LSTM篇章自编码----0106论文会

2016年1月11日 星期一　　下午11:49

一、简介

任务是对篇章进行自编码(autoencoder for docoument)，就是对一篇文章而言
(encoder)，再做解码(decoder)，这个过程可以看做是压缩/解压的过程。

是15年6月，在arXiv上发出来的paper。斯坦福的jiwei Li发表的。

二、特点

1、针对篇章的特点(篇章由句子组成、句子由词组成)，使用LSTM。

2、在1的基础上，针对篇章分层级的特点()篇章由句子组成、句子由词组成，使
LSTM (Hierarchical)。

3、在1和2的基础上，使用attention model ，attention可以理解为在LSTM中
技巧，作用在于聚焦重要的输入。

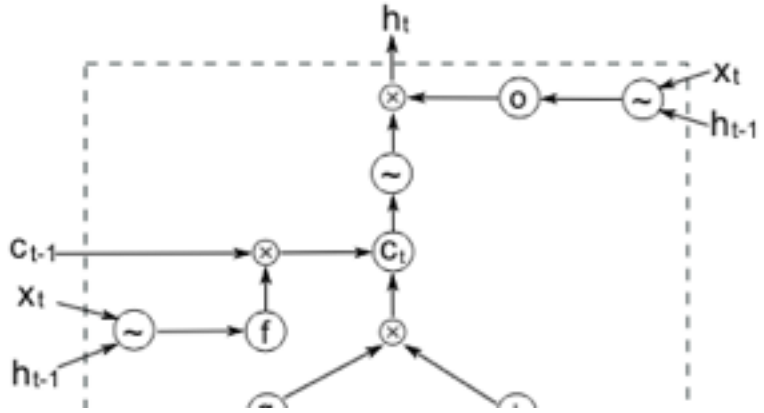
三、相关知识点

1、LSTM回顾

LSTM是考虑句子结构、历史信息，并且不忽视前面的历史信息。

由于RNN是序列化的输入，会导致遗忘靠前的输入的信息。在RNN中，对
序列中位置相关的，这个特点在很多应用中不合理。

具体做法是，通过一些参数门，控制遗忘/保留的比例。

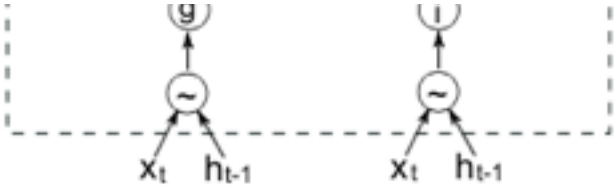


言，先做编码

使用分层级的

中，一个升级版的

对信息的遗忘是和



h:隐藏单元，c:存储单元， i:输入门，f:遗忘门，o:输出门，g:候选单元，⊗
激活函数

2、Attention Model

RNN，相对于BOW等方式，句子考虑结构信息。当对于单个的输入，相对
息。

LSTM是考虑句子结构、历史信息，并且不忽视前面的历史信息。

Attention Model是在LSTM基础上，不仅不忽视历史信息，而且能聚焦重
具体做法是，在t时刻，对前面输入的信息(t-1时刻)之前的，进行加权，叠
这个向量叫Attention Vector

这是借鉴于机器翻译中的方法。

3、encoder与encoder

自编码的过程（autoencoder），过程是把一个样本进行压缩，再进行解
原样本。

这个过程算法上的目的，可以理解为，是一个压缩-解压的过程，通过这一
本被做过一次抽象/特征提取，然后再被解压回来，是从特征生成原始输入
从应用上，一般用作神经网络模型训练的参数初始化，因为他有自编码的
对大规模无监督数据训练embedding

四、论文思路

1、整体思路

整体思路上，分为三步：

A standerd LSTM

word <-> sentence、sentence <-> document

B 分层级(hierarchical)

