|  |  |
| --- | --- |
| 成绩 |  |

装

订

线

网络空间安全与计算机学院实验报告

实验课程名称**：神经网络与深度学习实验**

实验项目名称： 卷积神经网络（3）LSTM文本分类

学生姓名： 刘新媛

专 业： 人工智能

学 号： 20221205037

实验地点： C1-320

实验时间： 2024.12.11第7-8节

指导教师： 魏勇刚

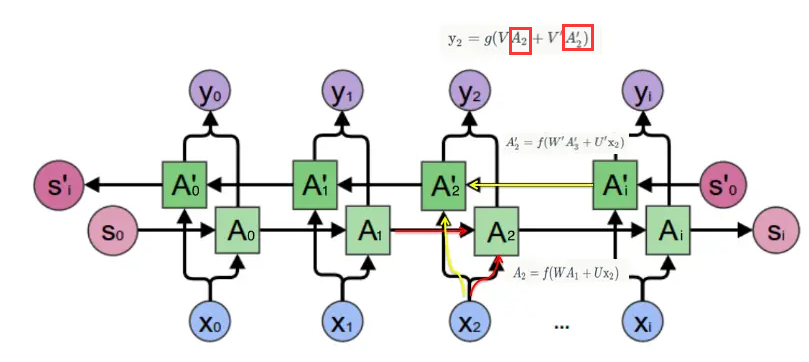
预习报告部分

1. **实验目的：**

基于双向LSTM模型完成文本分类任务实验

基于双向LSTM和注意力机制完成文本分类任务实验

1. **实验原理：**
2. **双向 LSTM （BiLSTM）**是在 LSTM 的基础上发展而来的。它的主要特点是能够同时处理正向和反向的序列信息。对于一个序列数据.....,，单向 LSTM 只能按照从序列的**开始到结束**的顺序来处理信息，而双向 LSTM 会同时从序列的开头和结尾进行处理。另一个是反向的 LSTM 层，按照从序列的**结束到开始**的顺序处理输入序列，得到一系列反向的隐藏状态。最终的输出是将正向和反向的隐藏状态进行组合。



1. 注意力机制(attention):注意力机制能通过**权重分布**从大量的候选信息中选择与特定任务更相关或更重要的信息。在基于双向LSTM和注意力机制的文本分类模型中，注意力机制被应用于LSTM层的输出，从隐状态中选择有用的信息，然后将其传递给分类层。在所有序列元素上计算注意力分布，涉及到一个查询向量和注意力打分函数。之后再使用Softmax函数对注意力打分值进行归一化，得到注意力分布（也称为注意力权重）。
2. **实验内容：**

**[点击查看作业内容](https://blog.csdn.net/qq_38975453/article/details/126800118?ops_request_misc=&request_id=&biz_id=102&utm_term=NNDLLSTM%E5%AE%9E%E7%8E%B0%E6%96%87%E6%9C%AC%E5%88%86%E7%B1%BB&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-12)**

1. **实验所用设备：**pycharm、pytorch

实验报告部分

1. **实验步骤：**

* **基于双向LSTM模型完成文本分类任务**

1. **数据集处理-IMDB 电影评论数据集**

IMDB 电影评论数据集是一份关于电影评论的经典二分类数据集．IMDB 按照评分的高低筛选出了积极评论和消极评论，如果评分 ≥7，则认为是积极评论；如果评分 ≤4，则认为是消极评论。数据集包含训练集和测试集数据，数量各为 25000 条，每条数据都是一段用户关于某个电影的真实评价，以及观众对这个电影的情感倾向。

对数据集的处理为：划分训练集、验证集、测试集----->构造Dataset类，实现words\_to\_id方法将序列中的每个词映射为对应的**数字编号**----->封装DataLoader，同时通过collate\_fn函数完成**长度限制和长度补齐**这两个主要功能。

