语音识别端到端模型调研

田正坤*

June 15, 2018

1 背景

语音识别在深度学习时代迎来了新的发展。传统的语音识别模型通常包含声学模型(Acoustic Model, AM)、发音词典(Lexicon)和语言模型(Language Model, LM)三部分组成。每一部分都需要单独的学习训练，如何建立一种端到端(End-to-End)的学习机制，可以使得模型的训练摒弃发音词典和语言模型，真正实现直接从语音转录成文本(Speech-to-Text)已经成为当下的研究热点。目前实现端到端学习主要有两种思路，一种是联结时序分类(Connectionist Temporal Classification, CTC)[1]，另一种是基于注意力机制的序列到序列的模型(Sequence-to-Sequence Model based Attention)。

2 CTC模型

CTC实质上是一种损失函数， 它带来的最大的优势就是减小了解码空间，大大加快了解码速度。CTC Loss在计算的过程中引入了空格'_'，将模型输出中的'_'和重复的字母去掉，综合成唯一的一条路径。原有框架中通常会把cca_tt, caaatt, c__att等看成不同的序列，但是他们实际对应的都是单词cat，这里明显存在着不合理，在实际计算损失的时候，就会削弱产生单词cat的概率。CTC的提出实际上解决了这个问题。CTC Loss在计算过程中，将同一label对应的多种路径的概率整合到一起后再进行梯度的计算。可以参考图示1。

CTC提出的实质就是想摆脱语言模型和发音词典，但是在实际中，没有语言模型的CTC方法仍然达不到实际应用。CTC存在着训练困难的问题[2], 直接进行训练的话，会导致模型收敛很困难，因此在实际中可能需要第一轮先用交叉熵损失进行训练，接下来再使用CTC Loss; 或者使用SortaGrad[3], 对训练的例子进行一个排序，先训练简单句，再训练复杂句。最近有研究显示[4]，CTC在以Syllable和Character为建模单元时

*Email: tianzhengkun2017@ia.ac.cn

候，性能超过基于Phoneme的模型。使用粗粒度建模单元带来的除了模型鲁棒性的提升，还有OOV问题，如何解决OOV问题，已经成为了目前一个重点的研究方向，其中必要流行的就是使用多种粒度建模单元的结合[5]。

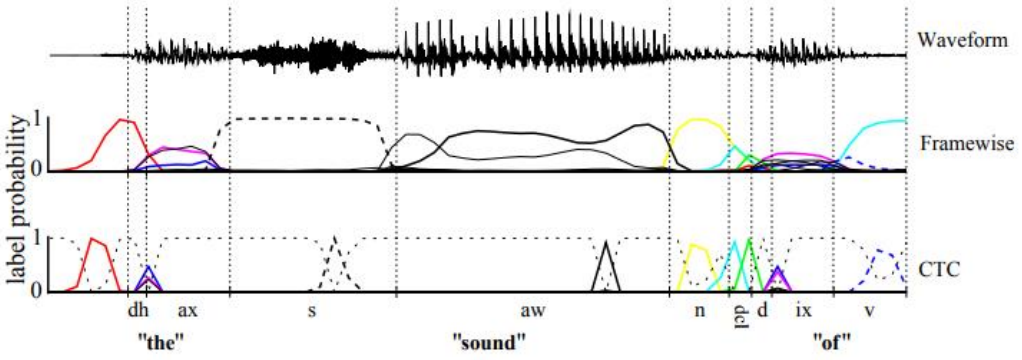

图 1: CTC图示

3 基于Attention的模型

Attention机制最先应用于机器翻译中[6]，并在机器翻译中取得了最好的效果。其主要思想就是通过编码器(Encoder)将原序列转换成一个固定长度的隐层表示，然后解码器(Decoder)再根据这个隐层表示生成解码序列，生成解码序列过程中考虑当前解码输出与隐层表示中哪一部分最相关，这部分就是注意力机制。

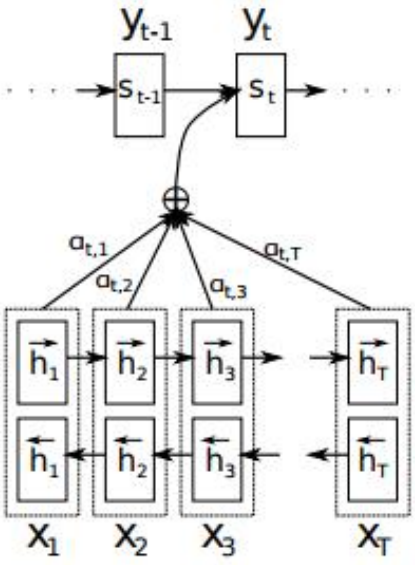

图 2: Attention模型基础

在这个模型结构中，每一个条件概率输出定义为：

$$p(y_i|y_1,\ldots,y_{i-1},X) = g(y_{i-1},s_i,c_i)$$

其中y_i表示第i时刻解码输出标记，X表示Encoder输入，y_{i-1}表示上一时刻解码输出，s_i表示i时刻的隐层状态，c_i表示上下文向量。其中s_i计算公式为

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

上下文向量c_i是Encoder输出隐变量h_i的加权和。

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_{ij}$$

其中α_{ij}即注意力权重。其计算过程如下:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ij})}$$

其中

$$e_{ij} = score(s_{i-1}, h_j)$$

上式中的e_{ij}表示注意力机制的打分结果，实际上相当于一个相关性计算，具体的分数计算有多种方式，其反映了上一时刻隐层状态s_{i-1}与向量表示h_j之间的相关性。

