3 RNN: 循环神经网络

RNN是一种能够对诸如自然语句、时间序列数据等变长序列化信息进行建模的前馈神经网络。一个简单RNN网络的工作流程如下：输入一个长度为T的序列$(x_1, \cdots, x_T)$，在每一个时间步中，RNN更新其隐藏层参数$(h_1, \cdots, h_T)$，并输出T个向量$(o_1, \cdots, o_T)$。在时间步变量t从1增长到T的过程中，h_t和o_t的更新遵循以下方程：

$$
\begin{aligned}
h_t &= tanh(Ux_t + Wh_{t-1} + b) \\
o_t &= Vh_t + c
\end{aligned}
\tag{1}
$$

其中，U、W和V分别表示隐藏层中输入层、隐藏层和输出层的权重矩阵（详见下图2(a)中对简单tanh-RNN网络参数的解释），b和c是偏置向量，$tanh(.)$是非线性的双曲正切函数。

理论上，RNN的梯度通过后向传播算法迭代计算得到[Rumelhart *et al.*, 1986]。然而在实际中，由于梯度消失或爆炸的问题[Bengio *et al.*, 1994]，简单RNN网络无法利用基于梯度的优化算法实现对较远距离外（序列信息间的）时间依赖性的学习。一种解决方案是增加能存储长远时序信息的“记忆”单元，譬如众所周知的长短时记忆单元LSTM[Hochreiter and Schmidhuber, 1997; Graves, 2013]和门控循环单元GRU[Cho *et al.*, 2014]。接下来，我们将简单介绍这两种单元的结构。

3.1 长短时记忆单元LSTM

与传统循环单元中在每轮训练时迭代上轮参数和结果的操作（方程1）不同，LSTM用一个记忆区c_t来记录第t位时间步上的训练结果。LSTM的输出h_t遵循以下方程[Hochreiter and Schmidhuber, 1997; Graves, 2013]：

$$
\begin{aligned}
i_t &= \sigma(x_t W_i + h_{t-1} U_i + c_{t-1} V_i) \\
f_t &= \sigma(x_t W_f + h_{t-1} U_f + c_{t-1} V_f) \\
\tilde{c_t} &= tanh(x_t W_c + h_{t-1} U_c) \\
c_t &= f_t c_{t-1} + i_t \tilde{c_t} \\
o_t &= \sigma(x_t W_o + h_{t-1} U_o + c_t V_o) \\
h_t &= \sigma_t tanh(c_t)
\end{aligned}
$$

其中，$\sigma(.)$是logistics sigmoid函数。输入门i_t决定了新记忆被添加到记忆区中的程度，遗忘门f_t决定了现有记忆被遗忘的程度。记忆c_t通过对现有记忆的遗忘和新记忆$\tilde{c}_t$的增加进行更新。输出门o_t给出输出记忆的大小。

3.2 门控循环单元GRU

与LSTM相似，GRU也使用门机制调整单元内的内容流量，不过它使用了更少的参数，因此也更简单。GRU层使用了以下函数[Cho *et al.*, 2014]：

$$
\begin{aligned}
z_t &= \sigma(x_t U_z + h_{t-1} W_z) \\
r_t &= \sigma(x_t U_r + h_{t-1} W_r) \\
\tilde{h_t} &= tanh(x_t U_h + (h_{t-1} \cdot r_t) W_h) \\
h_t &= (1 - z_t) h_{t-1} + z_t \tilde{h_t}
\end{aligned}
$$

其中，重置门r_t决定了新输入与此前记忆的结合方式，更新门z_t设定了旧记忆传递到当前时间步的程度，而$\tilde{h}_t$表示隐藏状态h_t的候选激活函数的计算结果（Candidate Activation）。

4 基于RNN的谣言检测

下文将详细描述对微博事件进行谣言检测的RNN模型。首先，我们会介绍一种将一簇微博转化为连续变长时间序列数据的方法；其次，我们会描述多种含有不同类型隐藏层的RNN分类模型。

4.1 问题描述

一般来说，微博的特点是篇幅短小、内容有限。而一个事件的讨论往往涉及众多与之相关的微博。我们不关心单篇博文，而是关心一簇相关博文体现的整体内容。因此，我们不判定每篇微博的真实程度，而是从事件层面对一簇相关微博进行整体的谣言检测。

我们定义事件集合为$E = \{E_i\}$，其中每个事件$E = \{(m_{i,j}, t_{i,j})\}$由时刻$t_{i,j}$时全部相关微博$m_{i,j}$表示。我们的任务是判定每个事件是否为谣言。