1. **模型构建**

实验的整个模型结果如图1所示。

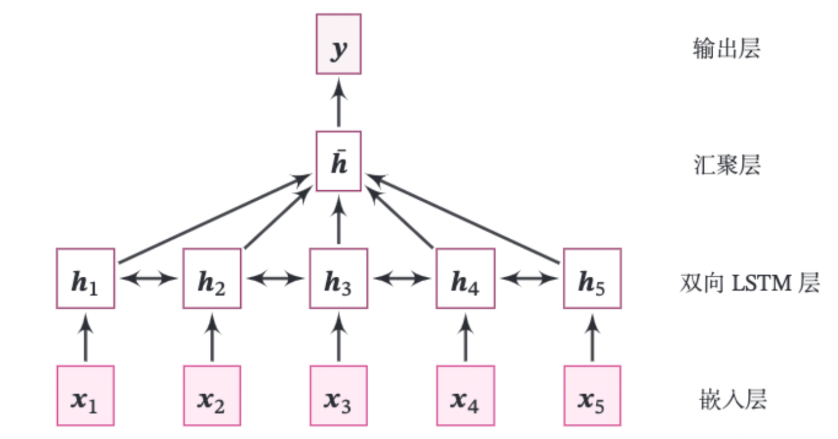


图 1 基于双向LSTM的文本分类模型结构

**（1）嵌入层：**将输入的数字序列（单词映射成的ID）进行向量化，即将每个数字映射为向量。 ---------------**使用Pytorch API：torch.nn.Embedding来完成**

**（2）双向LSTM层：**接收向量序列，分别用前向和反向更新循环单元。----------------**使用Pytorch API：torch.nn.LSTM来完成【**在**定义LSTM时设置参数bidirectional为True**，可**使用双向LSTM。】**

**（3）汇聚层：**将双向LSTM层所有位置上的隐状态进行平均，作为整个句子的表示。首先利用序列长度向量生成**掩码（Mask）矩阵**，用于对文本序列中[PAD]位置的向量进行掩 蔽，然后将该序列的向量进行相加后取均值。具体流程见图2所示。

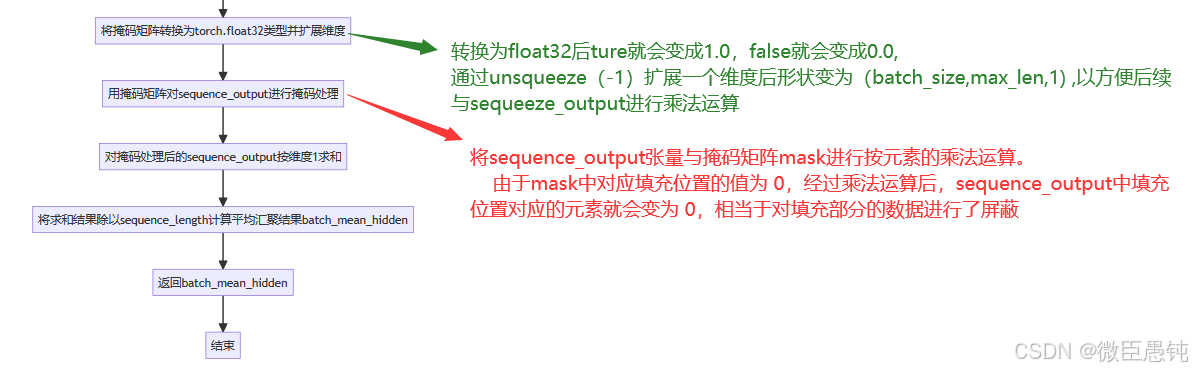


图 2 汇聚层实现流程

1. **输出层：**输出分类的几率。----------**调用torch.nn.Linear来完成。**
2. **模型训练**

基于RunnerV3进行训练，首先指定模型训练的超参，然后设定模型、优化器、损失函数和评估指标，其中损失函数使用nn.CrossEntropyLoss，该损失函数内部会对预测结果使用softmax进行计算，数字预测模型输出层的输出logits不需要使用softmax进行归一化，定义完Runner的相关组件后，便可以进行模型训练，具体如图3所示。

