么压缩前后输出的注意力权重分布应该也会类似，所以我们对比了压缩前后输出的注意力权重分布，如图 7，我们发现其整体趋势是一致的，并且最应该注意到源语端的词也同样得到了最大的权重，相较于其他词差异也十分明显。虽然对于其他词的权重略有变化，但是细粒度的权重分布与粗粒度差的绝对值不超过 0.05。这进一步从侧面证明了之前的计算确实存在着信息冗余的现象，即矩阵存在着 coarse 的情况，也解释了使用 coarse-to-fine 方法在提升速度的同时也不会带来明显的性能损失的原因。

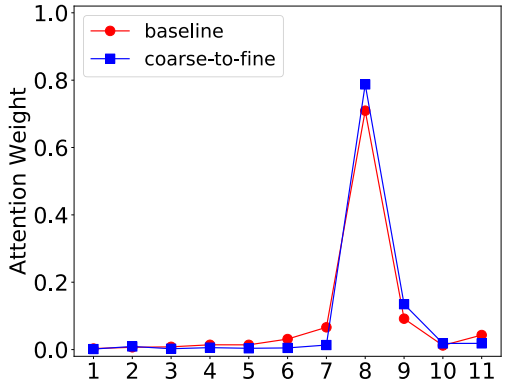


图 7 压缩前后第 6 层编码解码注意输出分布比较

4.6 敏感实验

为了测试方法的泛化能力，我们在中英任务上使用更大的参数量进行对比实验。我们将其隐藏层大小从 512 扩大到 1024，将前馈网络中的 filter 大小扩大到 4096，对比利用 coarse-to-fine 方法压缩前后质量和速度变化。

表 6 coarse-to-fine 方法对不同参数比较

模型	压缩比例(%)	BLEU(%)	Δ BLEU(%)	速度 (token/sec)	Δ 速度(%)
基线	0	45.18	-	186.57	-
coarse-to-fine	31.3	45.22	+0.04	206.19	9.5

可以从表 6 中看到，利用 coarse-to-fine 在将权重计算的整体维度压缩为原来的约 32% 之后，相较于基线系统，性能提升了 0.04 个 BLEU 值，同时推断速度提升了 9.5%。由于前馈网络计算部分占比变大，所以对注意力部分压缩对速度提升的比例有所下降。表 6 证明了 coarse-to-fine 方法在参数量更大的时候也有同样效果，并且速度提升效果同样显著，是一种

对模型参数不敏感的方法。

4.7 不同 Max 值选择比较

在 3.2 节中设计了两种 Max 值选取，第一种是计算出每一层的信息量然后选择其中最大的值，第二种选择每一层所能表示的最大的信息量，我们将两种方法在中英任务上进行了比较实验。由于只用使用第二种方法会使两者压缩差距比例过大，为了保证压缩比例大致相同，所以我们适当调大了压缩下界 a。同时由于总体压缩比例相近导致两种方法在速度上差异不大，不做进一步讨论。

表 7 Max 值选择实验

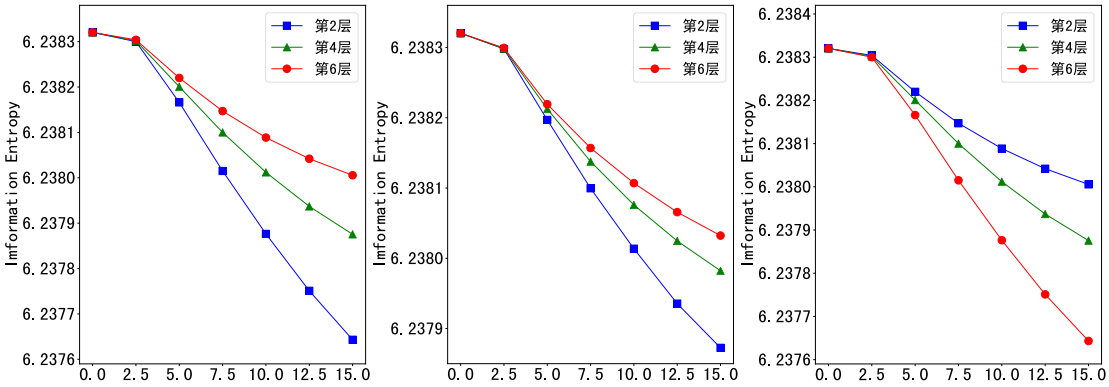
Max 值	压缩下界 a	压缩比例(%)	BLEU(%)	与基线 Δ BLEU(%)
计算出信息量中最大	0	40.3	44.03	-0.13
最大信息量	0.2	42.9	43.54	-0.65