4.2 （构造）变长时间序列数据

我们可以将每篇微博建模为一个输入，并使用RNN对将一组微博描述成长度为微博数量的时间序列数据进行建模。一个热度较高的事件可能涉及成千上万篇微博，但我们却只在每个事件对应的时间序列数据的最后一个时间步使用一个输出单元来预测其类别。当只使用这最后一步误差来表征大量先前时间步中积累的误差时，后向传播算法就会变得代价高昂且无效。因此，我们按时间段对微博进行切割，把每个时间段中的全部微博建模为一个输入后，再建立RNN序列。在这样构造时间序列数据时我们采用了参考的RNN序列长度。

时间段的长度表征了相关博文在传播时的密集程度，因此需要恰当地选择。我们认为时间段的总数应当约等于参考的RNN序列长度，故由此可以反向确定时间段的长度。算法1描述了构造一个事件的时间序列数据的整个过程：

(1)我们将整个时间线均分为N个时间段（N就等于参考长度）。

(2)通过从集合U_0中移除空时间间隔，我们的系统查找到非空时间间隔的集合U_1(非空指的是U_1中每个时间段内至少有一条相关微博)，再由此得到总时间间隔最长的连续时间段的集合$\tilde{U}_1$。

(3)如果$\tilde{U}_1$中时间段的数量小于N且大于上一轮$\tilde{U}_1$中时间段的数量，我们就将时间段二分并重复(1)、(2)；否则，返回输出$\tilde{U}_1$中已发现的连续时间段。