基本的Attention结构就是这样，接下来分析最近几年在这个结构基础上面的发展。

4 不同的Encoder结构

Encoder-Decoder结构本身就是针对序列到序列的问题提出，最容易想到的结构就是RNN及其变体(LSTM等)，目前常用的就是采用Bi-LSTM作为Encoder。由于Decoder本身对上一时刻输出的依赖，对于Decoder的改进比较难。但是可以探索不同的Encoder。LSTM对于序列问题的处理需要沿着时间步进行，这大大降低了Encoder的效率，探索更加快速的编码过程对于解决Seq2Seq问题具有重要的意义。

4.1 ConvS2S

Facebook在2017年提出ConvS2S[7]结构，把卷积结构首先引入NMT(Neutral Machine Translation)问题中，并且一度获得最好的效果。模型结构如图3所示。

由于没有时序结构，因此需要在Embedding的基础上面增加位置信息，模型中将Position Embedding(p_i)与原来的Embedding(w_i)直接进行相加，因此模型的输入序列为$e = \{e_1, \ldots, e_m\}$，其中$e_i = w_i + p_i$。Decoder在每一时刻的输入$g = \{g_1, ..., g_m\}$同样由两部分组成，分别是上一时刻输出的Word Embedding以及对应的Position Embedding。Decoder中第l个Block的输出定义为$h^l = (h_1^l, \ldots, h_n^l)$,Encoder中第$l$个Block的输出定义为$z^l = (z_1^l, \ldots, z_n^l)$。通过堆叠卷积结构，能够扩大感受野的面积，越高层结构获得的上下文信息越多。

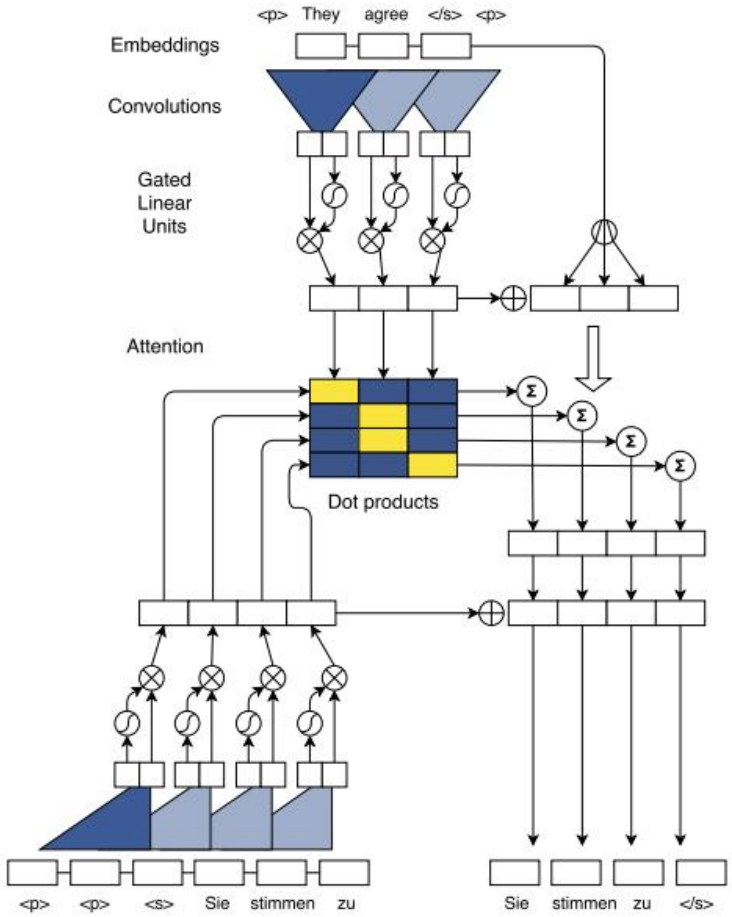

图 3: ConvS2S模型结构

模型中使用了残差连接和GLU(Gated Linear Units)。

$$h_i^l = v(W^l[h_{i-k/2}^{l-1}, \ldots, h_{i+k/2}^{l-1}] + b_w^l) + h_i^{l-1}$$

其中GLU的计算为

$$v([A, B]) = A \otimes \sigma(B)$$

不同于一般的Encoder-Decoder结构，这里的注意力机制使用了Multi-Step Attention。在Decoder的每层中都计算注意力机制分数。在解码器第l层的Attention计算公式为：

$$d_i^l = W_d^l h_i^l + b_d^l + g_i$$

$$a_{ij}^l = \frac{exp(d_i^l \cdot z_j^u)}{\sum_{t=1}^m exp(d_i^l \cdot z_t^u)}$$

$$c_i^l = \sum_{j=1}^m a_{ij}^l (z_j^u + e_j)$$

其中z_j^u表示Encoder最后哦一层的隐层状态，得到c^l之后，将其加到当前层的隐层状态h^l中，然后再输入到Decoder的第$l+1$层中，知道获得Decoder的最终隐层表示h^L。然

后就通过softmax层计算输出结果:

$$p(y_{i+1}|y_1,\ldots,y_i,x) = softmax(W_o h_i^L + b_o)$$

由于下一个时刻的输出要依赖于上一个时刻的输出，从而导致无法在整个序列上进行并行处理，这会引起训练时间过长的问题。针对这一问题，ConvS2S的做法是将CNN引入到Seq2Seq中，这样既可以处理序列变长的问题，又可以实现在序列不同位置的并行计算。RNN的另一个缺陷在于，对于一个长度为N的序列，要建立long-range dependencies，需要经过$O(n)$次运算，而对于卷积核宽度为k的多层CNN来说，则需要$O(n/k)$次运算。

4.2 Very Deep Encoder

受限于LSTM的计算速度问题，常见的Sequence-to-Sequence结构都采用的是浅层结构, 在ICASSP2017中，Yu Zhang等人[8]受Very Deep CNN在ASR任务中的优秀表现，提出使用更深层的网络来进行序列编码，代替浅层Encoder。