从表 7 可以看出，选取最大信息量的然后按比例压缩的方法相较于基线系统损失了 0.65，但是按照从每层计算出的信息量的方法只损失了 0.13 个 BLEU，我们分析认为，利用最大的信息量来压缩，会将信息量较多层压缩得更多，因为最大信息量和计算出得每层信息量差距还是很大。所以尽管选取最大信息量作为 Max 值看似更为合理，但是由于按比例计算对于信息量较多的层压缩过多，导致了很多有用信息的丢失。

4.8 训练过程中信息量变化

由于得到一个训练收敛的模型是一个十分耗时的过程，我们将中英任务训练过程中每经过 2.5 轮训练第 2，4，6 层的信息量变化描绘成图 8，以测试能否提前进行细粒度压缩。我们注意到训练初始状态每一层的信息量差异不大的，在经过 2.5 轮训练后各层之间并没有明显的差异，但是训练到 5 轮后，各层之间的差异开始显现了，训练轮数越大对于自注意力是层数越高信息量越大，对于编码解码注意力是层数越低信息量越高，这也符合我们之前结论。

我们进一步探索是否能在模型尚未完全收敛时候得到 coarse-to-fine 的压缩比例。由于公式（4）在计算压缩比例的时候是按照信息量在 Max 值和 Min 值之间进行映射，所以不受信息量值绝对值的大小影响，因此我们主要关注了编码解码注意力中间层相对于最大值最小值的比例。



(a)编码自注意力　(b)解码自注意力　(c)编码解码注意力

图 8 训练轮数与信息量变化

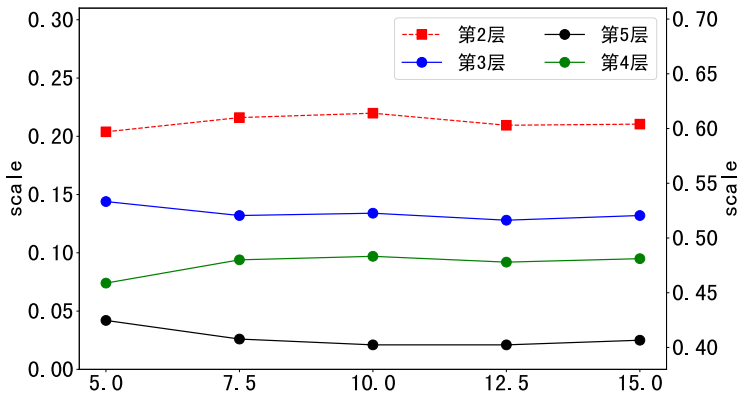
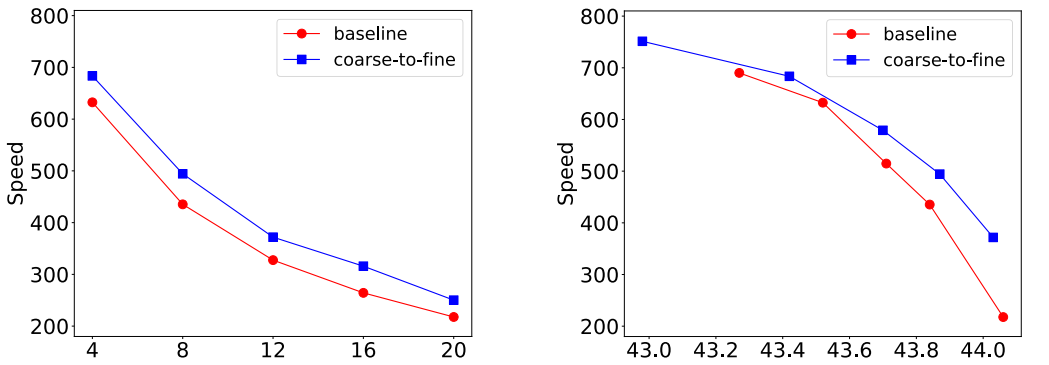


图 9 不同层信息量相较于第底层与最顶层所占比例(虚线对应右侧坐标，实线对应左侧坐标)