应当注意，对于每个事件：其时间序列数据的长度，尽管接近N，却随不同事件而变化；但其每个时间段的长度都是相等的。

算法1: 给定事件的相关微博集和RNN参考长度时构建变长时间序列数据的算法

输入：事件的相关微博集 $E = \{(m_{i,j}, t_{i,j})\}_{j=1}^{n_i}$ **，参考的RNN长度** N

输出：连续时间段的集合 $I = \{I_1, I_2, \cdots\}$

/*程序初始化*/

$L(i) = t_{i,n_i} - t_{i,1}; l = \frac{L(i)}{N}; k = 0;$

while true do

  $k++;$

  $U_k \leftarrow$ **根据** $L(i)$ **和** l **均分整个时间线；**

  $U_0 \leftarrow U_k$ **中的全部空时间段；**

  查找 $U_k - U_0$ **中总时间间隔最长的连续时间段的集合** $\tilde{U}_1$ **；**

  if $|\tilde{U}_k| < N$ **and** $|\tilde{U}_k| > |\tilde{U}_{k-1}|$ **then**

    /*二分时间段*/

    $l = 0.5l$

  else

    /*输出结果*/

    $I = \{I_0 \in \tilde{U}_k | I_1, ..., I_{|\tilde{U}_k|}\};$

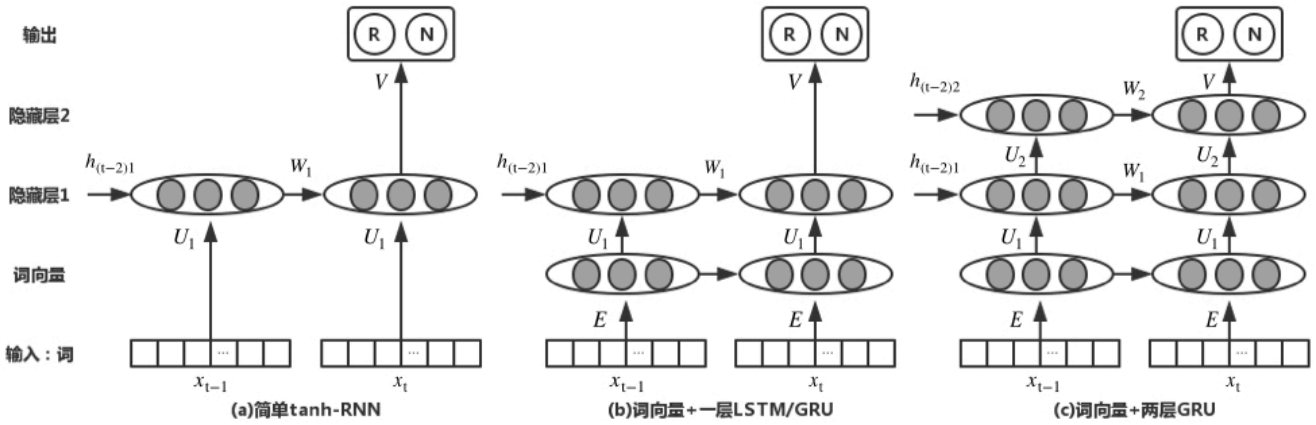
    return $I;$

  end

end

return $I;$

图2: 基于RNN的谣言检测模型。E是词向量权重矩阵，U、W和V表示隐藏层和输出层中的参数。R表示谣言，N表示非谣言。



4.3 建立模型

4.2节中构造的时间序列数据天然适配于RNN循环单元。对于每个时间段，我们使用词表中$tf*idf$值排序在Top-K及以上的词作为输入，因此输入特征的维度为K。如图2所示，我们构造了3种不同结构的RNN模型。要指出的是，输出层是与最后一个时间步相连接，并使用softmax函数给出事件分别属于谣言和非谣言的概率。

(1)**tanh-RNN**

tanh-RNN是不含隐藏单元的最基本模型，因此它只能有限地捕捉时间段之间的内容。

定义g_c是事件在谣言/非谣言分类上的真实二维多项分布，其中c表示两个类别的标注，谣言的概率分布区间为[1, 0]，非谣言的概率分布区间为[0, 1]。对于每个训练样本(即每个事件)，我们的目标是最小化预测概率分布与真实概率分布之间的方差：

$$\min \sum_{c} (g_c - p_c)2 + \sum_{c} ||\theta_i||2$$

其中g_c和p_c分别是真实及预测的概率分布，θ_i表示模型中待估计的参数，L2范数用于控制计算误差与模型规模之间的平衡。图2(a)展示了tan-RNN模型。

⑵**单层LSTM/GRU**

长远时序信息可以帮助我们捕捉一个事件的生命周期中谣言的特征与隐藏信息。因此，我们将基本循环单元修改为带LSTM或GRU的门控单元。图2(b)展示了带门控单元的RNN模型。门控单元不仅包含当前时间步的内容，还为止引入了此前时间步中的相关信息。

然而，门控单元的存在使得参数规模显著扩大。比如说，由于引入了重置门与更新门，GRU使原来的参数规模扩大到三倍。为了解决这个问题，我们在输入层与隐藏层之间引入了定长为100的词向量层，将稀疏的词袋输入特征转化低维词向量表示，使得整体的参数规模大大缩小。与从外部寻求预训练好的词向量不同，我们的词向量矩阵E是从我们的模型中训练得到的。

⑶**多层GRU**

通过在单层GRU上增加第二层GRU，我们进一步构建了能够捕捉不同时间步之间更高级交互特征的多层GRU模型。我们仅使用GRU来构造多层模型，这是因为LSTM的参数相比GRU太过复杂。图2(c)描述了这一模型的结构。隐藏单元的状态通过以下方程计算得到，其中方程2对应词向量层，方程组(3、4、5、6)描述第一层GRU，而方程组(7、8、9、10)表示了第二层GRU：

$$x_e = x_t E \tag{2}$$
$$z_{t1} = \sigma(x_e U_{z1} + h_{(t-1)1} W_{z1}) \tag{3}$$
$$r_{t1} = \sigma(x_e U_{r1} + h_{(t-1)1} W_{r1}) \tag{4}$$
$$\tilde{h}_{t1} = tanh(x_e U_{h1} + (h_{(t-1)1} * r_{t1}) W_{h1}) \tag{5}$$
$$h_{t1} = (1 - z_{t1}) h_{(t-1)1} + z_{t1} \tilde{h}_{t1} \tag{6}$$
$$z_{t2} = \sigma(h_{t1} U_{z2} + h_{(t-1)2} W_{z2}) \tag{7}$$
$$r_{t2} = \sigma(h_{t1} U_{r2} + h_{(t-1)2} W_{r2}) \tag{8}$$
$$\tilde{h}_{t2} = tanh(h_{t1} U_{h2} + (h_{(t-1)2} * r_{t2}) W_{h2}) \tag{9}$$
$$h_{t2} = (1 - z_{t2}) h_{(t-1)2} + z_{t2} \tilde{h}_{t2} \tag{10}$$

其中，E是词向量权重矩阵，(U_1, W_1)和(U_2, W_2)分别是两层GRU隐藏层中的参数。

⑷**模型的训练**

通过对损失的梯度推演，我们使用反向传播算法[Col- lobert *et al.*, 2011]对所有RNN模型中的全部参数进行训练。其中，参数更新使用的是AdaGrad算法[Duchi *et al.*, 2011]。根据经验，词表维度K、词向量定长、隐藏单元个数和学习率分别被设置为5000、100、100和0.5。在每轮训练中，我们使用了全部的训练样本；模型训练直到损失值收敛或训练轮数达到最大阈值时停止。