seq2seq

seq2seq 现在已经成为机器翻译、文本摘要、对话聊天等工作的重要模型。

1. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

该篇文章较早地使用了 seq2seq 模型来解决机器翻译的问题。

这篇文章提出了一种新的模型，叫做 RNN Encoder-Decoder，该模型由两个 RNN 组成，其中 Encoder 将输入序列表示为固定长度的向量，Decoder 将向量表示转化为目标序列。

RNN Encoder-Decoder

文章首先介绍了下 RNN 的有关内容。

RNN

RNN 包含一个隐藏状态 $\mathbf{h}$ 和可选输出 $\mathbf{y}$，输入为可变长度序列 $\mathbf{x}=(x_1,\cdots,x_T)$，在时刻 t，隐藏状态 $\mathbf{h}_{\langle t\rangle}$ 为：

$$\mathbf{h}_{\langle t\rangle}=f(\mathbf{h}_{\langle t-1\rangle},x_t)$$

其中，f 为非线性激活函数，比如 element-wise logistic sigmoid 函数或 LSTM 单元。RNN 可以通过训练来预测序列中下一个 symbol 的概率分布，时刻 t 的输出为条件分布 $p(x_t|x_{t-1},\cdots,x_1)$。比如，使用 softmax 激活函数可以得到 multinomial 分布：

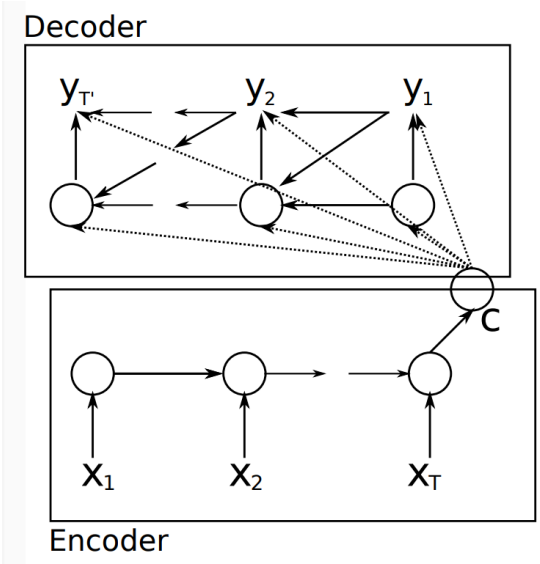
$$p(x_{t,j}=1|x_{t-1},\cdots,x_1)=\frac{\exp(\mathbf{w}_j\mathbf{h}_{\langle t\rangle})}{\sum_{j'=1}^{K}\exp(\mathbf{w}_{j'}\mathbf{h}_{\langle t\rangle})}$$

其中，$\mathbf{w}_j$ 为权重矩阵 $\mathbf{W}$ 的行。将这些概率组合起来，可以得到序列 $\mathbf{x}$ 的概率为：

$$p(\mathbf{x})=\prod_{t=1}^{T}p(x_t|x_{t-1},\cdots,x_1)$$

RNN Encoder-Decoder

模型的整体框架为：



从概率的角度上讲，RNN Encoder-Decoder 就是学习两个变长序列的条件分布 $p(y_1, \cdots, y_{T'}|x_1, \cdots, x_T)$，其中 T 和 T' 可能是不同的。

encoder 是一个 RNN，依次读取序列 $\mathbf{x}$，隐藏状态根据 $\mathbf{h}_{\langle t \rangle} = f(\mathbf{h}_{\langle t-1 \rangle}, x_t)$ 更新，最后得到整个输入序列的隐藏状态 $\mathbf{c}$。

decoder 也是一个 RNN，给定隐藏状态 $\mathbf{h}_{\langle t \rangle}$，预测 y_t 的概率。时刻 t 的隐藏状态更新为：

$$\mathbf{h}_{\langle t \rangle} = f(\mathbf{h}_{\langle t-1 \rangle}, y_{t-1}, \mathbf{c})$$

该时刻的输出概率为：

$$P(y_t|y_{t-1}, y_{t-2}, \cdots, y_1, \mathbf{c}) = g(\mathbf{h}_{\langle t \rangle}, y_{t-1}, \mathbf{c})$$

模型训练时则去最大化给定输入序列 $\mathbf{x}$ 时输出序列为 $\mathbf{y}$ 的条件概率：

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(\mathbf{y}_n|\mathbf{x}_n)$$

模型训练好之后，一种用法是给定输入序列来产生目标序列，另一种用法是对给定输入-输出序列打分，分数就是 $p_\theta(\mathbf{y}|\mathbf{x})$ 的概率。

该文也提出了一种类似于 LSTM 的 GRU 结构，具有比 LSTM 更少的参数，更不容易过拟合，这里不作深究。

2. Sequence to Sequence Learning with Neural Networks

该篇文章真正提出了 seq2seq 的思想。

DNN 在许多棘手的问题处理上取得了瞩目的成绩，比如语音识别等。如有好的学习策略，DNN 就能够在监督和反向传播算法下训练出很好的参数，解决许多计算上复杂的问题。

