• 资源

一个基于TensorFlow的简单故事生成案例：带你了解LSTM

2017-04-25 15:54:21 • Tensor Flow

在深度学习中，循环神经网络（RNN）是一系列善于从序列数据中学习的神经网络。由于对长期依赖问题的鲁棒性，长短期记忆（LSTM）是一类已经有实际应用的循环神经网络。现在已有大量关于 LSTM 的文章和文献，其中推荐如下两篇：

- Goodfellow et.al.《深度学习》一书第十章：http://www.deeplearningbook.org/
- Chris Olah：理解 LSTM：http://colah.github.io/posts/2015-08-Understanding-LSTMs/

已存在大量优秀的库可以帮助你基于 LSTM 构建机器学习应用。在 GitHub 中，谷歌的 TensorFlow 在此文成文时已有超过 50000 次星，表明了其在机器学习从业者中的流行度。

与此形成对比，相对缺乏的似乎是关于如何基于 LSTM 建立易于理解的 TensorFlow 应用的优秀文档和示例，这也是本文尝试解决的问题。

假设我们想用一个样本短故事来训练 LSTM 预测下一个单词，伊索寓言：

> long ago , the mice had a general council to consider what measures they could take to outwit their common enemy , the cat . some said this , and some said that but at last a young mouse got up and said he had a proposal to make , which he thought would meet the case . you will all agree , said he , that our chief danger consists in the sly and treacherous manner in which the enemy approaches us . now , if we could receive some signal of her approach , we could easily escape from her . i venture , therefore , to propose that a small bell be procured , and attached by a ribbon round the neck of the cat . by this means we should always know when she was about , and could easily retire while she was in the neighbourhood . this proposal met with general applause , until an old mouse got up and said that is all very well , but who is to bell the cat ? the mice looked at one another and nobody spoke . then the old mouse said it is easy to propose impossible remedies .