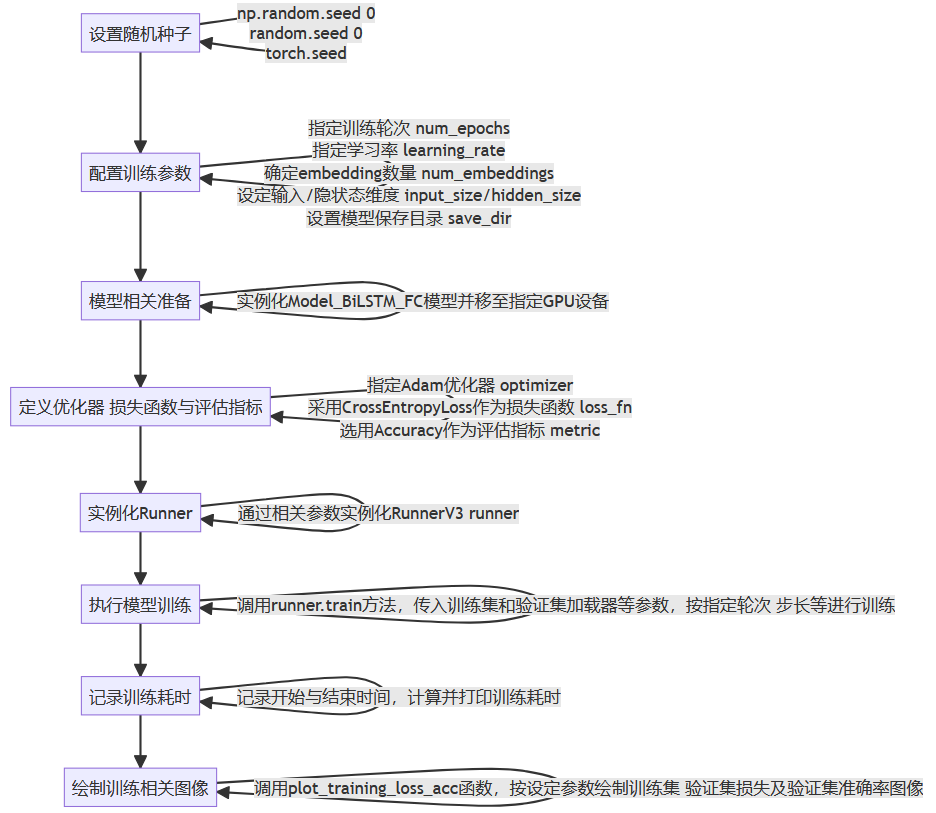


图 3 模型训练流程

1. **模型评价**

加载训练过程中效果最好的模型，然后使用测试集进行测试。

1. **模型预测**

给定任意的一句话，使用训练好的模型进行预测，判断这句话中所蕴含的情感极性。

* **基于双向LSTM和attention完成文本分类任务**

1. **数据集处理--IMDB 电影评论数据集**

同“基于双向LSTM模型完成文本分类任务”的处理步骤一致。（记得打乱训练集）

1. **模型构建**

整体模型框架如图4所示。

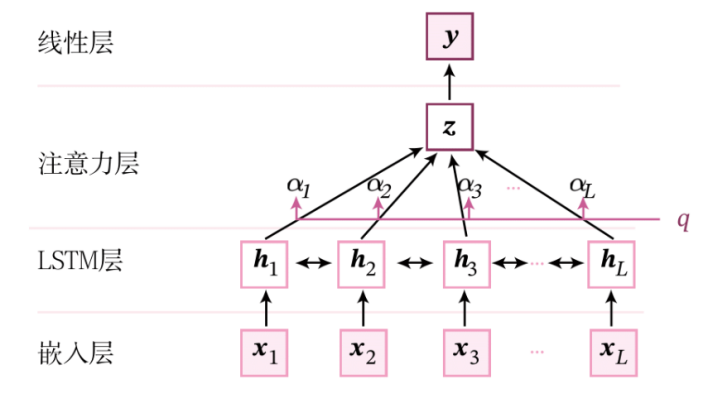


图 4 基于双向LSTM和注意力机制的文本分类模型

****（1）嵌入层：**将输入句子中的词语转换为向量表示**----------**使用Pytorch API：torch.nn.Embedding来完成**

****（2）LSTM层：**基于**双向LSTM网络**来建模句子中词语的上下文表示；**

****（3）注意力层：**使用**注意力机制**来从LSTM层的输出中筛选和聚合有效的特征，注意力计算中主要含有**注意力打分函数、注意力分布计算和加权平均**三个模块。**

|  |
| --- |
| ****注意力层实现步骤：**** |
| * ****注意力打分函数：**实现加性打分函数和点积打分函数两种。**   加性模型：假设输入序列为,其中B为批量大小，L为序列长度，D为特征维度，引入一个任务相关的查询向量（可学习参数），加性模型的公式如公式（1）所示。  **（1）**  点积模型：假设输入序列为,其中B为批量大小，L为序列长度，D为特征维度，引入一个任务相关的查询向量（可学习参数），点积模型的公式如公式（2）所示。  **q （2）**   * ****注意力分布计算：****在计算注意力分布的公式中，需要用到Softmax函数计算注意力分布。用表示一组样本的注意力打分值，其中B是批量大小，是填充补齐后的序列长度，每一行表示一个样本中每个元素的注意力打分值，注意力分布的计算如公式（3）所示。   **（3）**  其中softmax(⋅)是按行进行归一化，是掩码（mask）矩阵，比如[PAD]位置的元素值置为-1e9，其它位置的元素值置为0，A是归一化后的注意力分布，也称为注意力权重。（  [PAD]部分在填充-1e9之后，对应的Softmax输出变成了0，相当于把[PAD]这些没有特殊意义字符给屏蔽了，然后剩下元素计算注意力分布，这样做就减少了这些没有特殊意义单元对于注意力计算的影响。）   * ****加权平均：****加权平均就是在使用打分函数计算注意力分布后，用该分布的每个值跟相应的输入的向量相乘得到的结果，计算如公式（4）所示。   **（4）** |

具体代码流程如图5所示。

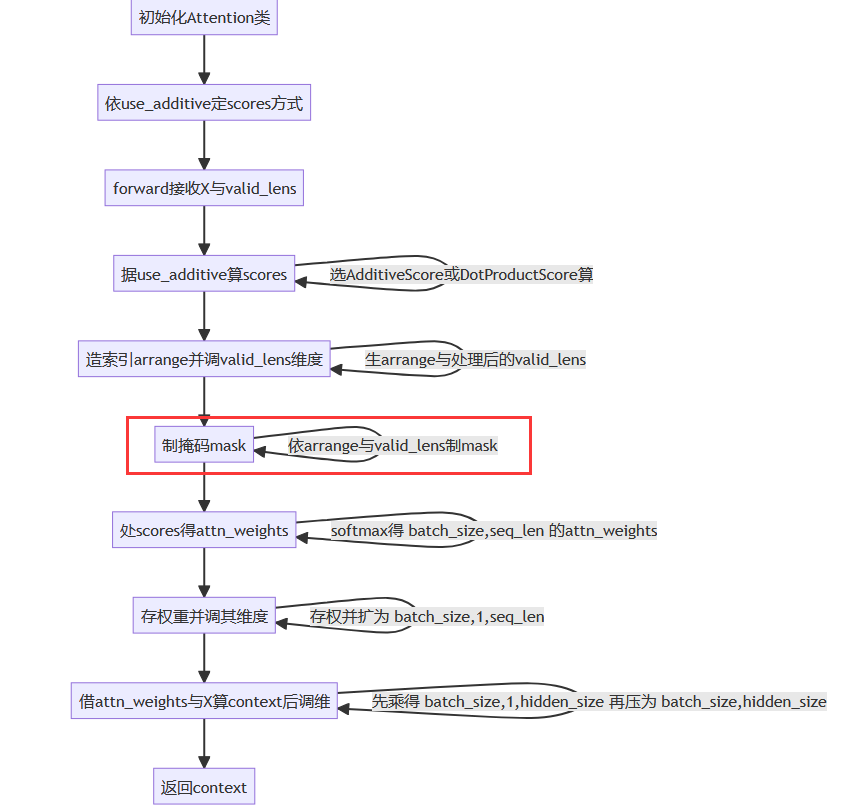


图 5 Attention层实现流程

****（4）线性层：**预测对应的类别得分。**----------**调用torch.nn.Linear来完成。**

1. **模型训练**

**基于RunnerV3来进行模型训练、评价和预测。 使用交叉熵损失函数，并用Adam作为优化器来训练，分别使用加性注意力模型和点积注意力模型。适当调参，并保存准确率最高的模型作为最佳模型。**

1. **模型评价**

**模型评价加载最好的模型，分别使用加性注意力模型和点积注意力模型，然后在测试集合上进行评价。**

**5 .注意力可视化**

**把**加性注意力**的权重提取出来，观察不同输入对应的权重，然后进行可视化分析。**

点击并拖拽以移动​

1. **实验数据及结果分析：**

[【实验16】基于双向LSTM模型完成文本分类任务-CSDN博客](https://blog.csdn.net/qq_73704268/article/details/144404876?sharetype=blogdetail&sharerId=144404876&sharerefer=PC&sharesource=qq_73704268&spm=1011.2480.3001.8118)

