第四天

2019年6月11日 9:30

文本情感分类

1. 准备数据
    1. 如何完成基础打Dataset的构建和Dataloader的准备
        i. 注意点：dataloader中的collate_fn的实现
            1) collate_fn(batch):batch中是一个个的getItem的结果
    2. 每个batch中文本的长度不一致的问题如何解决
        i. 长句子裁剪
        ii. 短句子填充
    3. 每个batch中的文本如何转化为数字序列
        i. {“词”:int}

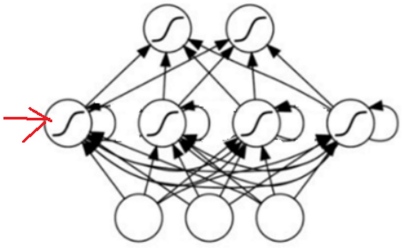
RNN：具有短期记忆的网络

1. 和普通网络的区别
    1. 能够更擅长解决时间序列问题
    2. 当前时刻神经元的输入有两个：
        i. 当前时刻的输入
        ii. 前一时刻的输出
2. RNN的常见类别
    1. one-one:图像分类
    2. one-many：图像描述
    3. many-one：文本的分类
    4. many-many：翻译
3. RNN的长依赖：带预测词语的位置比较远，很难影响到当前时间步的输出
4. LSTM：解决长依赖
    1. Cell state：细胞状态，存储的是记忆信息
    2. 遗忘门：决定什么信息会被忘记
    3. 输入门：把信息输入到cell state中
    4. 输出门：决定什么信息会被输出
5. GRU：更新门+ 输出门
6. 双向LSTM：让每个时间步既包含正向的信息，也包含反向的信息

pytorch中RNN的API

1. LSTM：
    1. torch.nn.LSTM

为什么LSTM默认batch_first=False
需要一个【seq_len,batch_size,embedding_dim】

2. 

lstm = LSMT(input_size=embedding_dim,hidden_size=4,num_layers=1,batch_first=True,bidriectional=False)
初始化 h_0,C_0
调用lstm：lstm(input,(h_0,c_0))
output,(h_n,c_n):
output:[batch_size,seq_len,4]
h_n,c_n,h_0,c_0： [1,batch_size,4]
embedding: [batch_size,seq_len,embeding_dim]
input:[batch_size,seq_len]

3. 双lstm中数据的拼接顺序
    i. **output：按照正反计算的结果顺序在第2个维度进行拼接，正向第一个拼接反向的最后一个输出**
    ii. **hidden state:按照得到的结果在第0个维度进行拼接，正向第一个之后接着是反向第一个**

第四天重点：

1. collate_fn(batch):
    1. batch ---> [（一个getitem的结果），() ()]
2. word sequence：实现方法把字符串转化为数字序列
    1. 文本padding，让batch中文本长度一致
    2. 对词频进行过滤
    3. 字符串--》数字，数字--》字符串
3. RNN
    1. 是什么：具有短期记忆的网络
        i. 为什么具有短期记忆：把前一次的输出作为当前的输入
    2. LSTM：能够解决长依赖问题
        i. 使用三个门结构实现长依赖
            1) 遗忘门：决定信息的遗忘
            2) 输入门：输入信息
            3) 输出门：输出信息
        ii. 门的理解：通过sigmoid和参数控制输出输入和输出的大小
    3. GRU和双向LSTM，双向GRU和LSTM的区别
        i. GRU2个门：
            1) 更新门
            2) 输出门
        ii. 双向LSTM、GRU
            1) 正向的第一个输出和方向的最后一个输出进行拼接
            2) 结果同时包含正向和方向信息

4. pytorch中LSTM的API
    i. torch.nn.LSTM(input_size,hidden_size,num_layers,batch_first,dropout,bidirectional)
    ii. 经常会用LSTM的最后一个时间步的输出表示文本