Listing 1．取自伊索寓言的短故事，其中有 112 个不同的符号。单词和标点符号都视作符号。

如果我们将文本中的 3 个符号以正确的序列输入 LSTM，以 1 个标记了的符号作为输出，最终神经网络将学会正确地预测下一个符号（Figure1）。

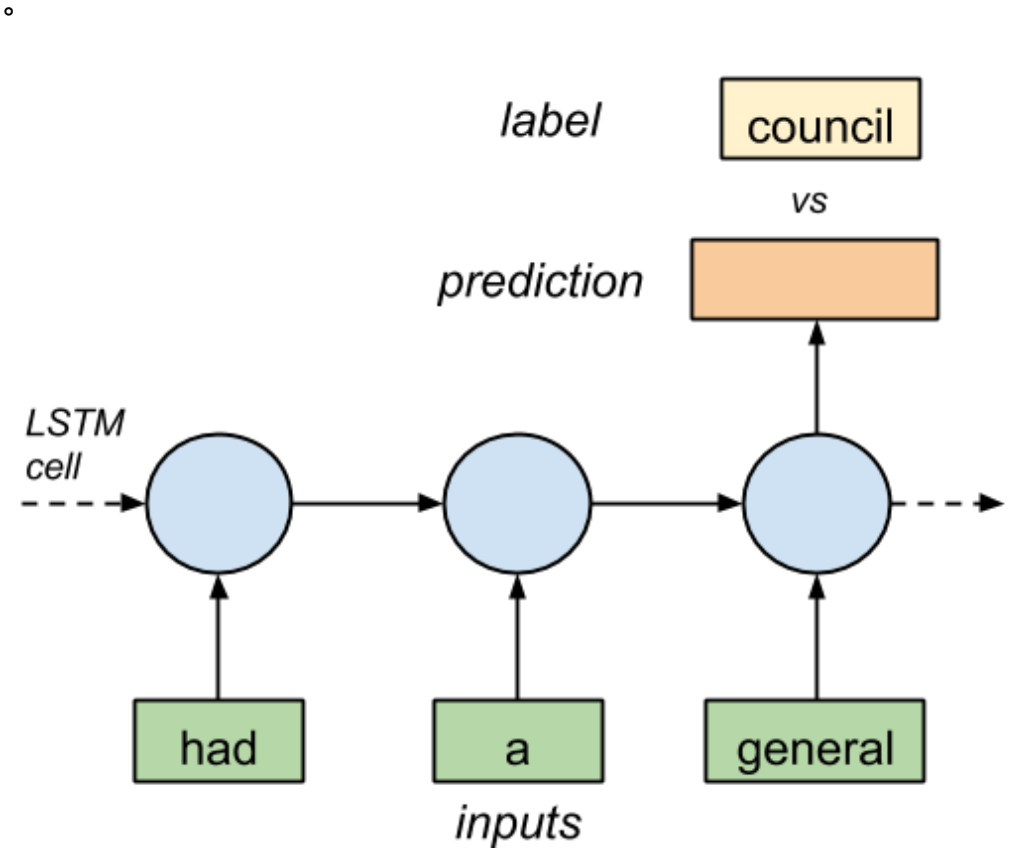


图 1．有 3 个输入和 1 个输出的 LSTM 单元

严格说来，LSTM 只能理解输入的实数。一种将符号转化为数字的方法是基于每个符号出现的频率为其分配一个对应的整数。例如，上面的短文中有 112 个不同的符号。如列表 2 所示的函数建立了一个有如下条目 [「,」: 0] [「the」: 1], …, [「council」: 37],…,[「spoke」= 111] 的词典。而为了解码 LSTM 的输出，同时也生成了逆序字典。

```
def build_dataset(words):
    count = collections.Counter(words).most_common()
    dictionary = dict()
    for word, _ in count:
        dictionary[word] = len(dictionary)
    reverse_dictionary = dict(zip(dictionary.values(), dictionary.keys()))
    return dictionary, reverse_dictionary
```

Listing 2．建立字典和逆序字典的函数

类似地，预测值也是一个唯一的整数值与逆序字典中预测符号的索引相对应。例如：如果预测值是 37，预测符号便是「council」。

输出的生成看起来似乎简单，但实际上 LSTM 为下一个符号生成了一个含有 112 个元素的预测概率向量，并用 softmax() 函数归一化。有着最高概率值的元素的索引便是逆序字典中预测符号的索引值（例如：一个 one-hot 向量）。图 2 给出了这个过程。

