深度学习1

pytorch对可变长度序列的处理

主要是用函数torch.nn.utils.rnn.

PackedSequence()和 torch.nn.utils.rnn. pack_padded_sequence()以及 torch.nn.utils.rnn. pad_packed_sequence ()来进行的,分别来看看这三个函数的用法。

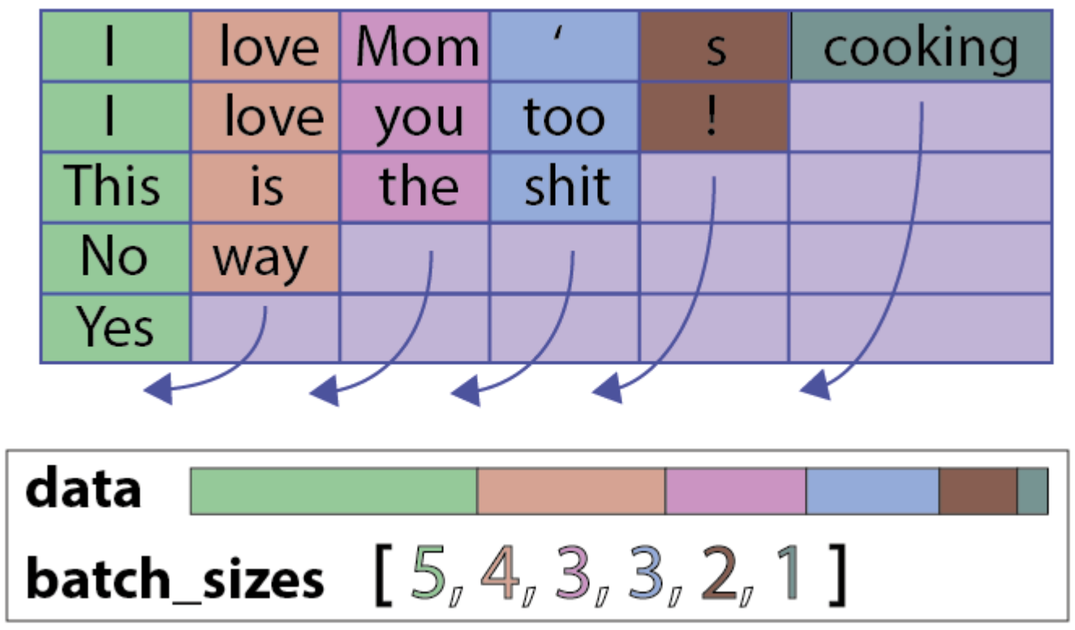
1、 torch.nn.utils.rnn. PackedSequence()

NOTE: 这个类的实例不能手动创建。它们只能被 pack_padded_sequence() 实例化。

PackedSequence对象包括:

- 一个 data 对象:一个torch.Variable(令牌的总数,每个令牌的维度),在这个简单的例子中有五个令牌序列(用整数表示):(18,1)
- 一个 batch_sizes 对象:每个时间步长的令牌数列表,在这个例子中为:[6,5,2,4,1]

用pack_padded_sequence函数来构造这个对象非常的简单:



如何构造一个PackedSequence对象(batch_first = True)

PackedSequence对象有一个很不错的特性,就是我们无需对序列解包(这一步操作非常慢)即可**直接在PackedSequence数据变量上**执行许多操作。特别是我们可以对令牌执行任何操作(即对令牌的顺序/上下文不敏感)。当然,我们也可以使用接受PackedSequence作为输入的任何一个pyTorch模块(pyTorch 0.2)。

2、 torch.nn.utils.rnn. pack_padded_sequence()

这里的 pack ,理解成压紧比较好。 将一个 填充过的变长序列 压紧。(填充时候,会有冗余,所以压紧一下)

输入的形状可以是(T×B×*)。 T 是最长序列长度, B 是 batch size , * 代表任意维度(可以是0)。如果 batch_first=True 的话,那么相应的 input size 就是 (B×T×*) 。

Variable 中保存的序列,应该按序列长度的长短排序,长的在前,短的在后。即 input[:,0] 代表的是最长的序列, input[:, B-1] 保存的是最短的序列。

NOTE: 只要是维度大于等于2的 input 都可以作为这个函数的参数。你可以用它来打包 labels ,然后用 RNN 的输出和打包后的 labels 来计算 loss 。通过 PackedSequence 对象的 .data 属性可以获取 Variable 。

参数说明:

- input (Variable) – 变长序列 被填充后的 batch
- lengths (list[int]) – Variable 中 每个序列的长度。
- batch_first (bool, optional) – 如果是 True ,input的形状应该是 B*T*size 。

返回值:

一个 PackedSequence 对象。

3、 torch.nn.utils.rnn. pad_packed_sequence ()

填充 packed_sequence 。

上面提到的函数的功能是将一个填充后的变长序列压紧。 这个操作和pack_padded_sequence()是相反的。把压紧的序列再填充回来。

返回的Varaible的值的 size 是 T×B×* , T 是最长序列的长度, B 是 batch_size,如果 batch_first=True ,那么返回值是 B×T×* 。

Batch中的元素将会以它们长度的逆序排列。

参数说明:

- sequence (PackedSequence) – 将要被填充的 batch
- batch_first (bool, optional) – 如果为True,返回的数据的格式为 B×T×* 。

返回值: 一个tuple,包含被填充后的序列,和batch中序列的长度列表。

例子:

```
import torch
import torch.nn as nn
from torch.autograd import Variable
from torch.nn import utils as nn_utils
batch_size = 2
max_length = 3
hidden_size = 2
n_layers =1

tensor_in = torch.FloatTensor([[1, 2, 3], [1, 0, 0]]).resize_(2,3,1)
tensor_in = Variable( tensor_in ) #[batch, seq, feature], [2, 3, 1]
seq_lengths = [3,1] # list of integers holding information about the batch size at each sequence step

# pack it
pack = nn_utils.rnn.pack_padded_sequence(tensor_in, seq_lengths, batch_first=True)

# initialize
rnn = nn.RNN(1, hidden_size, n_layers, batch_first=True)
```

```
h0 = Variable(torch.randn(n_layers, batch_size, hidden_size))

#forward
out, _ = rnn(pack, h0)

# unpack
unpacked = nn_utils.rnn.pad_packed_sequence(out)
print('111',unpacked)
```

输出：

```
111 (Variable containing:
(0 ,.,.) =
  0.5406  0.3584
 -0.1403  0.0308

(1 ,.,.) =
 -0.6855 -0.9307
  0.0000  0.0000
[torch.FloatTensor of size 2x2x2]
, [2, 1])
```


深度学习1
关注 - 2
粉丝 - 19
+加关注

« 上一篇： pytorch函数之torch.normal()
» 下一篇： pytorch之LSTM

posted @ 2017-12-17 14:32 深度学习1 阅读(26262) 评论(7) 编辑 收藏 举报