在论文中，作者使用了Network-in-Network(NiN)[9],Batch Normalization(BN)[10],Residual Networks(ResNets)[11]和Convolutional LSTM(ConvLSTM)[12]等方法构建模型。借鉴NiN中的1 × 1卷积，来增加网络的深度和模型的表征能力。BN和ResNets方法有助于训练更深层的结构。ConvLSTM中使用卷积操作代替LSTM内部的内积操作。

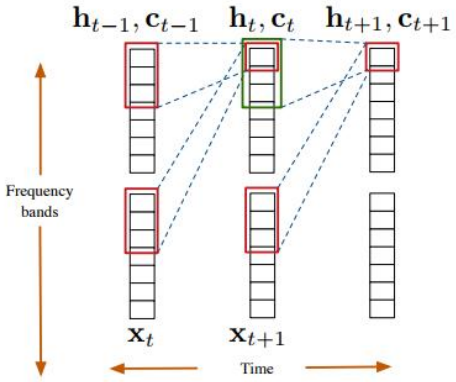

图 4: ConvLSTM结构

实验在WSJ数据集上进行，结果显示模型获得了WER为10.53。

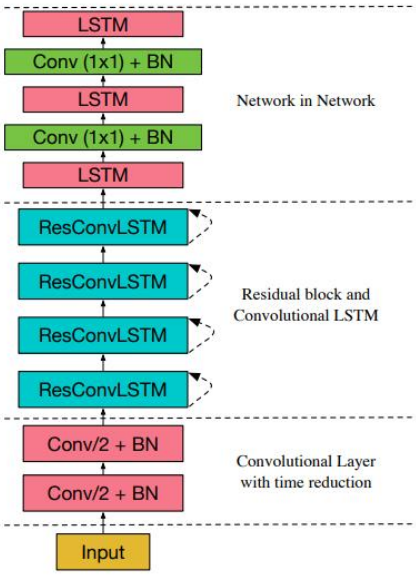

图 5: Very Deep Encoder(optimal)

Model	WER
CTC(Graves et al.,2014)	30.1
seq2seq(Bahdanau et al.,2016)	18.0
seq2seq+deep convolutional(optimal)	10.53

4.3 LAS

为了提高编码的速度，也可以使用金字塔结构的Encoder[14]。模型Encdoer每层都会将时间步减少一半。

$$h_i^j = pBLSTM(h_{i-1}^j, [h_{2i}^{j-1}, h_{2i+1}^{j-1}])$$

$$c_i = AttentionContext(s_i, h)$$

$$s_i = RNN(s_{i-1}, y_{i-1}, c_{i-1})$$

$$P(y_i|x, y_{<i}) = CharacterDistribution(s_i, c_i)$$

这里金字塔结构采用每层合并上一层相邻两个时间步或者三个都行，可以通过实验进行选择。CharacterDistribution是前馈网络结构。

5 不同的Attention机制

Encoder-Decoder结构能够获得飞速发展的一个重要原因就是采用了Attention机制。探索不同的attention机制也是一个重要的研究方向。

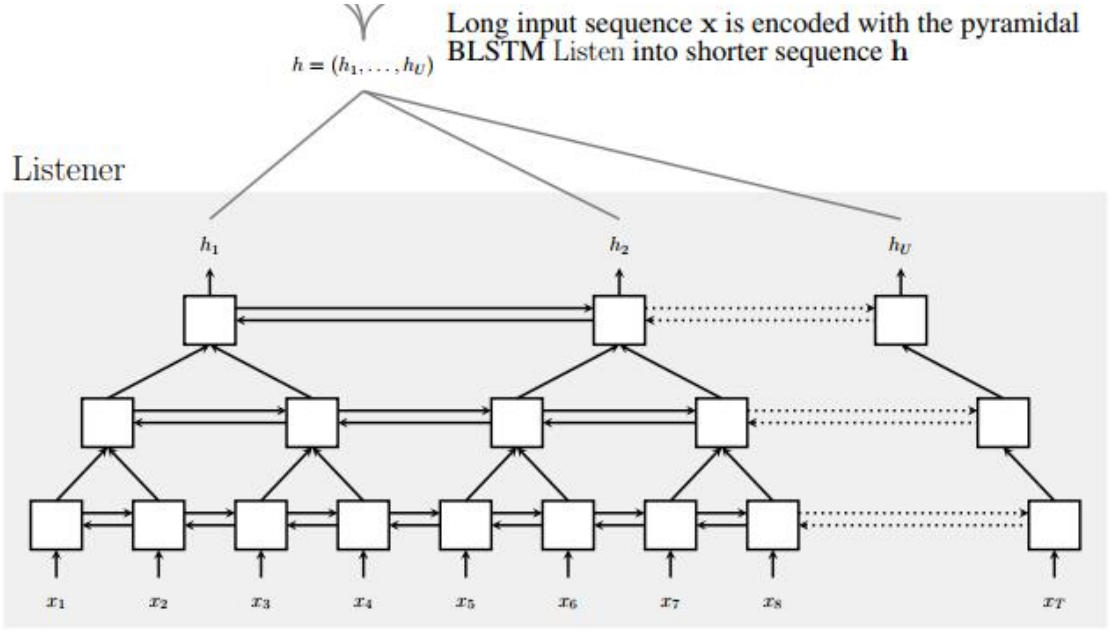

图 6: LAS Listener(Encoder)结构

5.1 不同的打分机制

Attention的核心思想就是计算当前要解码序列需要的输入信息与上下文信息之间的相关性。相关性的计算就是:

$$e_{ij} = score(s_{i-1}, h_j)$$

其计算方式有多种:

(1)计算余弦距离

LAS模型就是采用了这种方式，由于隐层状态s_{i-1}和上下文向量h_j并不在同一个特种空间中，要计算余弦距离，首先应该将他们转换到同一个空间中，因此文章中首先将他们各自输入到一个前馈网络中，计算网络输出的余弦距离，即实现了相关性计算。不同于其他模型，LAS中使用的是状态s_i。

$$e_{ij} =< \Phi(s_i), \Phi(h_j) >$$

其中$\Phi(\cdot)$即表示特征变换网络，其实前馈网络结构。

(2)直接进行计算[13]

这种方式没有考虑两个向量位于不同的特征空间，直接计算打分结果。常见的有以下几种方式:

dot:

$$score(h_t, \overline{h}_s) = h_t^T \overline{h}_s$$

general:

$$score(h_t, \overline{h}_s) = h_t^T W_a \overline{h}_s$$

concat:

$$score(h_t, \overline{h}_s) = v_a^T tanh(W_a[h_t; \overline{h}_s])$$

(3)打分过程中考虑上一时刻注意力权重[16]
那么打分过程变成$score(s_{u-1}, \alpha_{u-1}, h_t)$,具体计算公式如下：

$$e_{u,t} = V^T tanh(U s_{u-1} + W h_t + V f_{ut} + b)$$

其中$f_{ut} = F * \alpha_{u-1}$,$*$表示卷积操作。$U, W, V, F, b, v$都是被学习的参数矩阵。

5.2 Hard-Attention

Hard-Attention[15]是在Image Caption Generation任务中提出的。常见的注意力机制是经过softmax层输出之后有不同的权重。α是一个向量，里面元素都是范围在$[0, 1]$之间的小数，和为1。而采用Hard-Attention之后，注意力向量α中的元素只有一个是1，其余的都是0，也就是在每一个时间步，模型只关注一个位置。向量α是One-hot形式。

$$\hat{z}_t = \sum_i s_{t,i} h_i$$

其中$s_{t,i}$表示在第t时刻的attention是否关注位置i。

5.3 Soft-Attention

常见的注意力机制都是Soft-Attention。即注意力向量α中的不同位置的权重值不同，这样的soft attention是光滑的且是可微的。[15]中还对注意力机制进行了微调。

$$\hat{z}_t = \beta \sum_i \alpha_{t,i} h_i$$

其中$\beta = \sigma(f_\beta(s_{u-1}))$，用来调节上下文向量在LSTM中的比重。

在上面所讲的注意力权重$\alpha_{u,t}$都是标量权重，这意味着向量h_u中的每个元素都给了相同的权重。[16]中考虑为了使得h_u中的元素更加具有区分性，可以考虑把权重换成矢量$\alpha_{\mathbf{u},\mathbf{t}}$。
那么：

$$\mathbf{e_{u,t}} = tanh(U s_{u-1} + W h_t + V f_{ut} + b)$$

$$\alpha_{u,t,j} = \frac{exp(e_{u,t,j})}{\sum_{t'=u-\tau}^{u+\tau} exp(e_{u,t',j})}, j = 1, \ldots, n$$

5.4 Global-Attention

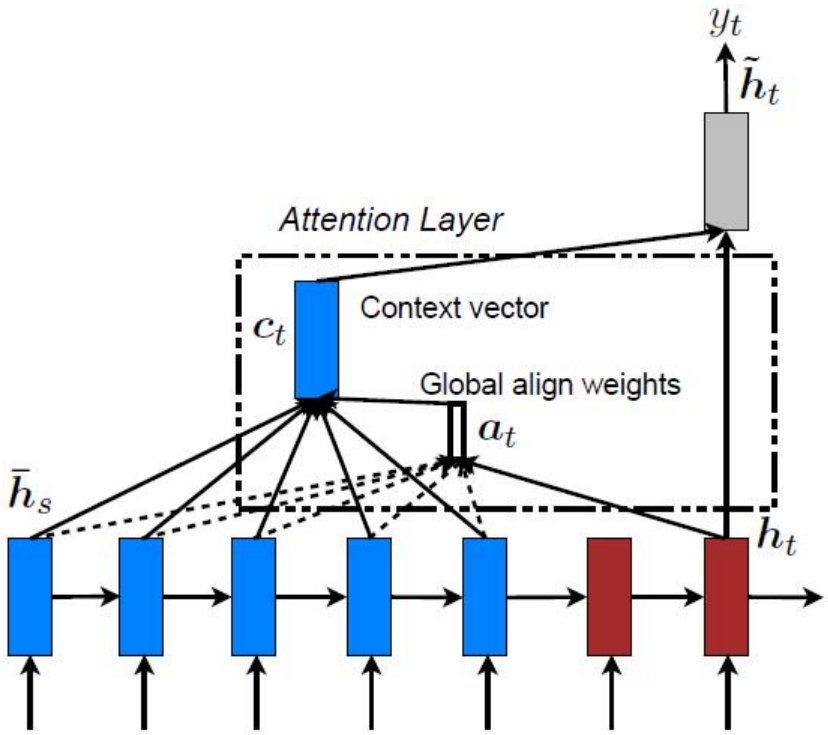

图 7: Global-Attention

global attention[13]就是在Decoder计算注意力权重的每一时刻都考虑全部的上下文信息，赋予不同位置的上下文信息不同的权重，并加权求和。其计算方式就是与通常的计算方式一致。

5.5 Local-Attention

Global方式关注全局上下文信息，一方面这样有很大的计算量，另一方面在像语音识别这样的任务中，由于两种序列时序一致，注意力主要集中的位置是在时序对应的位置，其他位置的不需要特别关注。并且采用local-attention有助于推进Sequence-to-Sequence这里的上下文向量的计算每次都只focus窗口$[p_t - D, p_t + D]$内的$2D+1$个source hidden states。D是经验选择的，文章中使用的是10。