[【实验】基于双向LSTM和注意力机制的文本分类-CSDN博客](https://blog.csdn.net/qq_73704268/article/details/144447370?sharetype=blogdetail&sharerId=144447370&sharerefer=PC&sharesource=qq_73704268&spm=1011.2480.3001.8118)

1. **实验结论：**
2. 在不加注意力机制的情况下，测试集上的准确率为0.84704，加入了加性注意力后，测试集的准确率为0.85592；相比于不加注意力机制的模型，加入注意力机制的模型效果会更好些。
3. 加性注意力机制比点积注意力机制在实验中效果更好。
4. **总结及心得体会：**

* **基于双向LSTM模型完成文本分类任务**

1. 在这次实验中又了解到一个新的库函数为from functools import partial，当我们需要经常调用某个函数时，但是其中某些参数是已知的固定值，这个时候就可以考虑使用partial函数，而不用每次都重复传入那些固定的参数值。感觉有点抽象，**举个例子:**

如果有函数def add(x, y): return x + y，使用partial(add, x=3)就会创建一个新函数，这个新函数调用时只需要传入y的值，相当于预先给add函数的x参数固定为了3。

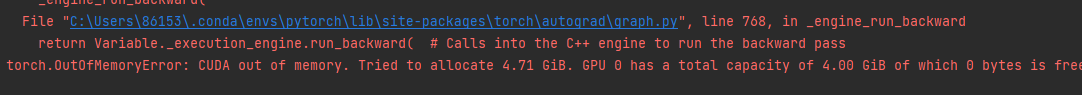
在本次实验中的collate\_fn函数，就将其中的pad\_val参数绑定为word 2id\_dict ["[PAD]"]这个填充值，将max\_seq\_len参数绑定为max\_seq\_len变量所代表的最大序列长度值。之后的代码流程中，调用这个新的collate\_fn时，就不需要再重复传入这两个已经固定好的参数值了，只需完成其他必要的参数传入操作。

1. 实验中我印象最为深刻的操作是**掩码的实现**，具体的实现步骤我在汇聚层流程图中也详细写出，这里再简单总结一下：为了方便以批量的形式将数据输入模型，需要把不同长度的序列统一成相同长度。实验采用的做法是对较短的序列进行[PAD]填充，将其补到和该批次中最长序列一样长。**解耦**在这里就是让汇聚层区分出从输出的包含 [PAD] 位置的向量，防止因为这些无意义的填充位置输出而影响汇聚效果。所有就有了**掩码**操作，采用一种标记机制，对应 [PAD] 位置的元素和原始有效数据位置的元素进行**不同的标记**来明确区分[PAD] 位置和真实数据位置（实验用的true和false）。在汇聚层进行具体运算时，可以依据这个掩码忽略掉 [PAD] 位置的数据。
2. 实验还是比较简单的，嵌入层和BiLSTM都是调用的API，主要是数据的处理和汇聚层的实现，IMDB数据集直接在官网下载的，划分过程和paddle是不一样的，需要注意的一点就是在单词映射的哈希表中添加[PAD]和[UNK]cv，原始的映射中是没有的！！

4、另外再说我实验中遇到的一个很头疼的问题，先说挂Gpu上运行这一步，一直报错，显示模型和数据不在同一设备，最后暴力解决添加了全局配置torch.set \_default\_tens or\_type('torch.cuda.FloatTensor') ，只要是tensor类型的我就给转移到gpu上了，然而并不是我想的那么顺利，这次实验的数据集相对较大，如下：



导致我挂GPU上一直**爆内存**，如下：



无奈之下只能去用服务器远程连接，因为之前zc学长培训教过，很顺利，数据一会就跑完了，比自己的GPU好使，但是**在远程服务器上不能做图形可视化**，只能跑数据，所以我就改了代码，记录每一步的loss，将其保存到了loss.txt文件，最后利用chatgpt做了一下loss的可视化。

