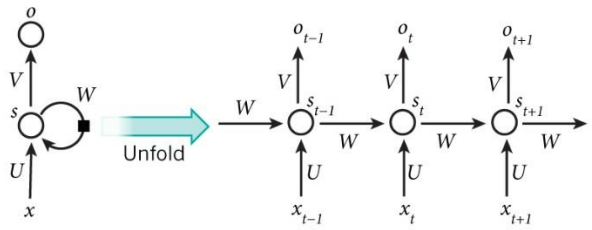
一．基础知识（大家需要提前学习这些基础知识）

1. RNN



RNN 输入是某个前后有关联的序列信息

2. LSTM

RNN 用反向传播算法进行训练，会有梯度爆炸或者梯度消失的问题，所有有了长短记忆时网络。

3. GRU

LSTM 的一个变形，当然作者并没有说出为什么 GRU 用代替 LSTM。

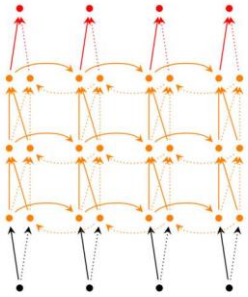
http://blog.csdn.net/meanme/article/details/48845793 这是网络一个关于 LSTM 与 GRU 比较的博客，结论是两者差别不大

4．双向神经网络

传统 RNN 用于文本时，可以理解为现在之所以为这个词跟前面出现的词有关，但实际中现在的词的出现跟后面出现的词也有关，就此有了双向 RNN。

5．深层 RNN

人们由于传统 ANN 的思想，觉得深层 RNN 一定是这样的



但其实还有三种深层的方式，一是从输入层到隐藏层的深层，二是从上一时刻的隐藏层到这一时刻的隐藏层的深层，三是从隐藏层到输入层的深层。具体方式都是在两层之间添加一些非线性中间层，比如 sigmod，tanth 层。具体可以看论文 How to construct deep recurrent neural networks

在这篇论文中将我们传统认为的深层 RNN 叫做 Stack of Hidden States，将隐藏层到输出层的深层 RNN 叫做 hidden-output（后面会用到这两个词）

之所以采取这样的深层，论文中的解释是，比如 input 到 hidden，可以更好的学习到输入的这些抽象特征的关系。没有具体的理论支持，但结果确实提高了一些。

二. 结合 Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation 与 sequence-to-sequence-learning-with-neural-networks

应用 Seq2seq 是一个框架，目前在机器翻译、对话系统、文本摘要、文本生成中均有应用

这篇论文提出了这个模型，在英语到法语翻译上实验。seq2seq 由两个 RNN 组成。第一个 RNN 用于将一个英文句子编码为一个固定维数的向量，第二个 RNN 根据这个向量解码出对应的法语。

第一个 RNN 用于学习英语语言，用的是 GRU 模型（GRU 模型也是这篇论文提出的），具体学习过程就是预测的过程，就是大家学 RNN 看博客时介绍的那种。这里所谓的将一个句子编码为 fixed-vector 其实就是 GRU 隐藏层最后时刻的向量 Ht 进行一个处理，比如 c=tanh（V*Ht）。论文中没有说关于 V 的情况，我在网上的一个博客上看到，一般这种情况 V 可以取元素值为 1 的对角矩阵，如果网络是个 Stack of Hidden States 深层网络，比如有四层，那这里的 fixed-vector 就是 4 个（每层一个）。为了简单讲解，这里我们只讲一层的情况。

我们用 c 来表示这个 fixed-vector，下面来看看他是怎么应用的（关于 GRU 里面重置门，更新门的前向传播公式就不在这里一一列出的）。
标准的 GRU 重置门的前向传播公式是

$$r_j = \sigma\left(\left[\mathbf{W}_r e(\mathbf{x}_t)\right]_j + \left[\mathbf{U}_r \mathbf{h}_{\langle t-1\rangle}\right]_j\right).$$

$$r'_j = \sigma\left(\left[\mathbf{W'}_r e(\mathbf{y}_{t-1})\right]_j + \left[\mathbf{U'}_r \mathbf{h'}_{\langle t-1\rangle}\right]_j + \left[\mathbf{C}_r \mathbf{c}\right]_j\right)$$

这里 c 的用处就是在每个前向传播公式中都加上 c，当然这里 Cr 是个需要训练的参数。

这里就是还有三个疑问，一是解码的 RNN 怎么确定第一个法语单词，毕竟万事开头难；二是解码的 RNN 的输出如何确定输出哪个法语单词；三是这个句子怎么才算结束。因为中间的训练就是 GRU 的那一套，大家现在还不懂的大概就只有这些了。这前两个问题其实可以算作一个问题，就是输出哪个法语单词。