这个方法像hard-attention,但是local-attention是可微的，因此更加容易训练。[13]提出了local-m(local monotonic alignment)和local-p(local predictive alignment)两种local attention计算方式。p_t是source position index，可以理解为attention关注的焦点。在local-m中，

$$p_t = t$$

在local-p中,

$$p_t = S \cdot sigmoid(v_p^T tanh(W_p h^t))$$

其中W_p和v_p都是模型的参数，S是source sentence的长度，那么$p_t \in [0, S]$。则权重计算如下:

$$a_t s = score(h_t, \overline{h}_s) exp(-\frac{(s - p_t)^2}{2\sigma^2})$$

我们很容易看出，距离中心p_t越远，其位置上的source hidden state对应的权重则被压缩的越厉害。

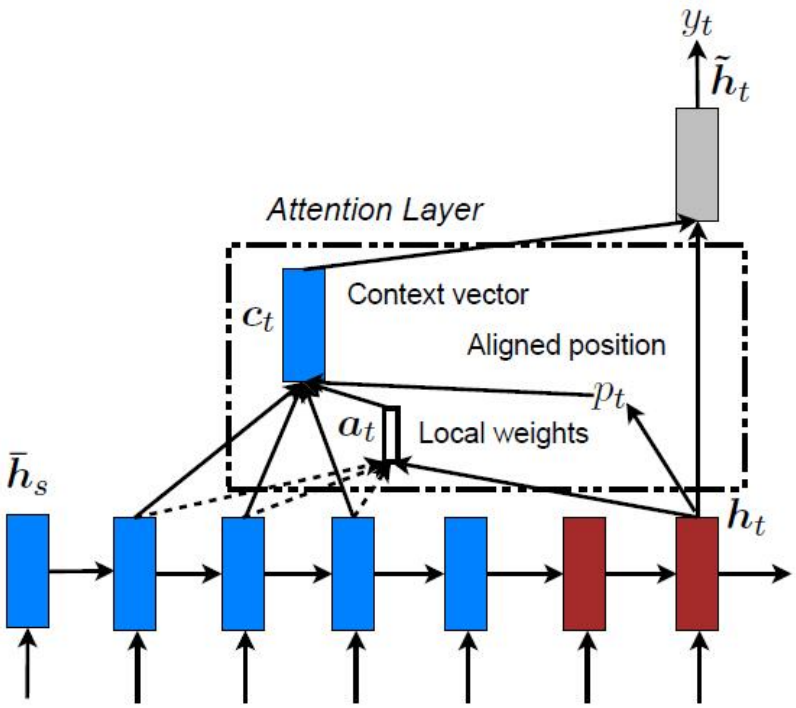

图 8: Local-Attention

5.6 Multi-Head Attention

Multi-Head Attention(MHA)最初出现在Google的工作[17, 18]中。MHA在传统注意力机制的基础上扩展了多个head，每个head能够生成不同的注意力分布。这个允许

每个head在focus编码器输出的时候，可以扮演不同的角色。这种方式能够帮助解码器更容易的从编码输出中检索出所需要的信息。传统的single-head attention更加依赖于编码器提供清晰的隐层表示以使得Decoder能够挑选出重要的信息。文章观察到MHA趋向于分配一个head去关注语句的开头，语句的开头往往包含大部分的背景噪声。因此可以猜测，在编码器输出不理想的情况下，MHA结构能够从噪声中区分语音信息。 为了

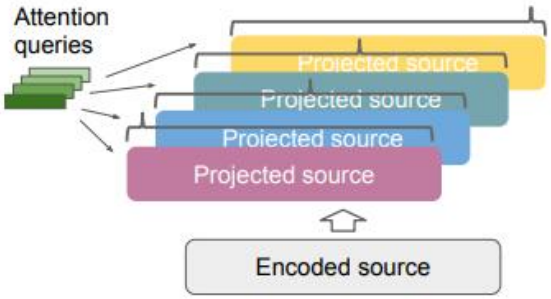

图 9: Multi-Head Attention

确保MHA在训练过程中确定能够关注到不同的位置，有文章研究在损失函数中增加正则项，以确保多个MHA之间存在差异[25]。

6 CTC与Attention混合结构

纯Attention方法虽然取得了不错的效果，但是在训练过程中存在着明显的收敛速度慢，震荡幅度大等问题。这很大程度上在于一开始Attention注意范围太广，难以收敛。因此有文章[19]提出使用CTC辅助attention模型的训练，实验表明这种方法能够极大的提高模型的收敛速度。模型结构如图10所示。

模型成功的关键在于在损失函数中引入CTC Loss。

$$Loss = \lambda L_{CTC} + (1 - \lambda) L_{Attention}$$

也可以理解为CTC帮助模型的Encoder获得了更加清晰的隐层表示。实验表明没有CTC辅助训练的结构在迭代9个epoch后仍然没有收敛，但是在CTC辅助训练的情况下，模型在5个epoch的时候已经收敛了。

Attention结构在采用大的建模单元时候往往能够获得很好的性能，但是建模单元越大，OOV问题也就越严峻，可以采用CTC与Attention共同解码来实现[20]。模型中CTC部分使用Charcter-Level的建模单元，Attention部分使用Word-Level的建模单元。两种模型结构公用一个Encoder部分。模型结构如图11所示。其实现的关键在于，