DNN 的一大缺陷是它只能处理输入、输出向量维度是定长的情形。有很多重要的问题，序列的长度是不固定的，这也限制了 DNN 的使用。文章指出，LSTM 可以解决一般的 seq-seq 问题。主要思路是，用一个 LSTM 读取输入序列，one timestep at a time，得到固定长度的向量表示，用另一个 LSTM 来获得输出序列，如下图所示：

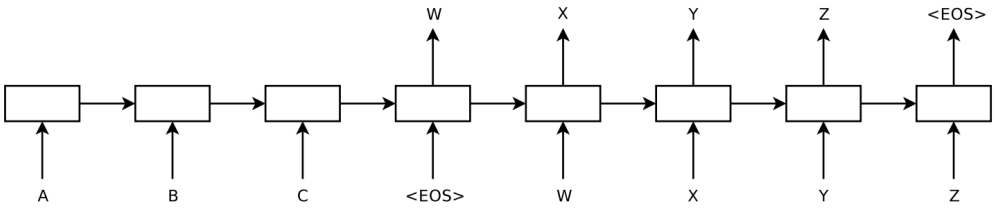


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

模型介绍

给定输入序列 $(x_1, \cdots, x_T)$，RNN 根据下式来生成输出序列 $(y_1, \cdots, y_T)$:

$$h_t = sigm(W^{hx}x_t + W^{hh}h_{t-1})$$

$$y_t = W^{yh}h_t$$

当输入输出之间的对齐关系已知的情况下，RNN 很容易处理序列-序列的映射，但当输入输出序列长度不同时，RNN 就无能为力了。一个策略就是用一个 RNN 将输入序列转化为固定长度的向量，用另一个 RNN 将向量转化为目标序列。但由于存在 long term dependencies 问题，RNN 很难训练（梯度消失）， 因而文章选择用 LSTM 来代替 RNN。

LSTM 要做的就是估计 $p(y_1, \cdots, y_{T'}|x_1, \cdots, x_T)$ 的条件概率，其中 $(x_1, \cdots, x_T)$ 为输入序列，$y_1, \cdots, y_{T'}$ 为对应的输出序列，T' 和 T 有可能不同。模型首先根据 LSTM 最后一个隐藏状态获得输入序列 $(x_1, \cdots, x_T)$ 的固定维度的向量表示 v，然后使用标准的 LSTM-LM 计算 $y_1, \cdots, y_{T'}$ 的概率：

$$p(y_1, \cdots, y_{T'}|x_1, \cdots, x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, \cdots, y_{t-1})$$

其中，每个 $p(y_t|v, y_1, \cdots, y_{t-1})$ 概率是所有词表上的 softmax。文章规定所有的句子都是以 "<EOS>" 结束，这样模型就可以计算任意长度句子的概率。

模型训练

使用 WMT'14 English to French 数据集，训练数据为 12M 句子的一个子集，包括 348M 个法语词汇，304M 个英语词汇。使用的词向量为16万英文词，8万法语词，未知单词采用 UNK。

训练的时候就是最大化对数概率，S 为源句子，T 为正确的翻译结果：

$$1/|S| \sum_{(T,S)\in S} \log p(T|S)$$

其中，S 为训练集。训练完成后，就可以求最大概率来得到翻译结果：

$$\hat{T} = \arg\max_{T} p(T|S)$$

在找最大可能翻译的过程中使用了 left-to-right beam search decoder。beam search 大概的思路是每次生成词是取使得整个概率最高的前 k 个词作为候选，显然 beam size 越大，效果越好，但是 beam size 越大会造成计算代价也越大，所以存在一个折中。

文章提出了一个小技巧：将输入序列翻转，效果更好。这里大概的解释思路就是，soft attention 或者 alignment 对于 seq2seq 这类问题有很大的提升，我们都知道 RNN 是一个有偏模型，顺序越靠后的单词在最终占据的信息量越大，那么如果是正序的话，最后一个词对应的 state 作为 decoder 的输入来预测第一个词，显然在 alignment 上来看，这两个词不是对齐的，反过来，如果用倒序的话，之前的一个词成了最后一个词，在 last state 中占据了主导，用这个词来预测 decoder 的第一个词，从某种意义上来说实现了 alignment，效果会更好一些。

训练的细节

使用 4 层 LSTM，每层 1000 个单元，词向量维度为 1000（输入词汇 16 万，输出词汇 8 万），深层 LSTM 表现更好，对每个输出使用 softmax。LSTM 有 384M 个参数，64M 为 pure recurrent connections（encoder LSTM 32M, decoder LSTM 32M）。

- LSTM 的参数初始化为 -0.08-0.08 的均匀分布；
- SGD 优化算法，学习率为 0.7。5 轮迭代后，每半轮迭代学习率减半，最终迭代 7.5 轮。
- 使用 mini-batch，每个 batch 128 个句子；
- 为了避免梯度爆炸，限制梯度大小。对每个 batch，计算 $s = \|g\|_2$，g 为梯度除以 128，当 $s > 5$ 时，另 $g = \frac{5g}{s}$。
- 保证每个 mini-batch 中的句子长度大致相等，这样可加快速度。