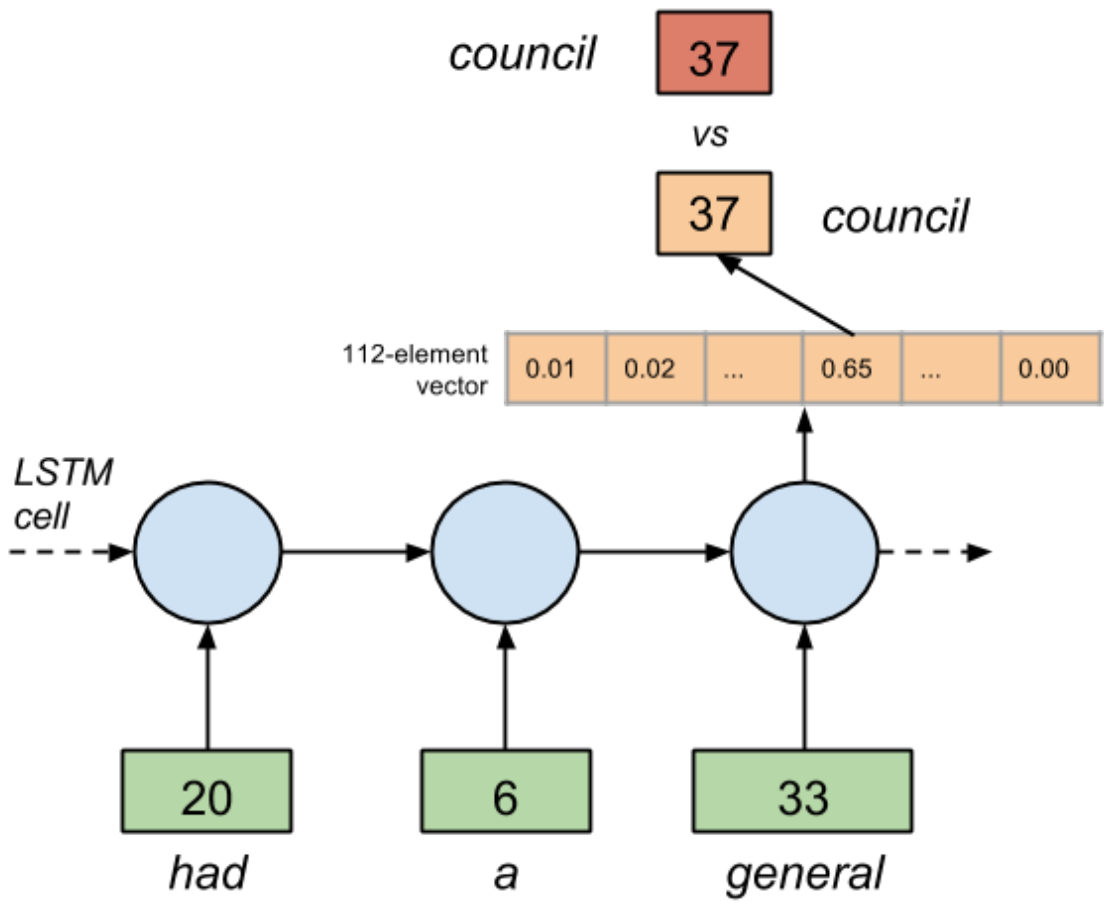


图 2．每一个输入符号被分配给它的独一无二的整数值所替代。输出是一个表明了预测符号在反向词典中索引的 one-hot 向量。

LSTM 模型是这个应用的核心部分。令人惊讶的是，它很易于用 TensorFlow 实现：

```
def RNN(x, weights, biases):

    # reshape to [1, n_input]
    x = tf.reshape(x, [-1, n_input])

    # Generate a n_input-element sequence of inputs
    # (eg. [had] [a] [general] -> [20] [6] [33])
    x = tf.split(x,n_input,1)

    # 1-layer LSTM with n_hidden units.
    rnn_cell = rnn.BasicLSTMCell(n_hidden)

    # generate prediction
    outputs, states = rnn.static_rnn(rnn_cell, x, dtype=tf.float32)

    # there are n_input outputs but
    # we only want the last output
    return tf.matmul(outputs[-1], weights['out']) + biases['out']
```

Listing 3．有 512 个 LSTM 单元的网络模型

最难部分是以正确的格式和顺序完成输入。在这个例子中，LSTM 的输入是一个有 3 个整数的序列（例如：1x3 的整数向量）

网络的常量、权值和偏差设置如下：

```
vocab_size = len(dictionary)
n_input = 3
```

```
# number of units in RNN cell
n_hidden = 512
```

```
# RNN output node weights and biases
weights = {
    'out': tf.Variable(tf.random_normal([n_hidden, vocab_size]))
}
biases = {
    'out': tf.Variable(tf.random_normal([vocab_size]))
}
```

Listing 4．常量和训练参数

训练过程中的每一步，3 个符号都在训练数据中被检索。然后 3 个符号转化为整数以形成输入向量。

```
symbols_in_keys = [ [dictionary[ str(training_data[i])]] for i in range(offset, offset+n_input) ]
```

Listing 5．将符号转化为整数向量作为输入

训练标签是一个位于 3 个输入符号之后的 one-hot 向量。

```
symbols_out_onehot = np.zeros([vocab_size], dtype=float)
symbols_out_onehot[dictionary[str(training_data[offset+n_input])]] = 1.0
```

Listing 6：单向量作为标签

在转化为输入词典的格式后，进行如下的优化过程：

```
_, acc, loss, onehot_pred = session.run([optimizer, accuracy, cost, pred], feed_dict={x: symbols_in_keys, y
: symbols_out_onehot})
```

Listing 7．训练过程中的优化

精度和损失被累积以监测训练过程。通常 50,000 次迭代足以达到可接受的精度要求。

```
...
Iter= 49000, Average Loss= 0.528684, Average Accuracy= 88.50%
['could', 'easily', 'retire'] - [while] vs [while]
Iter= 50000, Average Loss= 0.415811, Average Accuracy= 91.20%
['this', 'means', 'we'] - [should] vs [should]
```

Listing 8．一个训练间隔的预测和精度数据示例（间隔 1000 步）

代价是标签和 softmax() 预测之间的交叉熵，它被 RMSProp 以 0.001 的学习率进行优化。在本文示例的情况中，RMSProp 通常比 Adam 和 SGD 表现得更好。

```
pred = RNN(x, weights, biases)

# Loss and optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=pred, labels=y))
optimizer = tf.train.RMSPropOptimizer(learning_rate=learning_rate).minimize(cost)
```

Listing 9．损失和优化器

LSTM 的精度可以通过增加层来改善。

```
rnn_cell = rnn.MultiRNNCell([rnn.BasicLSTMCell(n_hidden),rnn.BasicLSTMCell(n_hidden)])
```

Listing 10. 改善的 LSTM

现在，到了有意思的部分。让我们通过将预测得到的输出作为输入中的下一个符号输入 LSTM 来生成一个故事吧。示例输入是「had a general」，LSTM 给出了正确的输出预测「council」。然后「council」作为新的输入「a general council」的一部分输入神经网络得到下一个输出「to」，如此循环下去。令人惊讶的是，LSTM 创作出了一个有一定含义的故事。

```
had a general council to consider what measures they could take to outwit their common enemy , the cat . so
me said this , and some said that but at last a young mouse got
```

Listing 11．截取了样本故事生成的故事中的前 32 个预测值

如果我们输入另一个序列（例如：「mouse」,「mouse」,「mouse」）但并不一定是这个故事中的序列，那么会自动生成另一个故事。

```
mouse mouse mouse , neighbourhood and could receive a outwit always the neck of the cat . some said this ,
and some said that but at last a young mouse got up and said
```