第一层 word <-> sentence

⊗:逐点乘积，~

对于考虑历史信

重要的历史信息。
加成一个向量，

压，期望能够还

个过程之后，样
入的过程。
过程，常常用于

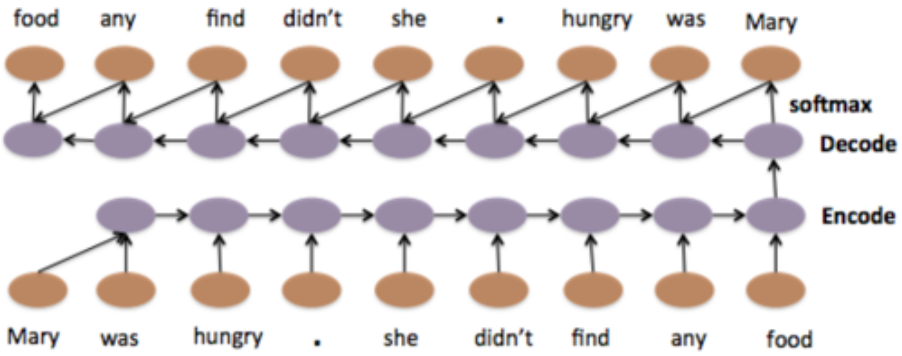
第二层 sentence <-> document

C Attention model

在LSTM基础上，聚焦重要的input

2、第一步 standerd LSTM

A encoder与decoder的过程本质是一样的



Encoder 词 -> 隐层 $h_t^w(\text{enc}) = LSTM_{\text{encode}}^{\text{word}}(e_t^w, h_{t-1}^w(\text{enc}))$

