

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。

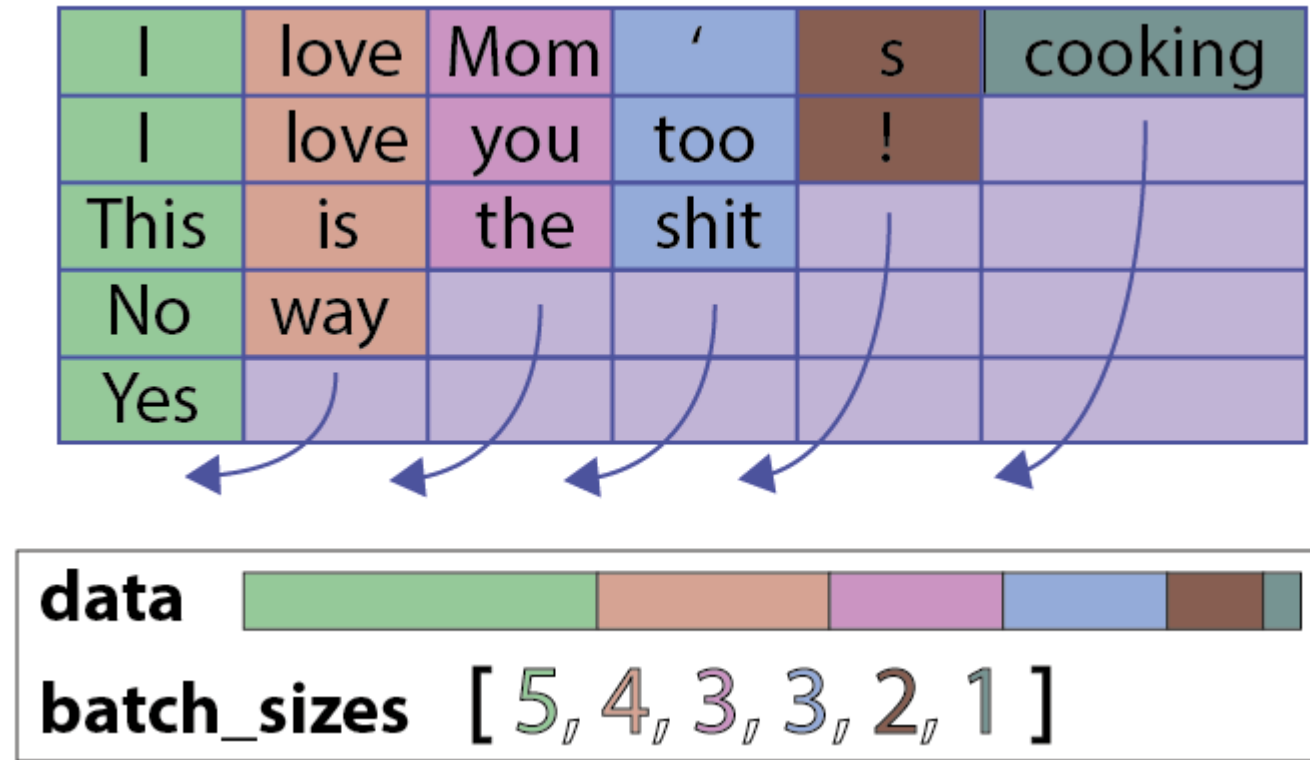
1、 `torch.nn.utils.rnn.PackedSequence()`

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Variable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:

```
1 import torch
2 import torch.nn as nn
3 from torch.autograd import Variable
4 from torch.nn import utils as nn_utils
5 batch_size = 2
6 max_length = 3
7 hidden_size = 2
8 n_layers = 1
9
10 tensor_in = torch.FloatTensor([[1, 2, 3], [1, 0, 0]]).resize_(2,3,1)
11 tensor_in = Variable( tensor_in ) #[batch, seq, feature], [2, 3, 1]
12 seq_lengths = [3,1] # list of integers holding information about the batch size at each sequence step
13
14 # pack it
15 pack = nn_utils.rnn.pack_padded_sequence(tensor_in, seq_lengths, batch_first=True)
16
17 # initialize
18 rnn = nn.RNN(1, hidden_size, n_layers, batch_first=True)
```

[博客园](#)

[首页](#)

[新随笔](#)

[联系](#)

