案例5：使用BiLSTM对IMDB做分类

2023 年 4 月 26 日

由于本框架计算太慢，句子长度不能设置太大，实际上大多数句子长度
都超过200了，仅仅依靠前面几个词根本无法预测，这里仅仅是展示模型是否跑的通
后面的解释可能会按256个词来解释

本案例为了能跑通，更改了以下几个参数：
1.max_seq_len=256 –> 8
2.vocab_lens=len(loader.dict) –> 20 (这样单词id会溢出词典，注意）
3.训练的样本数量：N= len(seqs) –> 12

[1]:

```
import sys
sys.path.append(r"D:\Rhitta_GPU")
import numpy as np
import cupy as cp
import rhitta.nn as nn
```

第一步：载入数据集

会获得两个列表，分别是句子和标签
IMDBLoader()接口接收一个指定句子长度的参数max_seq_len

[2]:

```
max_seq_len = 8 # 设置每个句子的长度
loader=nn.IMDBLoader(max_seq_len=max_seq_len)
seqs,labels=loader.load(r"D:\Rhitta_GPU\data\dataset")
```

[3]:

```
## test
for i in range(10):
    print(seqs[i],labels[i])
```

```
[552, 35, 14, 554, 171, 188, 34, 2] 1
[16, 5344, 946, 5, 2, 10059, 33, 3898] 1
```

```
[16， 21， 529， 19， 4， 3179， 4752， 5200] 0
[41， 218， 30， 5201， 12， 162， 37， 5812] 0
[25376， 4556， 3， 44405， 12118， 13， 2284， 37792] 1
[10， 6192， 123， 176， 83， 105， 14， 7] 1
[49， 18， 97， 29， 77， 54， 50， 22] 0
[10， 301， 14， 398， 34， 4， 464， 13] 0
[112， 464， 5， 1966， 1249， 14， 21， 20] 0
[8， 8， 49， 18， 7， 407， 5， 2124] 1
```

第二步：构造模型

先把句子的256个词，也就是256个数字丢进embedding层，变成256个向量，
再把256个向量丢进BiLSTM中，获得一个输出，最后送入分类头
注意：BiLSTM最后的汇聚层需要忽略掉pad过来的向量，
由于实现起来有些繁琐，这里就不忽略了，影响不大
不忽略相当于后面的神经元用于传递之前的信息，没有新信息加入

[4]:

```python
class zyw(nn.Module):
    def __init__(self):
        super(zyw,self).__init__()
        self.bilstm=nn.BiLSTM(input_size=6,hidden_size=4,time_dimension=max_seq_len,mode=1)
        self.linear=nn.Linear(8,1,activation = "Logistic") # 注意BiLSTM把隐藏层拼接了，向量维度变成2倍了
    def __call__(self,seq_embeddings,h_0,c_0,h_1,c_1):
        x=self.bilstm(seq_embeddings,h_0,c_0,h_1,c_1)
        x=self.linear(x)
        return x
vocab_lens=len(loader.dict)
vocab_lens=20  # 字典太大训练不动，但是取词的时候，词的id很容易超过这个数
embedding=nn.Embedding(numembeddings=vocab_lens, embeddingdim=6, paddingidx=0)
model = zyw()
```

第三步：构造计算图

坑节点包括：

句子列表：必须是一个固定不动的对象，后面需要往里面填写每个句子的数字

embedding一旦实例化，就不能变动，只能改输入对象的内部数值

初始隐藏状态节点：由于是双向LSTM，需要4个，形状(1,4)
标签节点：由于是二分类，形状为(1,1)

[5]:
```
# 构造坑位，注意，叶子节点不是输入的列表，而是编码器里面的词典，已经自动创建好了
# 当词典更新set_value时，所有下游节点全部reset_value
seq = [i for i in range(max_seq_len)]
h_0,c_0,h_1,c_1=nn.to_tensor((1,4)),nn.to_tensor((1,4)),nn.to_tensor((1,4)),nn.
↪to_tensor((1,4))
label = nn.to_tensor((1,1))

# 构造计算图
seq_embedding = embedding(seq)
output = model(seq_embedding,h_0,c_0,h_1,c_1)
loss = nn.BinaryClassLoss(output,label)
```

第四步：初始化优化器

[6]:
```
learning_rate = 0.01
optimizer = nn.Adam(nn.default_graph, loss, learning_rate=learning_rate)
```