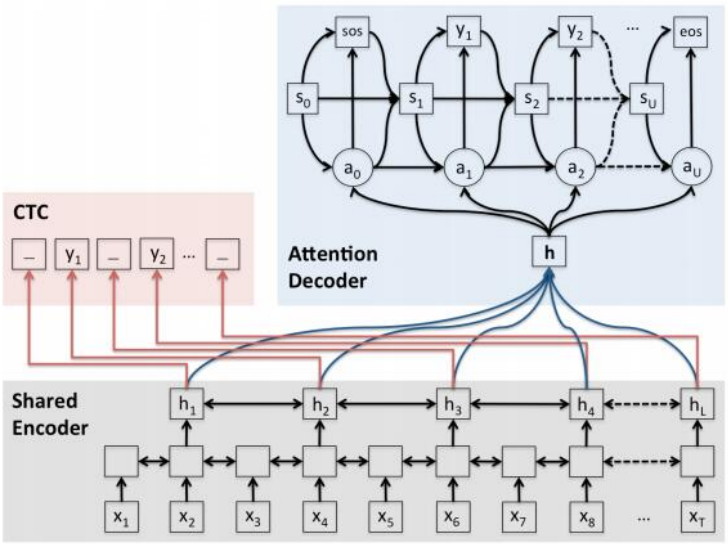

图 10: CTC与Attention混合结构

采用了混合Loss:

$$Loss = (1-\lambda)L_{charcter} + \lambda L_{word}$$

在解码阶段，如果对应于Attention的Decoder中非OOV的词汇，则使用对应的输出。如果最大概率的输出是OOV标记，则使用CTC中的结果进行代替。为了实现混合解码，CTC部分除了增加blank，还应该增加一个词边界标记wb。解码过程如图12所示。

7 Transformer(Self-Attention)

Transformer[18]是最初在NMT领域中获得了巨大的成功，到目前为止，最好的结果还是Transformer创造的，并且目前有很多工作尝试着将Transformer引入到ASR领域中[20, 21]。如图13所示。

Transformer与ConvS2S几乎同时提出，解决的问题也是相同的，即为了提高Encoder的并行度。文中最关键的点就是Self-Attention和Multi-Head Attention两种机制。Self-Attention是每个词都要和所有的词计算Attention，所以不管他们中间有多么长的距离，最大的路径长度也都是1，可以捕获长距离的依赖关系。Multi-Head Attention可以看成是Attention的Ensemble版本，不同head学习不同的子空间语义（关注编码器输出的不同部分）。

文中的两个关键公式是:

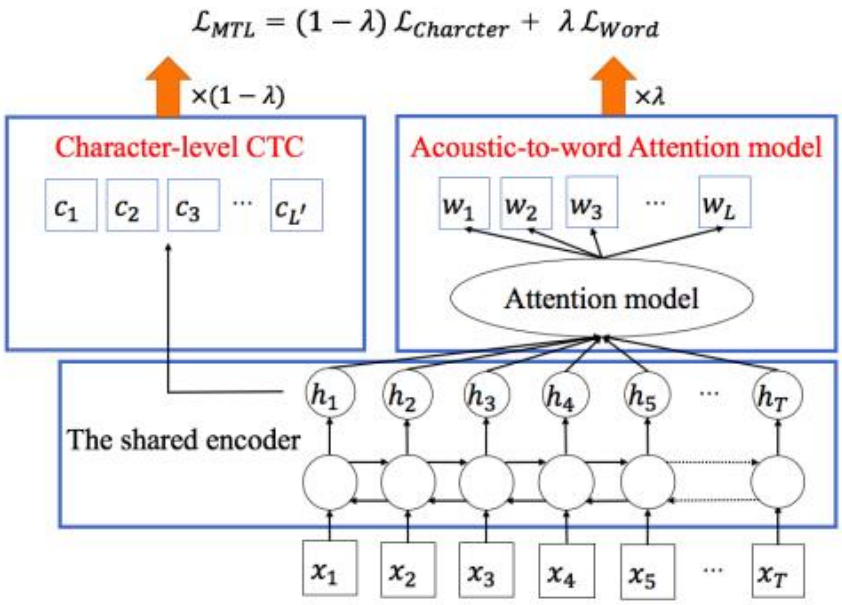

图 11: CTC与Attention混合解码结构

Self-Attention:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-Head Attention:

$$MultiHead(Q,K,V) = Concat(head_1,\ldots,head_h)W^o$$

$$head_i = Attention(QW_i^Q, K_i^Q, V_i^Q)$$

其中Q表示query，K表示Key,V表示value，在self-attention时候，$Q = K = V = inputs$。假设Q,K,V都是$n * d$的矩阵，表示一共有n个词，每个词用维度为d的向量表示，则$softmax(\frac{QK^T}{\sqrt{d_k}})$是一个$n * n$的矩阵，矩阵中第i行第j列元素表示第i个词和第j个词之间的相关性，$Attention(Q,K,V)$仍然是一个$n * d$的矩阵，其中每个元素都获得了全部的上下文信息，因此建模了长距离依赖。$\sqrt{d_k}$是一个尺度因子。