[订阅](#)

[管理](#)

随笔 - 63 文章 - 0 评论 - 25 阅读 - 52万

公告

昵称: 深度学习1
园龄: 4年9个月
粉丝: 19
关注: 2
+加关注

<	2022年5月						>
日	一	二	三	四	五	六	
1	2	3	4	5	6	7	
8	9	10	11	12	13	14	
15	16	17	18	19	20	21	
22	23	24	25	26	27	28	
29	30	31	1	2	3	4	
5	6	7	8	9	10	11	

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔档案

2018年1月(3)
2017年12月(47)
2017年11月(10)
2017年10月(2)
2017年7月(1)

阅读排行榜

1. pytorch梯度裁剪 (Clipping Gradient) : torch.nn.utils.clip_grad_norm (59050)
2. torch.nn.Embedding(55577)
3. np.savetxt()——将array保存到txt文件, 并保持原格式(53887)
4. Python3实现从文件中读取指定行的方法(47480)
5. pytorch函数之torch.normal()(31300)

评论排行榜

1. torch.nn.Embedding(8)
2. pytorch对可变长度序列的处理(7)
3. pip pytorch安装时出现的问题(4)
4. pytorch梯度裁剪 (Clipping Gradient) : torch.nn.utils.clip_grad_norm (3)
5. np.savetxt()——将array保存到txt文件, 并保持原格式(1)

推荐排行榜

1. python学习之argparse模块的使用(7)
2. torch.nn.Embedding(6)
3. np.savetxt()——将array保存到txt文件, 并保持原格式(2)
4. PyTorch学习系列(九)——参数_初始化(2)
5. Python的zip函数(1)

最新评论

1. Re:torch.nn.Embedding
有什么好纠结的? 源码里面不是写着的嘛 if _weight is None: self.weight = Parameter(torch.Tensor(num_embeddings, embeddin... --影醉阁轩窗

```
19 | h0 = Variable(torch.randn(n_layers, batch_size, hidden_size))
20 |
21 | #forward
22 | out, _ = rnn(pack, h0)
23 |
24 | # unpack
25 | unpacked = nn_utils.rnn.pad_packed_sequence(out)
26 | print('111',unpacked)
```

输出:

```
1 | 111 (Variable containing:
2 | (0 ,.,.) =
3 |  0.5406  0.3584
4 | -0.1403  0.0308
5 |
6 | (1 ,.,.) =
7 | -0.6855 -0.9307
8 |  0.0000  0.0000
9 | [torch.FloatTensor of size 2x2x2]
10 | , [2, 1])
```

好文要顶

关注我

收藏该文

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

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

编辑推荐:

- 阴影进阶，实现更加的立体的阴影效果！
- 如何写好B端产品的技术方案？
- [架构视角] 一篇文章带你彻底吃透Spring
- .NET性能优化-你应该为集合类型设置初始大小
- C#语法糖系列 —— 第二篇：聊聊 ref, in 修饰符底层玩法

最新新闻:

- 高管离职，团队解散，苹果造车真的凉凉了？
- 美国科技巨头的“中年危机”是如何形成的？
- Github开发大神教你玩转数据库编程
- 宇宙中的暗物质都去哪里了？
- 花了一年投资美股，赚得还不如朋友买辆车多
- » 更多新闻...

2. Re:pytorch梯度裁剪（Clipping Gradient）：torch.nn.utils.clip_grad_norm
如果是多个loss分别backward的话，请问是backward一次就要做一次clip_grad_norm吗？还是说backward多次，做一次就可以了？
--kkarl

3. Re:torch.nn.Embedding
@CrazyNong这个只是一个简单的例子，可以和你清楚的明白embedding的使用，若是置于一个完整的翻译网络结构中，反向传播的梯度更新不就把所设计到的网络参数都更新一遍吗...
--wangyy161

4. Re:pytorch梯度裁剪（Clipping Gradient）：torch.nn.utils.clip_grad_norm
@张贤同学 以范数大小作为分类依据，针对梯度消失的梯度裁剪，范数小于阈值的梯度就把它的范数缩放到阈值...
--Ghost#echo

5. Re:pytorch梯度裁剪（Clipping Gradient）：torch.nn.utils.clip_grad_norm
所以裁剪和范数有什么关系，为什么要设置范数
--张贤同学

10