

# RNN情感分析

原创 深度学习科研平台 深度学习科研平台 2020-05-08

## 回顾

上一节节课中，我们介绍了卷积神经网络的最经典模型——`LeNet`

- 1.首先那我们分析了卷积给神经网络带来的变化。
- 2.我们介绍了LeNet的基本结构。
3. 然后分析了`LeNet`在TensorFlow中的显示以及两个常用的scope.
- 4.最后介绍了如何使用第一部分介绍的基本逻辑，训练一个`LeNet`网络

## IMDB

### IMDB电影评论情绪分类

数据集来自IMDB的2.5万部电影评论，以情绪（正面/负面）标记。每条样本是一个txt文件。包括训练集，测试集，和没有标签的数据。

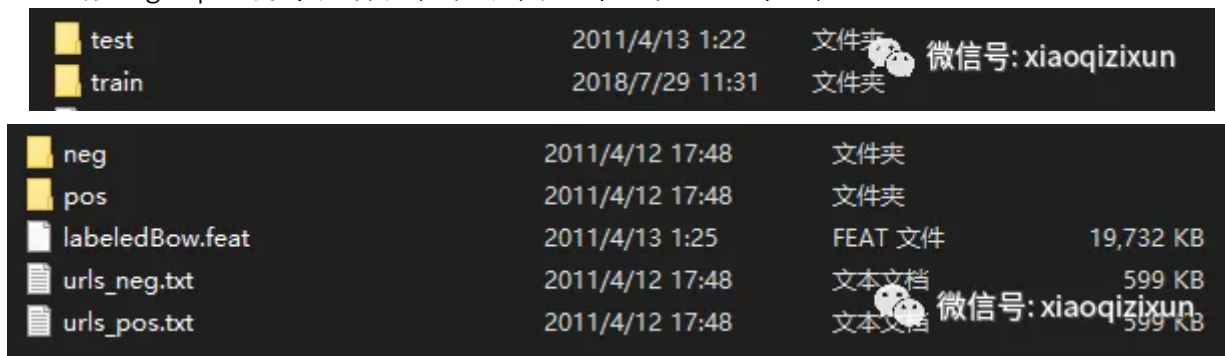
训练集：25000条，正负各12500条

测试集：25000条，正负各12500条

数据格式：

Train、test两个文件夹，分别代表训练集和测试集

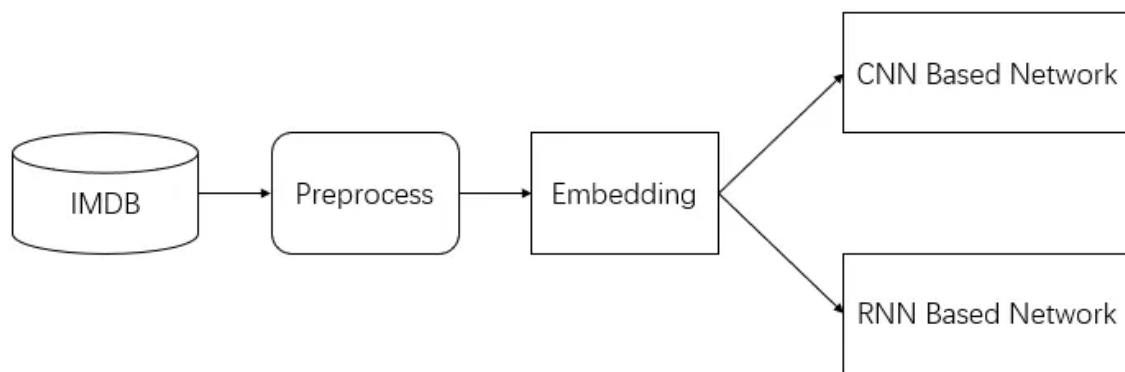
Train下面有neg、pos两个文件夹，代表负面评论和正面评论，test同理



test	2011/4/13 1:22	文件夹	
train	2018/7/29 11:31	文件夹	
neg	2011/4/12 17:48	文件夹	
pos	2011/4/12 17:48	文件夹	
labeledBow.feats	2011/4/13 1:25	FEAT 文件	19,732 KB
urls_neg.txt	2011/4/12 17:48	文本文件	599 KB
urls_pos.txt	2011/4/12 17:48	文本文件	599 KB

有一个叫做imdb.vocab的文件，代表单词表

## Model



微信号: xiaoqizixun

### BuildTFRecord

方案1: 直接将整个序列存起来

如: I am me -> "Iam me"

问题:

变长

浪费存储空间

解决方案:

每个单词在单词表的位置是一个整数, 单词表的大小是89527

最长的句子拥有的单词数为每个句子的单词数 (变长转定长)

多余位置用endchar (89528) 填充

可以运行一下dp.py看看效果?

