清华大学深圳研究生院

计算智能实验室 周报

姓名	研究方向	报告覆盖时间
李晨辉	NLP（法律文本分类）	2010.3.4.26-2019.3.10

本周主要完成的工作

一、深入理解 word2vec；

二、LSTM 等 RNN 模型的学习；

一、深入理解 word2vec

word2vec 是 google 在 2013 年推出的一个 NLP 工具，它的特点是将所有的词向量化，这样词与词之间就可以定量的去度量他们之间的关系，挖掘词之间的联系。在 word2vec 的实现中，有两种不同实现方式：

- 基于 Hierarchical Softmax 的模型

从隐藏层到输出的 softmax 层进行了改进。为了避免要计算所有词的 softmax 概率，采样了霍夫曼树来代替从隐藏层到输出 softmax 层的映射。

CBOW 模型

输入：基于 CBOW 的语料训练样本，词向量的维度大小 M，CBOW 的上下文大小 2c,步长 η

输出：霍夫曼树的内部节点模型参数 θ，所有的词向量 w

对于训练样本中的每一个词，其前面的 c 个词和后面的 c 个词作为了 CBOW 模型的输入,该词本身作为样本的输出，期望 softmax 概率最大。

步骤

1. 基于语料训练样本建立霍夫曼树。
2. 随机初始化所有的模型参数 θ，所有的词向量 w
3. 进行梯度上升迭代过程，对于训练集中的每一个样本(context(w),w)做如下处理：

a) e=0， 计算

Lw 为从根节点到 w 所在叶子节点包含的的节点总数

b) for j = 2 to l_w, 计算:

$$f = \sigma(x_w^T \theta_{j-1}^w)$$

$$g = (1 - d_j^w - f)\eta$$

$$e = e + g\theta_{j-1}^w$$

$$\theta_{j-1}^w = \theta_{j-1}^w + g x_w$$

c) 对于 context(w)中的每一个词向量 xi(共 2c 个)进行更新：xi=xi+e

d) 如果梯度收敛，则结束梯度迭代，否则回到步骤 3 继续迭代

Skip-Gram 模型

输入：基于 Skip-Gram 的语料训练样本，词向量的维度大小 M，Skip-Gram 的上下文大小 2c,步长 η

输出：霍夫曼树的内部节点模型参数 θ，所有的词向量 w

对于训练样本中的每一个词，该词本身作为样本的输入， 其前面的 c 个词和后面的 c 个词作为了 Skip-Gram 模型的输出,，期望这些词的 softmax 概率比其他的词大。

步骤

1. 基于语料训练样本建立霍夫曼树。
2. 随机初始化所有的模型参数 θ，所有的词向量 w
3. 进行梯度上升迭代过程，对于训练集中的每一个样本(w,context(w))做如下处理：

a) for i =1 to 2c:

i) e=0

ii)for j = 2 to l_w, 计算:

$$f = \sigma(x_i^T \theta_{j-1}^w)$$

$$g = (1 - d_j^w - f)\eta$$

$$e = e + g\theta_{j-1}^w$$

$$\theta_{j-1}^w = \theta_{j-1}^w + g x_i$$

iii)

$$x_i = x_i + e$$

b)如果梯度收敛，则结束梯度迭代，算法结束，否则回到步骤 a 继续迭代。

- 基于 Negative Sampling 的模型

输入：one-hot 的词表示，输出：对应词的得分，优化目标是使得正样例的得分接近于 1，负样例的得分接近于 0.

优化完成后得到的隐含层的词表示就是我们想要的词向量

输入层到隐藏层是词向量平均值，隐藏层到输出层为二元逻辑回归

CBOW 模型

输入：基于 CBOW 的语料训练样本，词向量的维度大小 Mcount，CBOW 的上下文大小 2c,步长 η，负采样的个数 neg

输出：词汇表每个词对应的模型参数 θ，所有的词向量 xw

步骤

1. 随机初始化所有的模型参数 θ，所有的词向量 w
2. 对于每个训练样本(context(w0),w0),负采样出 neg 个负例中心词 wi,i=1,2,...neg
3. 进行梯度上升迭代过程，对于训练集中的每一个样本 (context(w0),w0,w1,...wneg)做如下处理：

a) e=0，计算$x_{w_0} = \frac{1}{2c}\sum_{i=1}^{2c} x_i$

b) for i= 0 to neg, 计算:

$$f = \sigma(x_{w_0}^T \theta^{w_i})$$

$$g = (y_i - f)\eta$$

$$e = e + g\theta^{w_i}$$

$$\theta^{w_i} = \theta^{w_i} + g x_{w_0}$$

c) 对于$context(w)$中的每一个词向量x_k(共2c个)进行更新:

$$x_k = x_k + e$$

d) 如果梯度收敛，则结束梯度迭代，否则回到步骤3继续迭代。

Skip-Gram 模型

输入：基于 Skip-Gram 的语料训练样本，词向量的维度大小 Mcount，Skip-Gram 的上下文大小 2c,步长 η，负采样的个数 neg。

输出：词汇表每个词对应的模型参数 θ，所有的词向量 xw

步骤

1. 随机初始化所有的模型参数 θ，所有的词向量 w
2. 对于每个训练样本(context(w0),w0),负采样出 neg 个负例中心词 wi,i=1,2,...neg
3. 进行梯度上升迭代过程，对于训练集中的每一个样本 (context(w0),w0,w1,...wneg)做如下处理：

a) for i =1 to 2c:

i) e=0

ii) for j= 0 to neg, 计算:

$$f = \sigma(x_{w_{0i}}^T \theta^{w_j})$$

$$g = (y_j - f)\eta$$

$$e = e + g\theta^{w_j}$$

$$\theta^{w_j} = \theta^{w_j} + g x_{w_{0i}}$$

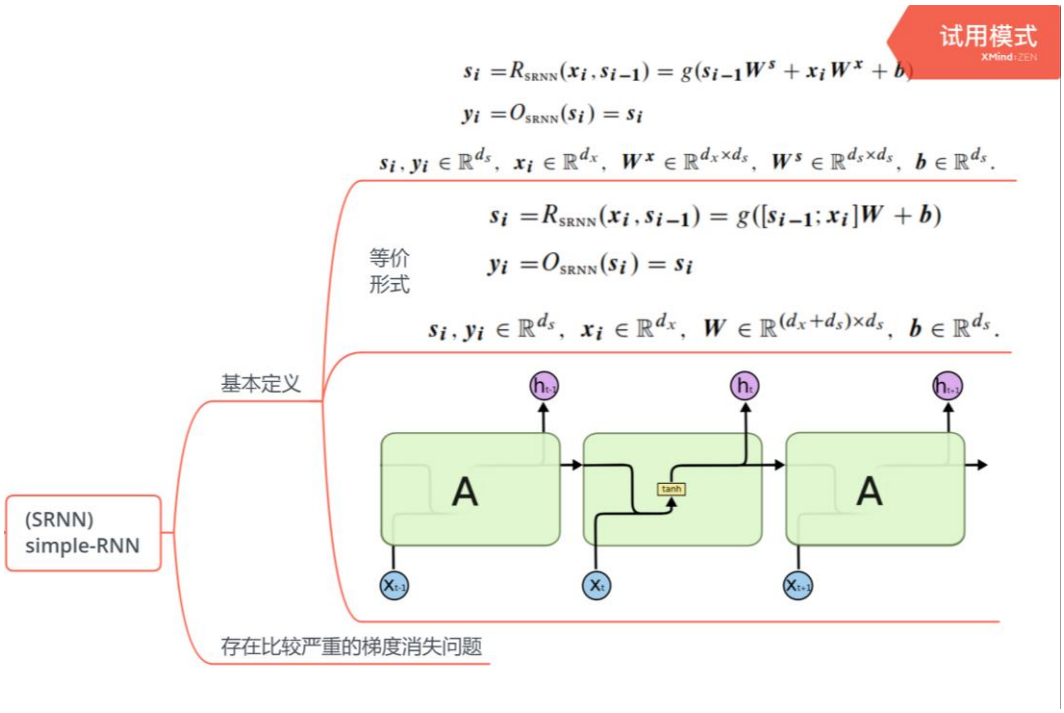
iii) 词向量更新:

$$x_{w_{0i}} = x_{w_{0i}} + e$$

b)如果梯度收敛，则结束梯度迭代，算法结束，否则回到步骤a继续迭代。

二、LSTM 网络学习

- 最简单 simple RNN 引入：



- LSTM (Long Short-Term Memory)

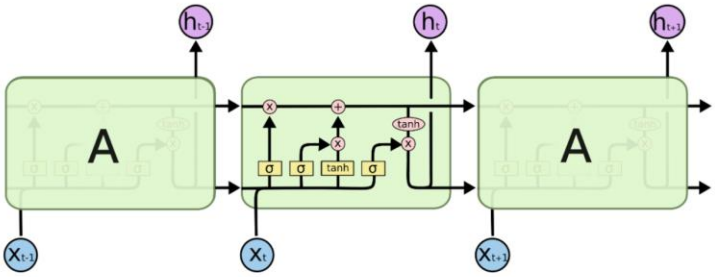
- LSTM (Long Short Term Memory)
  - Tips　hadamard乘积 $\boldsymbol{x} \odot \boldsymbol{g}$　表示元素级别的相乘
  - 被设计用于解决梯度消失问题
  - 定义