从图中可以看到 5 轮的中间各层的比例明显不同于 7.5 轮及以后的曲线，我们进一步对比 7.5 轮之后与 15 轮的数值，发现最大的波动仅为 0.04%。这证明了在模型收敛之后，虽然信息量数值在发生变化，但是模型之间的差距比例基本都已经固定，最后输出的结果也只有小幅波动，这也证明了按照信息量比例作为 coarse-to-fine 方法压缩比例的参数是合理的。同时我们可以进一步的看到，由于从 10 轮之后的信息量比例的波动很小，同时最后计算维度时需要向上取整，所以相较于训练结束每个头计算维度的变化最大不超过 2，不会对结果带来较大影响。因此我们也可以采用 10 轮的信息量作为最后维度压缩的参考，以避免为了得到收敛的模型而消耗大量的时间。

4.9 对比试验

为了获得推断加速的效果，最常见的使用方法是减小束搜索大小的方法，为了验证 coarse-to-fine 方法的有效性，我们将只改变束搜索大小和两种方法融合使用在中英任务上进行了对比。



(a)不同束搜索大小下推断速度比较(token/sec)　　(b)不同性能情况下与推断速度比较(token/sec)

图 10 coarse-to-fine 与束搜索方法比较

可以从图 10(a)中看到，相较于仅仅改变束搜索的大小，在其基础上继续使用 coarse-to-fine 方法也能进一步提升推断速度。我们将两种方法速度与性能损失做了比较，发现在损失同样性能的情况下，结合两种方法比单独减少束搜索大小能获得更高的速度提升，同时在同样推断速度的情况下，结合两种方法的性能损失也更小。

5. 相关工作

目前最优秀的模型都非常依赖于注意力机制，许多研究人员都尝试将注意力机制应用到模型中[8]，最早是Luong等人，提出了基于RNN的可加性注意力模型[9]，之后Gehring等人也将多层注意力也应用到了循环神经网络中[10]，然后Vaswani等人提出了Transformer结构，之后Transformer因为其易于大规模训练，不错的性能及优良的结果设计广受欢迎。

但是 Transformer 的推断速度甚至比基于 RNN 的同类算法还要慢。一部分原因是由于译码的自回归性质，一部分原因是由于注意力机制需要使用大量矩阵乘法而造成的，研究人员已经开始探索解决方案。例如，Gu 等人为 transformer 系统设计了一种非自回归推断方法，一次生成整个目标序列。这种模型虽然速度快，但难以实施。在另一项研究中，张等人提出了平均注意网络(AAN)，并将其应用于解码器端的自注意力子层。Xiao 等人提出了注意力共享方法。

在神经机器翻译中，网络加速的方法已被研究多年。其中包括 L'Hostis 等人，Sankaran 等人利用了词汇选择方法[11,12]，Hinton 等人，Kim 和 Rush 知识蒸馏[13,14]，Micikevicius 等人，Quinn 和 Ballesteros 利用了低精度计算[15,16]，Dabre 和 Fujita 发明了递归堆叠网络等[17]，我们的方法和之前的方法是相容的。以往的研究主要集中在减小模型大小和更快训练上，并没有重点关注到推断上。

但是之前的方法没有将模型中冗余的计算去掉，同时带来了明显的性能下降，有的方法还难以复现。我们提出的 coarse-to-fine 的方法只是修改了模型的一小部分，其方法易于实现，而且在几乎没有带来性能损失情况下带来了不错的速度提升，同时，其并不和其他的方法相斥。

6. 总结

我们发现了在 Transformer 结构进行推断时，计算注意力占到了大量的时间，但是其中存在着冗余的计算。我们利用衡量信息量的方法，发现了各部分且不同层之间的信息量都有着差异，在自注意力机制中，信息量从底层到顶层是上层的，而在编码解码注意力机制中，信息量反而是下降的，同时，相较于能表示的最大信息量，都存在信息冗余的情况。因此我们在提出了一种 coarse-to-fine 的方法，基于每层的信息量来指导每层压缩的比例，以此来达到加速推断的目的。

我们在实验中验证了 coarse-to-fine 的方法在中英和英德任务上进行压缩，在性能不发生明显损失的情况下，分别加速了 12.2%和 11.4%，也明显优于基于每层平均压缩的方法。同时我们分析了压缩前后信息量的变化，自注意力底层有所波动，但是顶层基本没有变化，而编码解码注意力恰好相反，符合我们的预期。我们还进一步比较了压缩前后的输出的注意力权重，发现没有太大变化，证明了冗余计算确实存在及我们的方法在加速的同时不会发生明显的性能下降，最后实验了不同模型参数压缩的效果，证明了 coarse-to-fine 方法对模型参数不敏感。

参考文献

[1] VASWANI A, SHAZEER N, PARMAR N, et al. Attention Is All You Need[C]. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, USA: MIT Press, 2017:6000-6010.