[	10	18	6	...	89528	89528	89528]	0
[	45761	23438	7460	...	89528	89528	89528]	1
[	9	63	199	...	89528	89528	89528]	1
[	1	27897	111	...	89528	89528	89528]	0
[	10	16	12	...	89528	89528	89528]	0
[	211	19	46	...	89528	89528	89528]	0
[	3	370	18	...	89528	89528	89528]	1
[	12	10	1	...	89528	89528	89528]	0
[	10	16	6	...	89528	89528	89528]	1
[	1	0	1034	...	89528	89528	89528]	1
[	9	138	5	...	89528	89528	89528]	0
[	4616	287	10	...	89528	89528	89528]	1
[	9	12	111	...	89528	89528	89528]	0
[	2216	946	122	...	89528	89528	89528]	1
[	9	25	5	...	89528	89528	89528]	0
[	1361	2096	6	...	89528	89528	89528]	1
[	216	1	425	...	89528	89528	89528]	1
[	41	2076	1174	...	89528	89528	89528]	0
[	8	3	17836	...	89528	89528	89528]	0
[	9	498	10	...	89528	89528	89528]	0
[	832	13	3	...	89528	89528	89528]	0

微信号: xiaoqizixun

Embedding

$$\begin{pmatrix} the \\ cat \\ sat \\ on \\ the \\ mat \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$



<i>anarchism</i>	0.5	0.1	-0.1
<i>originated</i>	-0.5	0.3	0.9
<i>as</i>	0.3	-0.5	-0.3
<i>a</i>	0.7	0.2	-0.3
<i>term</i>	0.8	0.1	-0.1
<i>of</i>	0.4	-0.6	-0.1
<i>abuse</i>	0.7	0.1	-0.4

one-hot表示方式很直观，但是有两个缺点：

1. 第一，矩阵的每一维长度都是字典的长度，比如字典包含10000个单词，那么每个单词对应的one-hot向量就是1X10000的向量，而这个向量只有一个位置为1，其余都是0，浪费空间，不利于计算。
2. one-hot矩阵相当于简单的给每个单词编了个号，但是单词和单词之间的关系则完全体现不出来。比如“cat”和“mouse”的关联性要高于“cat”和“cellphone”，这种关系在one-hot表示法中就没有体现出来。



微信号: xiaoqizixun


举个例子：

```
import tensorflow as tf
import numpy as np

vocab_size = 5
emb_size = 6
embedding = np.random.random([10, 6])
inputs = np.array([[2, 3], [1, 4], [1, 2], [1, 3]], dtype=np.int32)
embedding_layer = tf.nn.embedding_lookup(embedding, inputs)
sess = tf.Session()
print(sess.run([embedding_layer]))
```

向量是随机生成的

但是相同的单词编号会输出相同的变量


 微信号: xiaoqizixun

进一步：

```
import tensorflow as tf
import numpy as np

vocab_size = 5
emb_size = 6
embedding = tf.get_variable("embedding", [vocab_size, emb_size],
                             initializer=tf.truncated_normal_initializer(stddev=0.1))
inputs = np.array([[2, 3], [1, 4], [1, 2], [1, 3]], dtype=np.int32)
embedding_layer = tf.nn.embedding_lookup(embedding, inputs)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
print(sess.run([embedding_layer]))
```

由tensorflow随机生成一个矩阵（单词~向量表），起到和上段代码一样的效果

 微信号: xiaoqizixun

这个单词~向量矩阵不是一成不变的

可以看成是一个“全连接层”