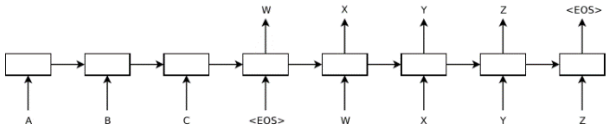
第一个问题，作者以非常简单的方式解决了，要确定第一个法语单词的输出，首先要知道第一个隐藏层的值是多少 $\mathbf{h'}^{\langle 0\rangle} = \tanh\left(\mathbf{V'c}\right)$ c 是 fixed-vector 这里的 V 就是个参数。

第二个问题，本文用一系列的 01 向量表示句子中的每个单词（比如，词典中有 3 个词，句子为 I love you ，那这个句子表示为(1,0,0),(0,1,0),(0,0,1)）。作者在这里采用了 hidden-output，即在隐藏层和输出层之间加一些非线性计算的层。 $\mathbf{s'}^{\langle t\rangle} = \mathbf{O}_h \mathbf{h'}^{\langle t\rangle} + \mathbf{O}_y \mathbf{y}_{t-1} + \mathbf{O}_c \mathbf{c}.$

这里的 s′维度是词典维度的两倍， $s_i^{\langle t\rangle} = \max\left\{s'^{\langle t\rangle}_{2i-1}, s'^{\langle t\rangle}_{2i}\right\}$ ，这里的结构是 maxout，类似于 cnn 里面的池化层，具体的理论还不是很懂。最后的输出就是一个归一化操作

$$p(y_{t,j}=1 \mid \mathbf{y}_{t-1},\ldots,\mathbf{y}_1,X)=\frac{\exp\left(\mathbf{g}_j\mathbf{s}_{\langle t\rangle}\right)}{\sum_{j'=1}^{K}\exp\left(\mathbf{g}_{j'}\mathbf{s}_{\langle t\rangle}\right)},$$ 。然后可以选择最大的概率作为这里的输出。

第三个问题，生成句子如何结束，其实是个很简单的问题。我们在训练编码 RNN 和解码 RNN 时，都在各自的词库加上一个词，比如“EOS”，把它作为句子的结束符。



，解码句子 EOS 概率最大，那就结束。

关于句子选择还有个小技巧，比如每次我选择前 k 个（一般取 2）最大的词，直接生成 n 条句子结束。每个单词被选择都是有一个概率，那么整个句子的概率为 $p(y_1,\ldots,y_{T'}|x_1,\ldots,x_T)=\prod_{t=1}^{T'}p(y_t|v,y_1,\ldots,y_{t-1})$ 。我们再选择里面最大概率的句子好了。这里也可以看出前面归一化操作的用处了，为了这里便于在同一个标准上进行比较。

思考：编码解码的思想很好，seq2seq 是一个框架，很多人基于整个框架进行改进，当然也也很多水论文，这个思想可以借鉴。

三. Neural Machine Translation by Jointly Learning to Alignand Translate

主要的想法是，标准的 seq2seq 把句子表示为 fixed-vector，并在之后的解码中运用这个 vector，那对于每个解码出的单词，编码句子中的单词对他的贡献一样（原理上分析，这里不考虑 RNN 训练过程中的损失）。这篇论文加入了注意力，就是每个解码出的单词与编码句子中的哪个单词最想关。

关于编码训练模型，本文中采取的是双向 RNN，关于隐藏向量的表示

$$forward\,hidden\,states\,(\overrightarrow{h}_1,\cdots,\overrightarrow{h}_{T_x}).$$

$$hidden\,states\,(\overleftarrow{h}_1,\cdots,\overleftarrow{h}_{T_x}).$$

$$h_j=\left[\overrightarrow{h}_j^{\top};\overleftarrow{h}_j^{\top}\right]^{\top}$$

下面详细介绍解码模型

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

这里 s 是隐藏层的值，f 是非线性函数，或者表示一个记忆单元（比如有 reset，update，这里都用 f 表示）。上文中这里是 c，是一个通用的 fixed-vector，这里是 ci，是更加与第 i 个词匹配的向量。重点在于这个向量是如何得到的。Ci 的计算跟一系列“注释”$(h_1,\cdots,h_{T_x})$有

关。hi，表示 input sentence 全部的信息，但更加注重第 i 个词周围的信息。看上去很高级，不知道如何计算，其实很简单，直接把编码 RNN 的隐藏层的值拿来用就 ok 了！

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

那么 ci 是如何计算出来的呢？

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

那α_{ij}是怎么计算的呢？

Tx 是 input sentence 的单词的长度。

那e_{ij}是如何计算出来的呢？ $e_{ij} = a(s_{i-1}, h_j)$

s_{i-1}是前一个时刻隐藏层的值，a 是一个前向神经网络（这里就是对s_{i-1}，h_j的一系列计算）。

那现在就还有一个问题了，一是 a 这个神经网络是怎么样的

a 的定义为 $a(s_{i-1}, h_j) = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$，接下来就是训练参数。

思考：深度学习的论文，可解释性真的不强，得到的主要是 idea，比如这个注意力模型，真实蛮给力的。不过正是由于这种可解释性不强的玄学特性，当我们有个 idea 的时候，但不能直接客观的对他建模，如果用上 deep learning 的技术，或许可以得到不一样的结果。