像ConvS2S一样，由于没有RNN结构，因此需要引入位置信息，在原embedding表示上嵌入位置编码。

$$PE_(pos, 2i) = sin(pos/10000^{2i/d_{model}})$$

$$PE_(pos, 2i+1) = cos(pos/10000^{2i/d_{model}})$$

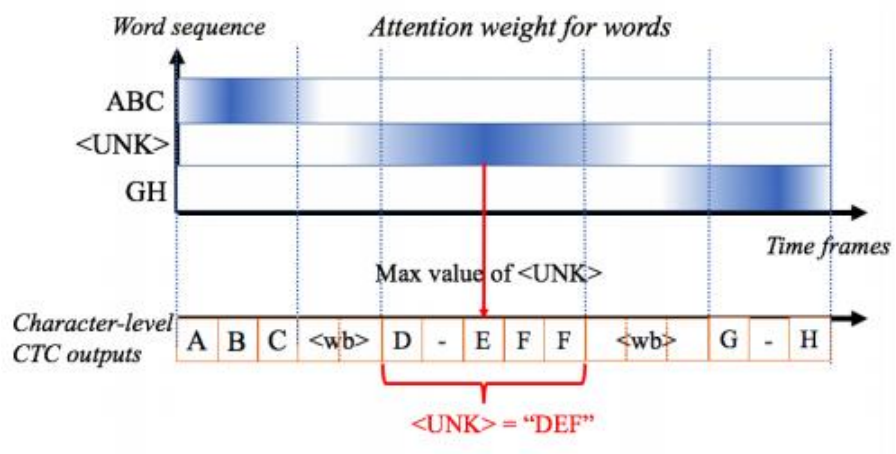

图 12: CTC与Attention混合解码过程

之所以采用这种位置编码方式是因为文章认为这种编码方式用于模型学习词之间的相对位置关系，pos表示位置，i表示维度。

模型中还采用了残差连接结构，以及layer-normalization(LN)。

$$Output = LN(x + sublayer(x))$$

其中$sublayer$结构表示multi-head部分的输出。Output输出到Encoder Block的第二部分Position-wise FC层。对于一个$n * d$的矩阵的每一行进行操作（相当于把矩阵按行铺平，接一个FC），同一层的不同行FC层用一样的参数，不同层用不同的参数（对于全连接的节点数目，先从512变为2048，然后再缩小为512）。

Decoder的输出也是$n * d$的矩阵，假设已经翻译出来了k个词，向量的维度还是d,在self-attention阶段，需要添加mask来遮住未来的信息。

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

8 Attention模型的训练技巧

目前至少证明有以下几个技巧对于训练基于Attention的模型有帮助。

（1）采用大的建模单元，sub-word, word等，这样的建模单元更加稳定并且有助于语言建模。

（2）引入Scheduled Sampling[22]，解决Attention模型中训练和推理过程的不匹配问题。

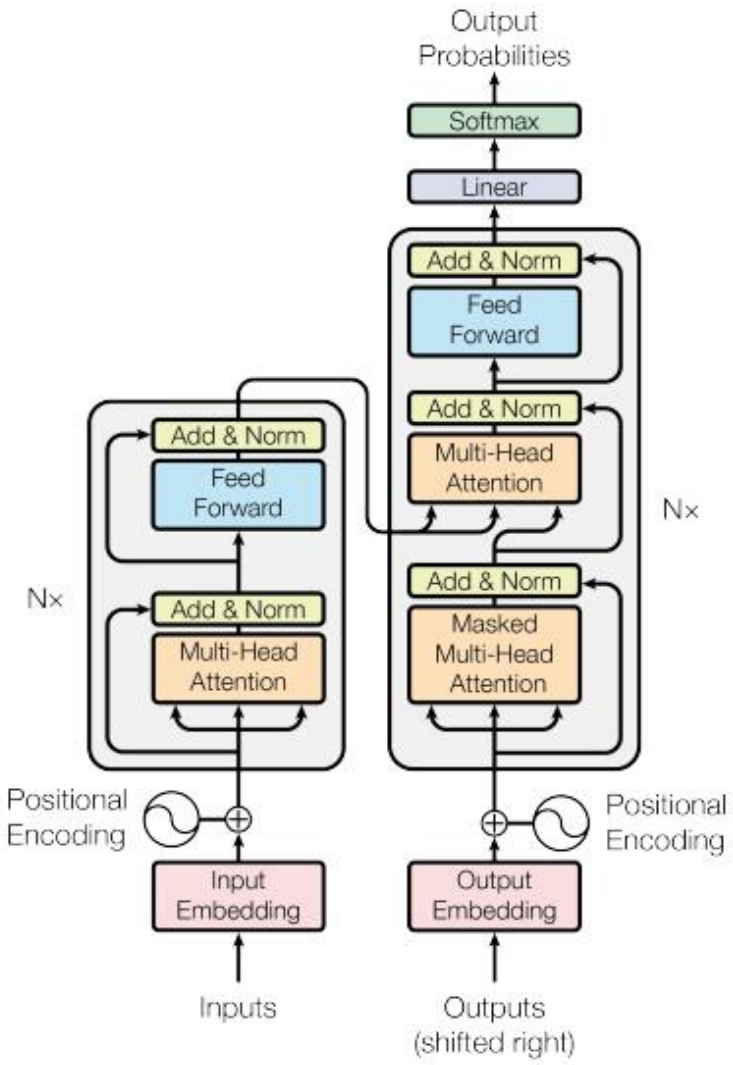

图 13: Transformer结构

（3）Multi-Head Attention，MHA方法能够生成不同的注意力分布，在Decoder解码过程中，关注上下文向量的不同方面。

（4）采用Label smoothing[23]方法来避免模型对于自己的预测结果过于自信。

（5）整合额外的语言模型（在更大的文本语料上训练的），因为Attention模型的受限于语料规模，没有语言模型的结构还是不能与常见的模型相匹敌。

（6）使用最小化词错误率[24]的方式进行区分性训练。

（7）模型除了训练和推理过程中的不匹配问题，还存在损失函数之间的不匹配，

训练时通常使用CE，而在评价阶段使用WER等，[25, 26]中引入强化学习中的Policy Gradient方法，直接优化WER，来进行端到端的训练。

9 关于Attention的几点思考

根据目前Attention在语音识别领域的发展，结合自己的理解，列出以下几个需要考虑的问题。

Attention模型最初应用于NMT领域，然后我们直接将其应用到ASR领域，虽然ASR也是序列到序列的转换问题，但是ASR和NMT问题之间还是存在着明显的区别。首先是语序问题，NMT是在两种语言直接的编码解码转换，两种语言的语序存在着明显的区别，然而ASR问题语音信号和文本序列之间存在着明显的时序对应关系，是不是应该考虑如何在模型中应用这种时序对应关系帮助我们进行模型训练。另一个明显的区别在于词边界问题，NMT问题是文本之间的转录，文本中存在着明显的词边界，也就是我们可以认为NMT问题中的Encoder能够提供更加清晰的隐层表示，对于ASR问题，怎么获取更加清晰和更加有区分性的隐层表示也是一个值得考虑的问题。