这个矩阵的每一行都是一个共享的参数

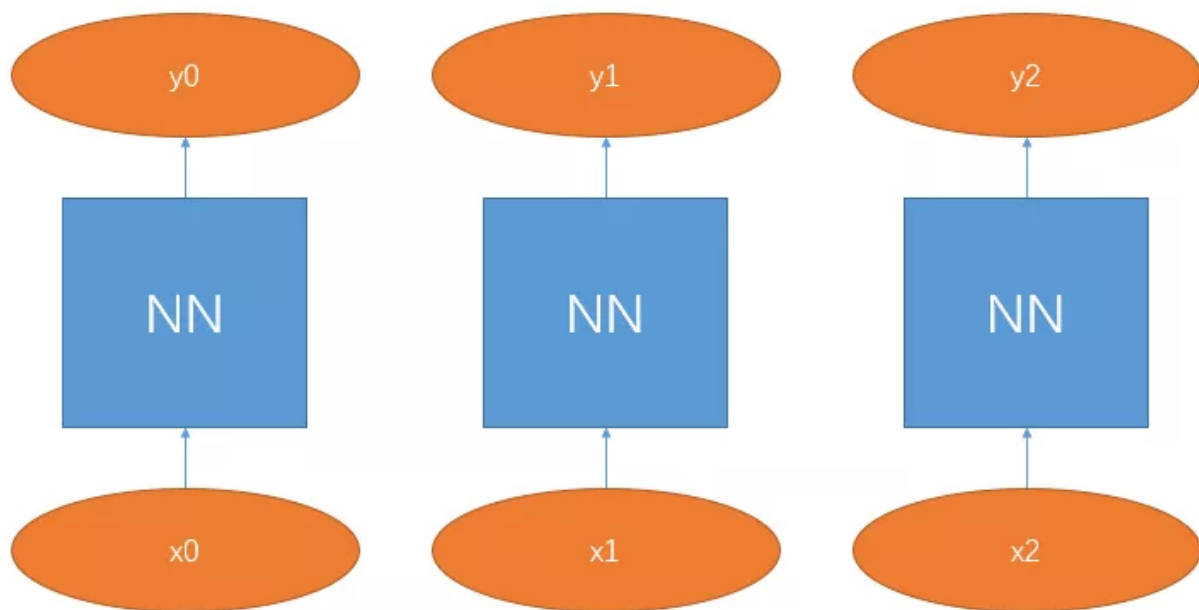
可以利用神经网络的方法进行训练操作

```

1  # 第一个卷积层，卷积为1维卷积。
2  net = end_points['conv1'] = tf.layers.conv1d(embedding_layer, hide_dim, 3,
3        kernel_regularizer=tf.contrib.layers.l2_regularizer(weight_decay),
4        kernel_initializer=tf.truncated_normal_initializer(stddev=0.1),
5        activation=tf.nn.relu,
6        padding='same', name='conv1')
7  # Maxpooling
8  net = end_points['pool1'] = tf.reduce_max(net, [1], name='pool1')
9  # 旁路分支卷积层，卷积为1维卷积。
10 net = end_points['conv2'] = tf.layers.conv1d(embedding_layer, hide_dim, 2,
11        kernel_regularizer=tf.contrib.layers.l2_regularizer(weight_decay),
12        kernel_initializer=tf.truncated_normal_initializer(stddev=0.1),
13        activation=tf.nn.relu,
14        padding='same', name='conv2')
15 # Maxpooling
16 net = end_points['pool2'] = tf.reduce_max(net, [1], name='pool2')
17 # 将两路卷积的分支进行拼接，构成一个完整的特征向量。
18 net = tf.concat([end_points['pool1'], end_points['pool2']], axis=1, name='concat')