第五步：开始训练

[7]:
```
batch_size = 2 # 因为只拿12个句子，这里batch_size如果取16，模型就不更新了
epochs = 3
print("更新前的随机词典：")
print(embedding.vocab.value)
for epoch in range(epochs):
    count = 0
    N= len(seqs)
    N = 12 # 就拿前10条句子跑吧，否则还是跑不动

    # 填坑并训练
    for i in range(N):
        # 句子的列表对象填坑
        for j in range(max_seq_len):
```

```
        if seqs[i][j] < vocab_lens :
            seq[j]=seqs[i][j]
        else:
            seq[j]=0
    # 输入隐藏状态
    h_0.set_value(cp.zeros((1, 4)))
    c_0.set_value(cp.zeros((1, 4)))
    h_1.set_value(cp.zeros((1, 4)))
    c_1.set_value(cp.zeros((1, 4)))
    # 输入标签
    label.set_value(labels[i])
    # 前向反向传播
    optimizer.one_step()
    # 更新计数器
    count += 1
    # 计数器达到batch_size就更新模型参数
    if count >= batch_size:
        optimizer.update()
        count = 0

# 每个epoch后评估模型的平均平方损失
acc_loss = 0
for i in range(N):
    for j in range(max_seq_len):
        if seqs[i][j] < vocab_lens :
            seq[j]=seqs[i][j]
        else:
            seq[j]=0
    h_0.set_value(cp.zeros((1, 4)))
    c_0.set_value(cp.zeros((1, 4)))
    h_1.set_value(cp.zeros((1, 4)))
    c_1.set_value(cp.zeros((1, 4)))
    label.set_value(labels[i])
    loss.forward()
    acc_loss += loss.value
average_loss = acc_loss / N
```

```
    print("epoch:{} , average_loss:{}".format(epoch+1, cp.
↪sqrt(average_loss)[0][0]))
print("更新后的词典： ")
print(embedding.vocab.value)
```

```
更新前的随机词典：
[[ 0.          0.          0.          0.          0.          0.        ]
 [ 0.04601878  0.00404048  0.04015696 -0.06519445 -0.03926611  0.0139227 ]
 [-0.06951123  0.08241394  0.02143262  0.06112293  0.01719459 -0.03234663]
 [-0.00495689 -0.00952639 -0.08017523 -0.0049339   0.06948158 -0.01158497]
 [-0.05801717 -0.04864316 -0.01222767  0.09520323 -0.0561946   0.0125205 ]
 [-0.09940845  0.03058567 -0.06553359 -0.09607915 -0.03470116 -0.07746006]
 [ 0.02574757  0.08103257 -0.00861659 -0.08763861  0.00053078  0.06172405]
 [-0.01261335 -0.01195699 -0.00790909  0.06745773  0.01438359 -0.0432668 ]
 [-0.07392157  0.00809994  0.08484132  0.08505075 -0.01008637  0.06179531]
 [ 0.0030789   0.0487923   0.02765147 -0.02074502  0.0231011   0.09824966]
 [-0.03578965 -0.08761168  0.02594208 -0.01307079  0.06163082 -0.01358586]
 [ 0.05578862  0.02245893  0.092424   -0.05362544 -0.00972471 -0.04428356]
 [ 0.07484926 -0.02366823 -0.09911224  0.04430081 -0.00841923 -0.00920663]
 [ 0.02676214  0.06327197  0.08703353  0.06904646 -0.03202954  0.03444296]
 [-0.01285214  0.07879407  0.04201142  0.00959265  0.06963184  0.01009394]
 [-0.08374249 -0.05770571 -0.02031106 -0.01629067  0.05843795 -0.07895623]
 [-0.04759348 -0.08748662  0.00497911  0.00683976 -0.0933935  -0.0852092 ]
 [-0.02791182 -0.06868168 -0.05379565 -0.00373487 -0.04043877  0.0174353 ]
 [-0.08165128 -0.03259038 -0.09113629 -0.08790202  0.00799482 -0.04439018]
 [ 0.0215529   0.01567511 -0.06697443  0.02731935 -0.03042926  0.07366363]]
epoch:1 , average_loss:0.9893749000353604
epoch:2 , average_loss:0.8663498296578028
epoch:3 , average_loss:0.8411084832551297
更新后的词典：
[[-0.05649016 -0.04927701 -0.04806061  0.15907731  0.00178527  0.05822795]
 [-0.00156807 -0.06583477  0.15503936  0.04629727 -0.00317504  0.07930903]
 [-0.14579108  0.02817459  0.06145587  0.22798977  0.09353109  0.02435572]
 [-0.01749295 -0.10734621 -0.14469249 -0.04212084  0.12125494  0.05556165]
 [-0.01051013 -0.11818024 -0.0661573  -0.00720834  0.01919932  0.04678992]
 [-0.06861676 -0.06855189 -0.10726222 -0.27821925  0.01145983 -0.01735449]
 [-0.1242651   0.13774998 -0.00981059 -0.02559029  0.03498846  0.07474569]
[-0.07225565 -0.00482027  0.07053885  0.14182547  0.08221801  0.02598057]
[-0.07392157  0.00809994  0.08484132  0.08505075 -0.01008637  0.06179531]
[ 0.0030789   0.0487923   0.02765147 -0.02074502  0.0231011   0.09824966]
[-0.03578965 -0.08761168  0.02594208 -0.01307079  0.06163082 -0.01358586]
[ 0.05578862  0.02245893  0.092424   -0.05362544 -0.00972471 -0.04428356]
[ 0.07484926 -0.02366823 -0.09911224  0.04430081 -0.00841923 -0.00920663]
[ 0.02676214  0.06327197  0.08703353  0.06904646 -0.03202954  0.03444296]
[-0.01285214  0.07879407  0.04201142  0.00959265  0.06963184  0.01009394]
[-0.08374249 -0.05770571 -0.02031106 -0.01629067  0.05843795 -0.07895623]
[-0.04759348 -0.08748662  0.00497911  0.00683976 -0.0933935  -0.0852092 ]
[-0.02791182 -0.06868168 -0.05379565 -0.00373487 -0.04043877  0.0174353 ]
[-0.08165128 -0.03259038 -0.09113629 -0.08790202  0.00799482 -0.04439018]
[ 0.0215529   0.01567511 -0.06697443  0.02731935 -0.03042926  0.07366363]]
```