**中文文本分类**

**摘要**

如今是一个信息爆炸的时代，数据瞬息万变，人们一方面每天通过电子设备或者纸质媒介被动接收海量信息，另一方面要主动的搜寻真正满足自身所需的信息。文本作为信息的重要载体在数据信息中具有有无可代替的地位，对应于文本的分类技术也就应运而生。其作为自然语言处理的一个重要组成，通过有监督的学习，将学习到的规律应用于新的文本，将其映射到某一类别的主题上，在情感分析、话题标记、新闻分类等场景中得到广泛应用。

本文将基于中文新闻文本数据，使用深度学习的文本分类流程：对文本进行预处理、建立模型、对模型进行评估、最后实现对分本的类别进行预测。

**关键词**：Textcnn，中文文本分类，自然语言处理

1. **问题重述与分析**
   1. **问题重述**

已有一个中文新闻数据集，给出数据标签和对应的中文标签，训练集共10000条，测试集共1999条，共10个分类标签。要求掌握自然语言处理中文本分类的基本流程，自行搭建深度学习模型，对所给的文本数据集进行文本分类。

* 1. **问题分析**

想要实现对文本的分类，文本中的字词就是文本分类的依据。与英文垃圾邮件分类区别：

1. 英文句子本身就可以以空格为分割符，方便将句子切分获取词汇。而中文文本无空格来划分语义单元，因此使用了jieba分词对中文文本进行分词。
2. 基于简单朴素贝叶斯的垃圾邮件分类，统计的是在两个类别邮件下词汇是否出现，出现标记1，没出现是0，用一段无序的单词序列来表达一段文本。但是这种统计方式各个词汇都是独立的，看不出两个词汇之间是否有关联，语义上是否相似，即忽略了文本的语法和语序；且如果字典词汇数目太大，一段文本用大部分都是的0稀疏向量表示，维度太大计算困难。因此需要使用其他方法，word2vec是把词汇用一个向量表示，我们可以给向量定一个维度，每个维度是其一个特征，机器会学习出来词汇的许多不同的特征，尽管我们不知道具体的含义。词嵌入是把一段文本的每个语义单元都有特征向量表示，组合成一个文本长度\*特征向量维度的矩阵。

文本预处理之后需要构建模型，本次实现学习并搭建了textCNN模型。Embedding层生成文本对应的特征向量矩阵；Convolution层进行特征的提取，侦测一段文本中的某个特征；Max pooling保留关键的信息；Flatten把得到的不同特征拉直后交给全连接层，经过非线性变换的激活函数进行分类，最后使用softmax激活函数得到对应可能类别的概率。

模型训练好之后，对测试集进行文本预处理，进行预测，最后通过argmax得到对应的标签，分析损失值和准确率，改变模型的超参数进行优化。

1. **文本分类流程**

Text

Preprocess

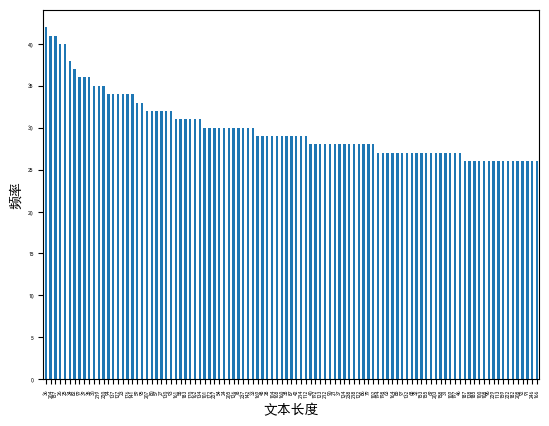
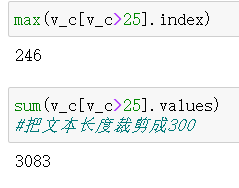
Model-train