```

微信号: xiaoqizixun

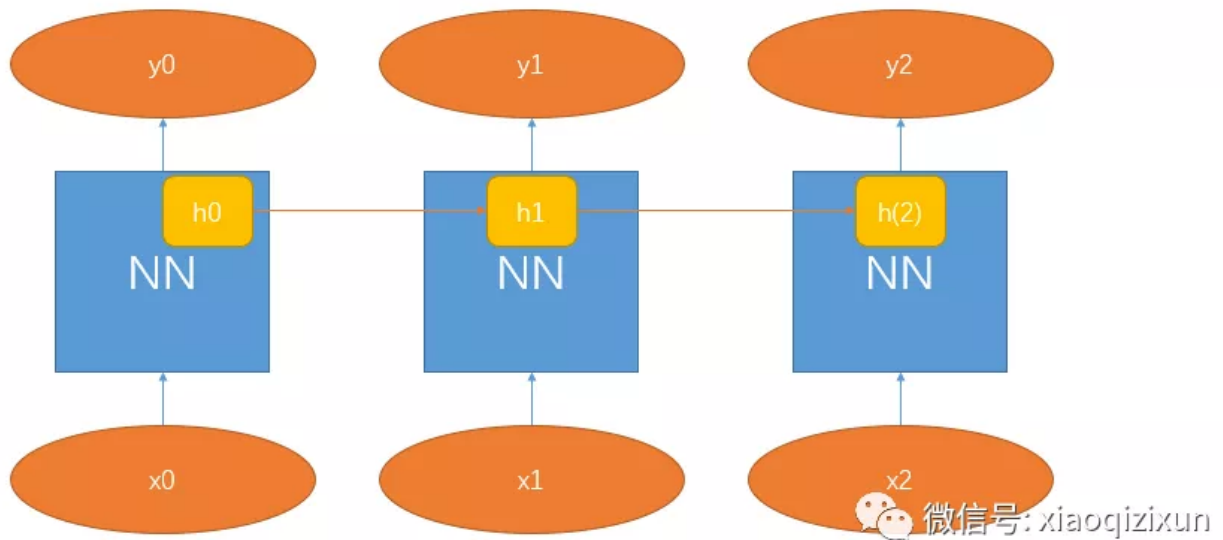


微信号: xiaoqizixun

RNN

$$y_i = f(Ux_i + Wx_i + b)$$

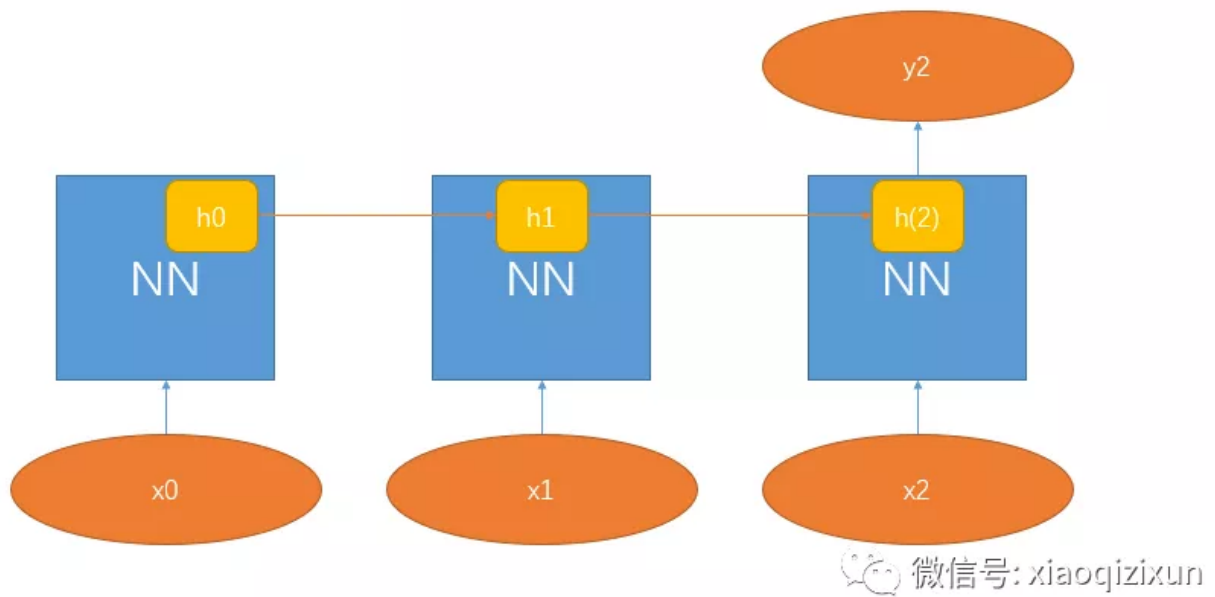
$$h_i = f(Ux_i + wh_{i-1} + b)$$



RNN

$$y_i = f(Ux_i + Wx_i + b)$$

$$h_i = f(Ux_i + Wh_{i-1} + b)$$



$$y_i = f(Ux_i + Wx_i + b)$$

$$h_i = f(Ux_i + Wh_{i-1} + b)$$

当  $W < 1$  时可能出现梯度消失

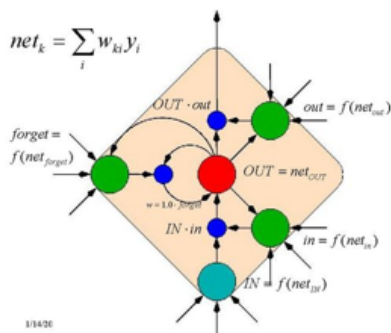
当  $W > 1$  时可能出现梯度爆炸

Tanh

Relu?