Decoder 隐层 -> 词 $p(w|\cdot) = \text{softmax}(e_w, h_{t-1}^w(\text{dec}))$

B encoder

对于每个输入的词，都会做一个LSTM的过程，生成隐层，句子之间

最终形成的Encoder的向量，相当于篇章级别的向量表示。

C decoder

encoder的向量，在decoder过程中作为初始状态/输入。

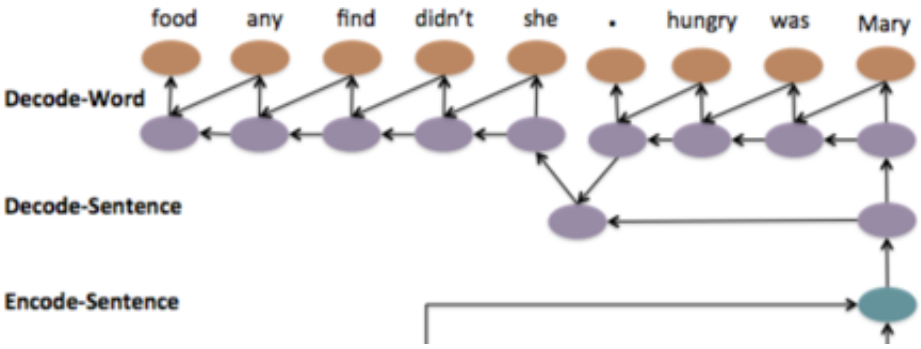
首先，对每个词解码的过程，其实和语言模型一样，是做一个多分类

其次，生成隐层的过程，和encoder是一致的。在生成t时刻隐层的时

时刻隐层和t时刻输入，只不过在decoder过程中，t时刻的输入不是

的词是要被预测的)，是t-1时刻的词(这里和语言模型类似)。

3、第二步 Hierarchical的LSTM

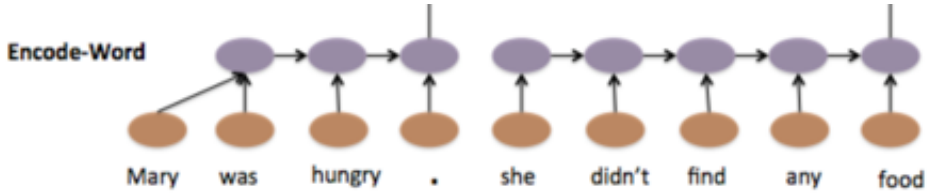


仅仅用句号分隔。

类任务。

时候，会基于t-1

t时刻的词(t时刻



其实思路很简单，就是分为两层：词->句子 句子->篇章

唯一值得注意的地方：

decoder过程中，同样的是用篇章级别向量作为初始输入，作为第一

但是在decoder到第二个句子的过程中，第二个句子的隐层(这里注意

子的向量)是怎么生产的呢？

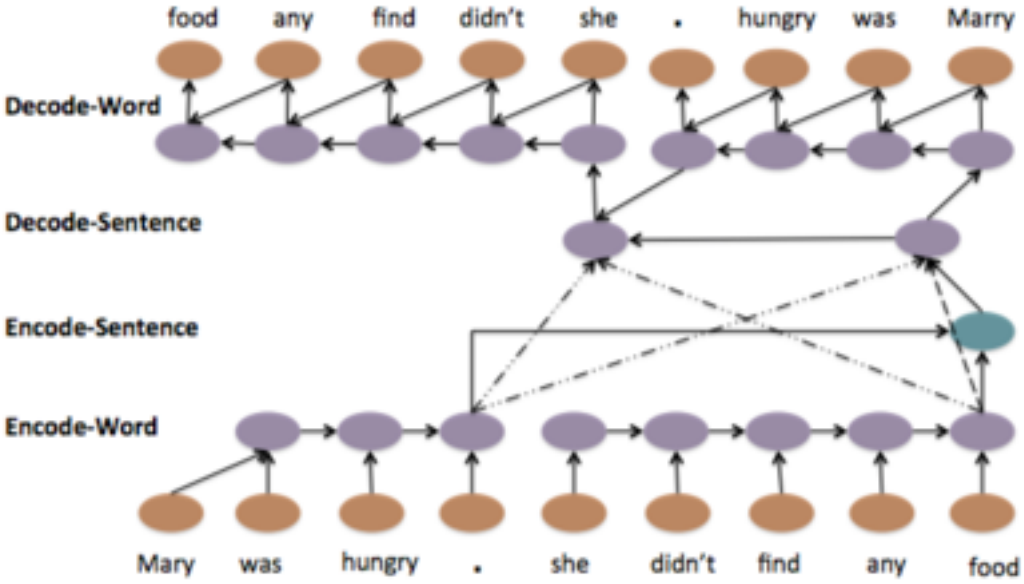
是用第一个句子最后一个词的隐层和第一个句子的隐层，为什么要用

一个词的向量呢？其实在这里，和encoder的过程一致，都是要用t-

时刻的输入，这里第一个句子最后一个词的隐层可以看做是第一个句

量，所以这里使用第一个句子最后一个词的隐层

4、第三步 LSTM + Hierarchical + Attention



图示

实线是Model 2， 虚线是Attention相关部分

A 加入Attention动机

a 聚焦重要的input

b 在decoder生成句子表示过程中，可以用更多encoder过程中信息

B 对网络结构的改进

一个句子的隐层。
意，不是第二个句

用第一个句子最后
-1时刻的隐层和t
句子的句子级别向

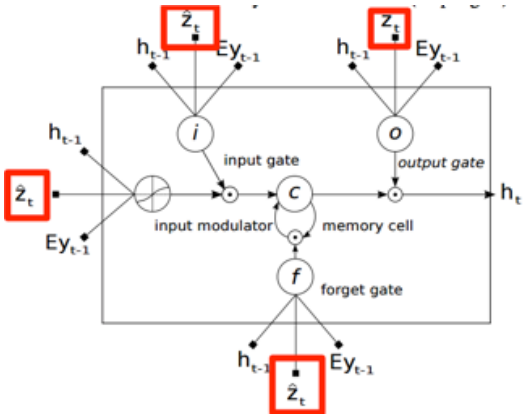
增加Attention vector (仅在decoder生成句子表示中)

Attention vector是对encoder中句子表示的加权平均

权重会参考decoder和encoder的句子表示

C 对网络结构改进的图示

把LSTM改成Attention Model (Kelvin Xu 2015)



D 在LSTM基础上，增加Attention的公式

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t \end{bmatrix}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t$$

$$h_t^s = o_t \cdot c_t$$

->

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1}^s \\ e_t^s \\ m \end{bmatrix}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t$$

$$h_t^s = o_t \cdot c_t$$

E Attention Vector的生成方式

Attention vector是对encoder中句子表示的加权平均

- $m_t = \sum_{i \in [1, N_D]} a_i h_i^s(\text{enc})$

权重(a_i)会参考decoder和encoder的句子表示

- $v_i = U^T f(W_1 \cdot h_{t-1}^s(\text{dec}) + W_2 \cdot h_i^s(\text{enc}))$
- $a_i = \frac{\exp(v_i)}{\sum_{i'} \exp(v_i')}$

第t个词的Attention Vector是对样本中t-1个句子的隐层做加权平均的结

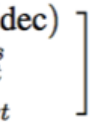
这个权重生成的方式是：对decoder的t-1个句子和encoder中每一个

数矩阵之后，做非线性，加上矩阵，再归一化。

四、参考文献

Kelvin Xu 2015

Show, attend and tell: Neural image caption generation with visual atten



结果。

i句子，都经过参

tion.

Show, attend and tell: Neural image caption generation with visual atten

五、我的附件

A Hierarchic…

A
Hierarchic…