相比于传统AM,LM,发音词典独立的模型结构，Attention方法在建模语言之间的关联关系方面存在着缺陷，有没有方法能够在不增加整体语音语料，同时也不外加语言模型的情况下，单独提高模型对于表征单词之间联系的能力。

随着建模单元的逐渐增加，怎么更加高效的解决attention方法面临的OOV问题，也是思考的方向之一。

目前CTC和Attention方法可能都不是最优的端到端建模的方法，探索新的建模方法也是未来的重点之一。

附录

这部分介绍目前我所了解到用于端到端语音识别的资源。

(1)Transformer
Tensorflow: https://github.com/tensorflow/tensor2tensor
Pytorch: https://github.com/jadore801120/attention-is-all-you-need-pytorch

(2)Joint CTC-Attention
Pytorch and Chainer: https://github.com/kan-bayashi/espnet

(3)Listen, Attend and Spell:

Tensorflow:

Nabu: https://github.com/vrenkens/nabu

Pytorch:https://github.com/XenderLiu/Listen-Attend-and-Spell-Pytorch

(4)ConvS2S:

Torch(Lua): https://github.com/facebookresearch/fairseq

(5)Seq2Seq:

Pytorch: https://github.com/IBM/pytorch-seq2seq

Tensorflow:https://github.com/google/seq2seq

参考文献

[1] Graves, Alex, and F. Gomez. "Connectionist temporal classification:labelling unsegmented sequence data with recurrent neural networks." International Conference on Machine Learning ACM, 2006:369-376.

[2] Yu, Dong, and J. Li. "Recent Progresses in Deep Learning Based Acoustic Models." IEEE/CAA Journal of Automatica Sinica 4.3(2017):396-409.

[3] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos et al., "Deep speech 2: End-to-end speech recognition in English and Mandarin," arXiv preprint arXiv:1512.02595, 2015.

[4] Wei Zou, Dongwei Jiang, Shuaijiang Zhao, Xiangang Li. A comparable study of modeling units for end-to-end Mandarin speech recognition. http://arXiv.org/1805.03832.pdf

[5] Li, Jinyu, et al. "Advancing Acoustic-to-Word CTC Model." (2018).

[6] Bahdanau, Dzmitry, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." Computer Science (2014).

[7] MLAGehring, Jonas, et al. "Convolutional Sequence to Sequence Learning." (2017).

[8] Zhang, Yu, W. Chan, and N. Jaitly. "Very Deep Convolutional Networks for End-to-End Speech Recognition." (2017):4845-4849.

[9] M. Lin, Q. Chen, and S. Yan, "Network in network," in ICLR, 2013.

[10] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in ICML, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in CVPR, 2016

[12] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting,” in NIPS, 2015.

[13] Luong, Minh Thang, H. Pham, and C. D. Manning. ”Effective Approaches to Attention-based Neural Machine Translation.” Computer Science (2015).

[14] Chan, William, et al. ”Listen, Attend and Spell.” Computer Science (2015).

[15] Xu, Kelvin, et al. ”Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.” Computer Science (2015):2048-2057.

[16] Das, Amit, et al. ”Advancing Connectionist Temporal Classification With Attention Modeling.” (2018).

[17] Chiu, Chung Cheng, et al. ”State-of-the-art Speech Recognition With Sequence-to-Sequence Models.” (2017).

[18] Vaswani, Ashish, et al. ”Attention Is All You Need.” (2017).

[19] Kim, Suyoun, Takaaki Hori, and Shinji Watanabe. ”Joint CTC-attention based end-to-end speech recognition using multi-task learning.” international conference on acoustics, speech, and signal processing (2017): 4835-4839.

[20] Sei Ueno et al.,acoustic-to-word attention-based model complemented with character-level ctc-based model, ICASSP2018, http://sap.ist.i.kyoto-u.ac.jp/lab/bib/intl/UEN-ICASSP18.pdf

[21] Shiyu Zhou, et al. ”Syllable-Based Sequence-to-Sequence Speech Recognition with the Transformer in Mandarin Chinese.” (2018).

[22] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks,” in Advances in Neural Information Processing Systems, 2015, pp. 1171–1179.

[23] A. Kannan, Y. Wu, P. Nguyen, T. N. Sainath, Z. Chen, and R. Prabhavalkar, “An analysis of incorporating an external language model into a sequence-to-sequence model,” in Proc. ICASSP, 2018.

[24] R. Prabhavalkar, T. N. Sainath, Y. Wu, P. Nguyen, Z. Chen, C.-C. Chiu, and A. Kannan, "Minimum word error rate training for attention-based sequence-to-sequence models," in Proc. ICASSP, 2018.

[25] Lin, Zhouhan, et al. "A STRUCTURED SELF-ATTENTIVE SENTENCE EMBEDDING." international conference on learning representations (2017).

[26] Tjandra, Andros, S. Sakti, and S. Nakamura. "Sequence-to-Sequence ASR Optimization via Reinforcement Learning." (2017).

[27] Sequence traning of encoder-decoder model using policy gradient for end-to-end speech recognition