# LSTM – Long Short-Term Memory



Allow each time step to modify

- Input gate (current cell matters)  $i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1})$
- Forget (gate 0, forget past)  $f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1})$
- Output (how much cell is exposed)  $o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1})$
- New memory cell  $\tilde{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1})$

Final memory cell:

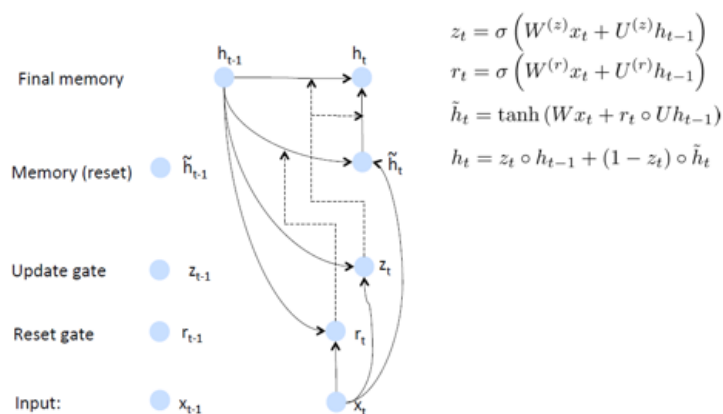
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

Final hidden state:

$$h_t = o_t \circ \tanh(c_t)$$

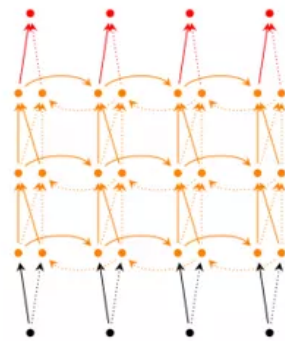
微信号: xiaoqizixun

## RNN -- GRU



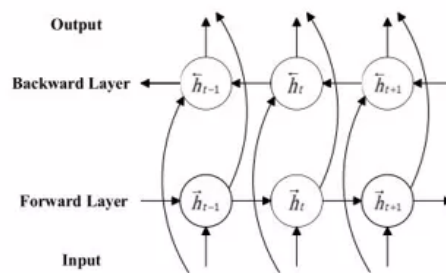
微信号: xiaoqizixun

# RNN



Multi-RNN

```
1 num_units = [128, 64]
2 cells = [BasicLSTMCell(num_units=n) for n in num_units]
3 stacked_rnn_cell = tf.nn.nn.MultiRNNCell(cells)
```



Bi-RNN

```
1 tf.nn.static_bidirectional_rnn(
2     cell_fw,
3     cell_bw,
4     inputs,
5     initial_state_fw=None,
6     initial_state_bw=None,
7     dtype=None,
8     sequence_length=None,
9     scope=None
10 )
```

微信号: xiaoqizixun

## RNNModel

```
1 # LSTM作为最基本的RNN单元。
2 rnn_cell_basic = tf.nn.nn.LSTMCell(hide_dim, use_peepholes=True,
3                                     initializer=tf.truncated_normal_initializer(stddev=0.1))
4 # 如果需要多层的RNN可以设置这个参数。
5 # rnn_cell_basic = tf.nn.nn.MultiRNNCell([rnn_cell_basic] * num_rnn_layers)
6 # 这里需要将输入的数据，在每一个step上进行拆分，每次作为一个输入。
7 steps_features = [tf.squeeze(input_, [1]) for input_ in tf.split(embedding_layer, max_seq_length, 1)]
8 # 放入到RNN网络，输出为每个step的特征以及对应的最后的状态特征。
9 outputs_rnn, _ = tf.nn.nn.static_rnn(rnn_cell_basic, steps_features, dtype=tf.float32)
10 net = tf.concat(outputs_rnn, axis=1, name='concat')
```

微信号: xiaoqizixun

## 总结

在这节课中，我们以IMDB数据集为例，介绍了文本情感分析的一个具体实例。

1. 首先我们简单的介绍了一下IMDB数据集的构成。

2. 我们介绍了处理文本的整体思路。

1. WordEmbedding

2. CNN or RNN

3. 然后介绍了一下Embedding的具体实现。

4. 最后我们分别以CNN以及RNN为例，构建了一个可以训练的情感分析模型。

喜欢此内容的人还喜欢

迁移学习概念、分类及应用

深度学习科研平台