[2] DZMITRY D, CHO K, and BENGIO Y. Neural machine translation by jointly learning to align and

translate[C]. In Proceedings of the 3rd International Conference on Learning Representations, 2015.
[3] ZHANG B, XIONG D, and SU J. Accelerating neural transformer via an average attention network[C]. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018:1789–1798.
[4] XIAO T, LI Y, ZHU J, et al. Sharing Attention Weights for Fast Transformer[J]. arXiv preprint arXiv:1906.11024 ,2019.
[5] GU J, BRADBURY J, XIONG C, et al. Non-Autoregressive neural machine translation[J]. In International Conference on Learning Representations, 2018.
[6] WANG Q, LI F, XIAO T, et al. Multi-layer Representation Fusion for Neural Machine Translation[C]. Proceedings of the 27th International Conference on Computational Linguistics, ACM, 2018:3015-2036.
[7] XIAO T, ZHU J, ZHANG H, et al. NiuTrans: an open source toolkit for phrase-based and syntax-based machine translation[C]. Proceedings of the ACL 2012 System Demonstrations. Association for Computational Linguistics, 2012: 19-24.
[8] WU Y, SCHUSTER M, CHEN Z, et al. Google's neural machine translation system: Bridging the gap between human and machine translation[J]. arXiv preprint arXiv:1609.08144, 2016.
[9] LUONG M, PHAM H, and D.MANNING C. Effective approaches to Attention-based neural machine translation[C]. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015:1412–1421.
[10] GEHRING J, AULI M, GRANGIER D, et al. Convolutional sequence to sequence learning[C]. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, ACM, 2017:1243–1252.
[11] L'HOSTIS G, GRANGIER D, and AULI M. Vocabulary selection strategies for neural machine translation[J]. CoRR, abs/1610.00072, 2016.
[12] SANKARAN B, FREITAG M, and Al-ONAIZAN Y. Attention-based Vocabulary Selection for NMT Decoding[J]. CoRR, abs/1706.03824, 2017.
[13] HINTON G, VINVALS O, and DEAN J. Distilling the knowledge in a neural network[C]. In NIPS Deep Learning and Representation Learning Workshop, 2015.
[14] KIM Y and M.Rush A. Sequence-level knowledge distillation[C]. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, 2016:1317–1327.
[15] MICIKEVICIUS P, NARANG S, ALBEN J, et al. Mixed precision training. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 2018.
[16] QUINN J and BALLESTEROS M. Pieces of eight: 8-bit neural machine translation[C]. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HTL 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 3 (Industry Papers), 2018:114-120.
[17] DABRE R and FUJITA A. Recurrent stacking of layers for compact neural machine translation models[C]. In Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI), 2019.

Research on inference acceleration method of Neural Machine Translation system based on coarse-to-fine

ZHANG Yuhao, XU Nuo, LI Yinqiao, XIAO Tong and ZHU Jingbo
(NLP Laboratory, Northeastern University, Shenyang, Liaoning, 110819, China)

Abstract : In recent years, Transformer system has effectively improved the translation quality of the translation model through the introduction of multi-layer attention network, but at the same time, the use of a large number of attention operations has also led to the overall inference efficiency of the model is relatively low. In order to solve this problem, this paper proposes a method based on coarse-to-fine, which compresses the information representation according to the difference of the amount of information in the attention weight, and finally achieves the purpose of accelerating decoding. The experimental results show that on the Chinese-English translation task of NIST and the English-German translation task of WMT, the inference speed of this method can be improved by 12. 2% and 11. 4% respectively on the premise of ensuring the performance of the model. At the same time, this paper further analyzes the information difference of attention operation under different representation granularity, which provides support for the rationality of coarse-to-fine method.

Key words: Neural machine translation; Inference acceleration; coarse-to-fine