* **基于双向LSTM和attention模型完成文本分类任务**

1. 这个实验同样还是掩码操作让我觉得很有趣，和上个实验使用false标记不同，这次是把[PAD]部分填充为**-1e9**之后，这样对应的Softmax输出变成了0，相当于把[PAD]这些没有特殊意义字符给屏蔽了，然后剩下元素计算注意力分布，这样做就减少了这些没有特殊意义单元对于注意力计算的影响。这里就和transformer模型中的掩码实现方式一样了。
2. 有了之前的数据集和代码基础，很顺利完成了，主要是注意力层的实现，不过结果有点奇怪，paddle文档说点积注意力函数效果更好(attention论文中也是点积注意力)，但是我的实验结果明显加性注意力模型的效果更好（不知道是不是数据集的问题），其中我还调了参，由于模型比较复杂，epoch在5以上就已经开始过拟合了，epoch=1也能取得不错的效果。还有一点需要注意就是**paddle中的代码没有打乱训练集**，写代码的时候要打乱一下。
3. 最近很多地方都看到了掩码，在看VIT模型的论文里了解一点点BERT，也使用了一种掩码机制，但是还没仔细研究过这个模型具体是怎么样的。我感觉试着去读论文真的能学到很多奇奇怪怪的知识哈哈哈哈，还挺有意思的。

|  |
| --- |
| **双向LSTM参考资料：** |
| [NNDL 实验6（下） - HBU\_DAVID - 博客园](https://www.cnblogs.com/hbuwyg/p/16617689.html) |
| [情感分析](https://ai.stanford.edu/~amaas/data/sentiment/)--数据集来源 |
| [IMDB数据集的解释\_imdb数据集介绍-CSDN博客](https://blog.csdn.net/qq_23869697/article/details/86505343) |
| [一幅图真正理解LSTM、BiLSTM\_bilstm和lstm的区别-CSDN博客](https://blog.csdn.net/weixin_42118657/article/details/120022112) 写的超细！！ |
| [长短期记忆神经网络（LSTM）介绍以及简单应用分析 - 舞动的心 - 博客园](https://www.cnblogs.com/liuzhen1995/p/11625684.html) |
| [【python函数】partial函数用法解析\_torch partial-CSDN博客](https://blog.csdn.net/qq_43426908/article/details/118530231) |
| [【掩码】深度学习时为什么需要掩码（Mask）？](https://blog.csdn.net/weixin_51636139/article/details/143382822?ops_request_misc=%7B%22request%5Fid%22%3A%224bb1c0a9ae065ad9759e1565a6fab2d3%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=4bb1c0a9ae065ad9759e1565a6fab2d3&biz_id=0&utm_medium=dist) |
| **双向LSTM+attention实验参考资料：** |
| [aistudio.baidu.com/projectdetail/8666087](https://aistudio.baidu.com/projectdetail/8666087) --->源项目地址 |
| [【深度学习】(1) CNN中的注意力机制（SE、ECA、CBAM）](https://blog.csdn.net/dgvv4/article/details/125112972?ops_request_misc=%7B%22request%5Fid%22%3A%229c420694f1b5f6dccec1492febadd4f0%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=9c420694f1b5f6dccec1492febadd4f0&biz_id=0&utm_medium=distribute.pc_) |
| [通俗易懂理解注意力机制(Attention Mechanism)-CSDN博客](https://blog.csdn.net/m0_37605642/article/details/135958384) --->点赞点赞，学到了很多 |
| [注意力机制的两种常用形式-----加性注意力（Additive Attention）和点积注意力（Dot-product Attention](https://blog.csdn.net/m0_51200050/article/details/140034769?ops_request_misc=%7B%22request%5Fid%22%3A%228687b2e1bd081d2d21aefbbebf4c5b9a%22%2C%22scm%22%3A%2220140713.130102334.pc%5Fall.%22%7D&request_id=8687b2e1bd081d2d21aefbbebf4c5b9a&biz_id=0&utm_medium=) |
| [关于Attention的超详细讲解\_attention详解-CSDN博客](https://blog.csdn.net/qq_42363032/article/details/124651978?ops_request_misc=%7B%22request%5Fid%22%3A%22462e68a22b711adb8572f98c0e1898b4%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=462e68a22b711adb8572f98c0e1898b4&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-124651978-null-null.142^v100^pc_search_result_base4&utm_term=attention&spm=1018.2226.3001.4187) |