Evaluation

Predict

**2.1文本预处理**

1. 句子切分及过滤。先读取数据，利用jieba分词将句子分词分为以空格为分隔符的一个个最小语义长度的词语或字的列表，并提前准备好停用词表，过滤掉停用词。
2. 划分数据集在分词结束后进行，这样字典里面不会有训练集的词，评估的时候更加有代表性。
3. 使用keras的文本预处理模块。首先实例化一个Tokenizer类，构建一个分词器；然后使用.fit\_on\_texts (texts)，作用是对词语进行编号和统计将我们的文本序列；最后使用.text\_to\_word\_sequence将训练集和验证集的文本序列变成数字序列。由于输入模型中的文本长度需要一致，因此还需要使用keras的序列预处理模块中的pad\_sequences把句子裁剪成固定的长度。标签数据要用独热编码表示。

长度的选择是先统计所有文本的长度，发现长度小于或等于246的文本出现每条出现的次数大于25次，占样本数10000约1/3，因此把大部分文本的大约长度作为句子裁剪的长度。

**2.2 模型构建**

使用text\_cnn，将卷积神经网络应用到文本分类，利用多个不同大小的卷积核来提取句子中的关键信息，能够捕获局部相关性。

Embedding

Convolution 1

Max pooling

concatenate

Convolution 2

Convolution 3

Max pooling

Max pooling

flatten

Fully connected network

Inputs

Softmax

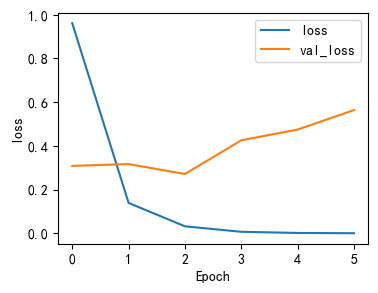
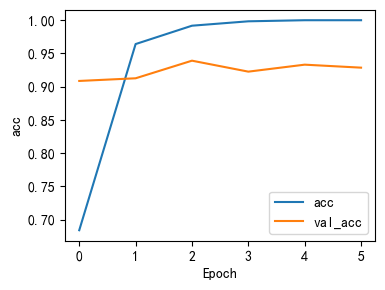
Outputs

编译模型：指定损失函数为交叉熵损失，优化方法和监控指标.

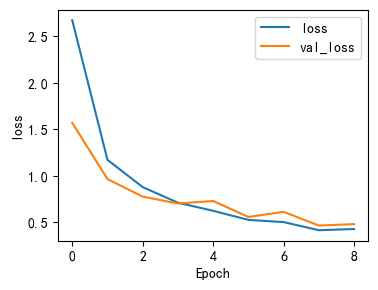
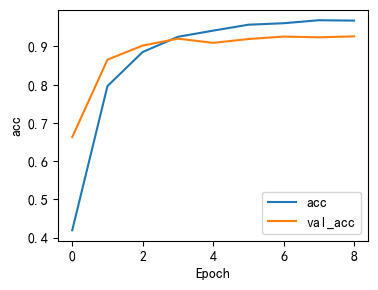
**2.3超参数实验**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 参数  实验 | Filter | Batch\_  Size | Embed\_  Dim | Max\_len | Hidden\_  dim | Pool\_  Size | Window\_  Size | Acc |
|  | 32 | 32 | 32 | 300 | 24 | 2 | [3,4,5] | 0.944 |
| 一 | 64  128 | 32  32 | 32  32 | 300  300 | 64  64 | 2  2 | [3,4,5]  [3,4,5] | 0.946  0.943 |
| 二 | 32  32 | 64  128 | 32  32 | 300  300 | 128  128 | 2  2 | [3,4,5]  [3,4,5] | 0.944  0.958 |
| 三 | 32  32 | 64  64 | 64 128 | 300  300 | 64  64 | 2  2 | [3,4,5]  [3,4,5] | 0.953  0.954 |
| 四 | 32 | 64 | 128 | 200 | 64 | 2 | [3,4,5] | 0.958 |
| 五 | 32 | 64 | 128 | 200 | 24 | 4  Global | [3,4,5] | 0.944  0.952 |
| 六 | 32 | 64 | 128 | 200 | 24 | 2 | [4,5,6] | 0.953 |
|  | 32 | 64 | 128 | 200 | 24 | 2 | [2,3,5] | 0.945 |

