2015

1. Abstract

机器翻译。
encoder + decoder
encoder: a fixed-lenght representation
decoder: a correct translation from this representation

RNN Encoder-Decoder
gated recusive rconvolutional neural network

the neural machine translation performs relatively well on short sentences without unknown words.
but degrades rapidly as the lengthe of the sentence and the number of unknown words.

gated recursive convolutional networks learns a grammatical struture

2. Introduction

SMT仅仅记忆一小部分内容
GRU可以记忆500MB内容

没有工作来分析这些模型的属性以及表现

understand the properties and behavior

grConv is able to learn, without supervision, a kind of syntactic struture over the sourece language.

> rnn, cnn paper

不懂概率的形式

> K是class labels 数量

p(x): joint distribution

以上抛弃

Learning Phrase Representations using RNN Encoder–Decoder
for Statistical Machine Translation

其实所谓的概率模型$p(x_1, x_2, ..., x_n)$， 其实是链式法则，直接看最后一层即可，这也就是RNN实际的含义。

RNN其实的目标是 maximize the conditional log-likelihood.

LSTM

1. the problem, with conventional "Back-Propagation Through TIme).
    - blow up
    - vanish

直接已经有的知识那么用

为什么RNN不能记忆知识？
反向传播时，<1.0的数引起梯度消失， >1.0的数引起梯度爆炸

LSTM: 输入、忘记、输出

主线剧情由输入分线进行控制

输入的内容会写入主线，然后忘记的内容会影响到输入的内容

RNN到底是什么？

network对于sequence序列问题解决的很好
上一个状态和当前状态

lstm使得其中cell的运算变得复杂

https://www.youtube.com/watch?v=EC3SvfW0Z_A

https://blog.csdn.net/FlyingLittlePig/article/details/72229041

> 非常好，有代码实现部分

https://www.yunaitong.cn/understanding-lstm-networks.html

> 这篇也有意思

https://www.youtube.com/watch?v=8HyCNIVRbSU

> 好像有乘积符号

> 传统RNN:

> $$h = tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

> LSTM:
>
> 1. forget gate layer
> $f_t = \delta(W_i \cdot [h_{t-1}, x_t] + b_i)$
> 2. input gate layer
> $i_t = \delta(W_i \cdot [h_{t-1,x_t}] + b_i)$
> $\tilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
> 3. the current state
> $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C_t}$
>
> > f_t, i_t分别表示forget gate的权重， input gate所占的权重
> > 根据维度进行猜想，那么这里实际上就不是元素的乘法了？？？
>
> 4. output layer
> $o_t = \delta(W_o \cdot [h_{t-1}, x_t] + b_o)$
> $h_t = o_t \odot tanh(C_t)$
> 5. predict
> $y_t = softmax(W_y \cdot h_t + b_t)$

> GRU:
> $z_t = \delta(W_z \cdot [h_{t-1}, x_t])$
> $r_t = \delta(W_r \cdot [h_{t-1}, x_t])$
> $\tilde{h}_t = tanh(W \cdot [r_t \odot h_{t-1}, x_t])$
> $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$
>
> > LSTM的变体，主要的改变在于， forget gate f_t和input gate i_t用了一个update gate z_t来替代.
> > 然后， current state的变化再加了一层， $\tilde{h}_t = tanh(W \cdot [r_t \odot h_{t-1}, x_t])$, 这里引入了一个$r_t$叫做reset gate.
> > 其次就是没有了output layer层，直接输出
> > 计算开销更小

> 参考
>
> 1. https://zhuanlan.zhihu.com/p/32481747
> 2. https://www.yunaitong.cn/understanding-lstm-networks.html
> 3. https://blog.csdn.net/FlyingLittlePig/article/details/72229041
> 4. https://www.youtube.com/watch?v=EC3SvfW0Z_A

GGNN

propgagation model

1. 初始化
   $h_v^{(1)} = [x_v^T, 0]^T$
2. 图操作
   $a_v^{(t)} = A_{v:}^T[h_1^{(t-1)T} ... h_{|V|}^{(t-1)T}]^T + b$
3. GRU
   $z_v^t = \delta(W^z a_v^{(t)} + U^z h_v^{(t-1)})$
   $r_v^t = \delta(W^r a_v^{(t)} + U^r h_v^{(t-1)})$
   $\hat{h}_v^{(t)} = tanh(W a_v^{(t)} + U(r_v^t \odot h_v^{(t-1)}))$
   $h_v^{(t)} = (1 - z_v^t) \odot h_v^{(t-1)} + z_v^t \odot \hat{h}_v^{(t)}$

new

1. 初始化
   $H^{(0)} = [X_v, P]$
   $A_v = [A_{in}, A_{out}]$

2. 图操作
   $T^{(t-1)} = A_v(H^{(t-1)} W^A)$

3. 节点操作
   $H^{(t)} =$
   $\delta(T^{(t-1)} W^{z_a}) \odot tanh(T^{(t-1)} W_{h_a}) +$
   $\delta(T^{(t-1)} W^{z_a}) \odot tanh\{\delta[(T^{t-1)} W^{r_a} \odot H^{(t-1)}) W^h]\} +$
   $\delta(T^{(t-1)} W^{z_a}) \odot tanh\{\delta[(H^{(t-1)} W_r) \odot H^{(t-1)}] + H^{(t-1)}\} +$
   $tanh(T^{(t-1)} W^{h_a}) \odot \delta(H^{t-1} W^{z_h}) +$
   $tanh\{\delta[(T^{(t-1)} W^{r_a} \odot H^{(t-1)}) W^h)]\} \odot \delta(H^{(t-1)} W^{z_a}) +$
   $\delta(H^{(t-1)} W^{z_h}) \odot tanh\{\delta[(H^{(t-1)} W^{r_h} \odot H^{(t-1)}) W^h] - H^{(t-1)}\} + H^{(t-1)}$