Listing 12．并非来源于示例故事中的输入序列

示例代码可以在这里找到：https://github.com/roatienza/Deep-Learning-Experiments/blob/master/Experiments/Tensorflow/RNN/rnn_words.py

示例文本的链接在这里：https://github.com/roatienza/Deep-Learning-Experiments/blob/master/Experiments/Tensorflow/RNN/belling_the_cat.txt

小贴士：

1. 用整数值编码符号容易操作但会丢失单词的意思。本文中将符号转化为整数值是用来简化关于用 TensorFlow 建立 LSTM 应用的讨论的。更推荐采用 Word2Vec 将符号编码为向量。

2. 将输出表达成单向量是效率较低的方式，尤其当我们有一个现实的单词量大小时。牛津词典有超过 170,000 个单词，而上面的例子中只有 112 个单词。再次声明，本文中的示例只为了简化讨论。

3. 这里采用的代码受到了 Tensorflow-Examples 的启发：https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/3_NeuralNetworks/recurrent_network.py

4. 本文例子中的输入大小为 3，看一看当采用其它大小的输入时会发生什么吧（例如：4，5 或更多）。

5. 每次运行代码都可能生成不同的结果，LSTM 的预测能力也会不同。这是由于精度依赖于初始参数的随机设定。训练次数越多（超过 150,000 次）精度也会相应提高。每次运行代码，建立的词典也会不同

6. Tensorboard 在调试中，尤其当检查代码是否正确地建立了图时很有用。

7. 试着用另一个故事测试 LSTM，尤其是用另一种语言写的故事。SYNCED

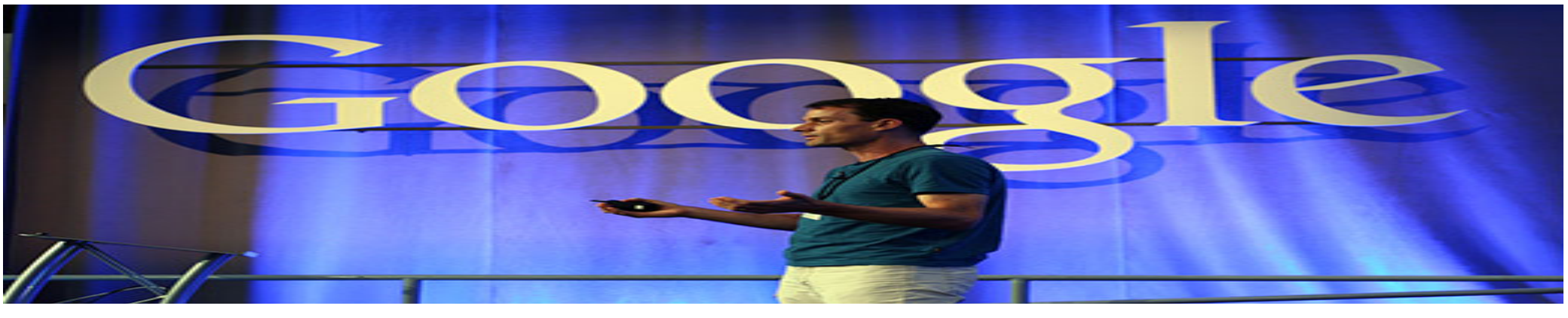
原文链接：https://medium.com/towards-data-science/lstm-by-example-using-tensorflow-feb0c1968537

参与成员：

(/user/author/index/pid/najnmfo0o0o6)

吴攀

相关文章



Tensor Flow开源后，第三方开发者们都捣鼓了些什么？

(/article/428)

评论

💬共有0条评论，点击展开

(/user/author/index/pid/NajnMFO0O0O6)
吴攀
邮箱 /pid/NajnMFO0O0O6)

204
文章 (/user/author/index/pid/NajnMFO0O0O6)
0
评论 (/user/author/comment/pid/NajnMFO0O0O6)
0
收藏 (/user/author/favorite/pid/NajnMFO0O0O6)

24小时最热 一周最热

微软**RobustFill**：无需编程语言，让神...
2017年04月22日
(/article/2691)

三张图读懂机器学习：基本概念、五大流派与九种...
2017年04月22日
(/article/2690)

用人工智能做金融风控？这里是一位实践者的思考
约23时前
(/article/2715)

绿公司年会 | 依图创始人朱珑：**AI**无与伦比...
2017年04月24日
(/article/2710)

谷歌**TPU**之后还有高通，人工智能芯片竞赛已经...
约23时前
(/article/2713)