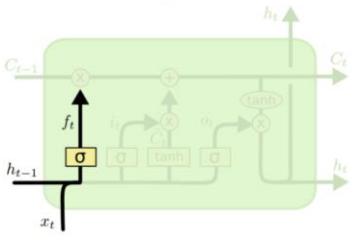
$$
\begin{aligned}
\boldsymbol{s_j} = R_{\text{LSTM}}(\boldsymbol{s_{j-1}}, \boldsymbol{x_j}) &= [\boldsymbol{c_j}; \boldsymbol{h_j}] \\
\boldsymbol{c_j} &= \boldsymbol{f} \odot \boldsymbol{c_{j-1}} + \boldsymbol{i} \odot \boldsymbol{z} \\
\boldsymbol{h_j} &= \boldsymbol{o} \odot \tanh(\boldsymbol{c_j}) \\
\boldsymbol{i} &= \sigma(\boldsymbol{x_j} \boldsymbol{W^{xi}} + \boldsymbol{h_{j-1}} \boldsymbol{W^{hi}}) \\
\boldsymbol{f} &= \sigma(\boldsymbol{x_j} \boldsymbol{W^{xf}} + \boldsymbol{h_{j-1}} \boldsymbol{W^{hf}}) \\
\boldsymbol{o} &= \sigma(\boldsymbol{x_j} \boldsymbol{W^{xo}} + \boldsymbol{h_{j-1}} \boldsymbol{W^{ho}}) \\
\boldsymbol{z} &= \tanh(\boldsymbol{x_j} \boldsymbol{W^{xz}} + \boldsymbol{h_{j-1}} \boldsymbol{W^{hz}})
\end{aligned}
$$

$$
\boldsymbol{y_j} = O_{\text{LSTM}}(\boldsymbol{s_j}) = \boldsymbol{h_j}
$$

$$
\boldsymbol{s_j} \in \mathbb{R}^{2 \cdot d_h},\ \boldsymbol{x_i} \in \mathbb{R}^{d_x},\ \boldsymbol{c_j}, \boldsymbol{h_j}, \boldsymbol{i}, \boldsymbol{f}, \boldsymbol{o}, \boldsymbol{z} \in \mathbb{R}^{d_h},\ \boldsymbol{W^{xo}} \in \mathbb{R}^{d_x \times d_h},\ \boldsymbol{W^{ho}} \in \mathbb{R}^{d_h \times d_h}
$$

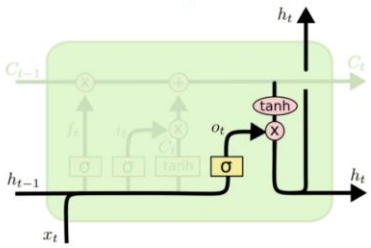
  - 拆解





$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

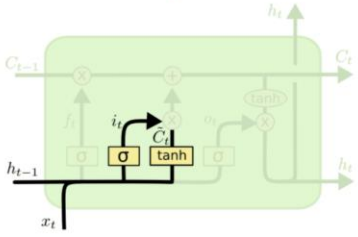
遗忘门



$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

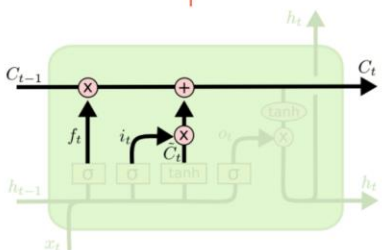
输出门



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

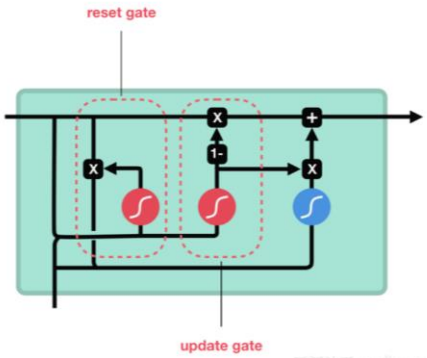
输入门



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

记忆组件

- GRU（Gated Recurrent Unit）



GRU
(Gated Recurrent Unit)

$$s_j = R_{\mathrm{GRU}}(s_{j-1}, x_j) = (1-z) \odot s_{j-1} + z \odot \tilde{s_j}$$
$$z = \sigma(x_j W^{xz} + s_{j-1} W^{sz})$$
$$r = \sigma(x_j W^{xr} + s_{j-1} W^{sr})$$
$$\tilde{s_j} = \tanh(x_j W^{xs} + (r \odot s_{j-1}) W^{sg})$$
$$y_j = O_{\mathrm{GRU}}(s_j) = s_j$$
$$s_j, \tilde{s_j} \in \mathbb{R}^{d_s},\ x_i \in \mathbb{R}^{d_x},\ z, r \in \mathbb{R}^{d_s},\ W^{xo} \in \mathbb{R}^{d_x \times d_s},\ W^{so} \in \mathbb{R}^{d_s \times d_s}$$

丢弃机制

目前在所有组成部分（层内、层间）采用相同的丢弃掩码效果最好

下周工作主要安排

(1) 统计自然语言处理方法的学习；

(2) 继续学习一些其他的数据结构；