结论：



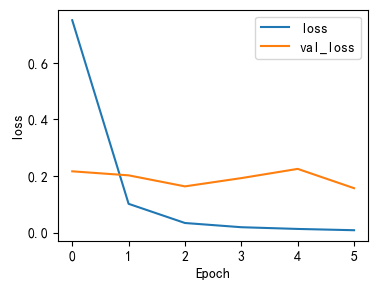
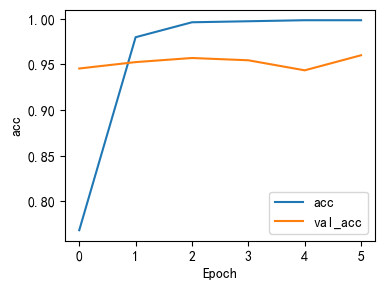
实验：给dense层加上了L2正则化，达到收敛需要的epoch次数增加了，适度的正则化和添加dropout层有利于减少过拟合问题；



实验一：Filter个数：最终特征的维度，filter越多，深度越大，实验结果说明适当的filter个数有利于提高准确率。本次实验filter增加，准确度变化较小，32个filter也能取得不错的结果。全连接层隐藏层维度，将特征值整合成一个值。

实验二：batch\_size大，训练时间会变长，不适合大样本训练，梯度比较准确；若batch\_size小，梯度变化波动大，不容易收敛。

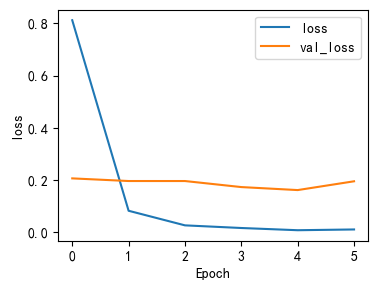
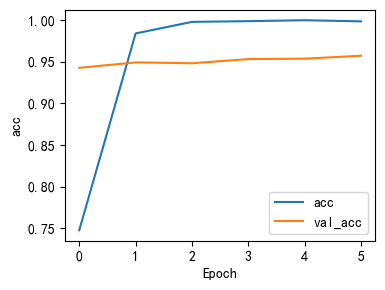
实验三：把embedding的维度在第一次的基础上增大，准确度稍提高；再增大，准确度变化不大。



实验四：合理的pad\_size有利于保存更多有利信息，过大会填充大量无用信息，过小会丢失重要信息。

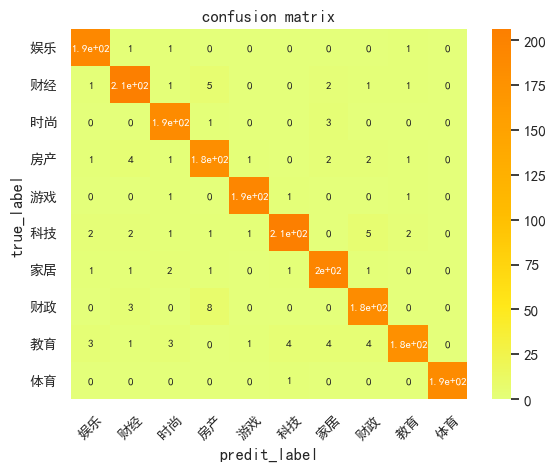
实验五：池化层的滑窗尝试过2，4，还有全局池化，实验结构是相差不大但是常用2。

实验六：Filter尺寸定了抽取n\_gram特征的长度，也设置了三种不同长度组合，长度大虽然能提取信息多，但是参数也会增多。



**2.4模型评估**

评估指标：交叉验证数据上的交叉熵损失作为衡量模型性能的主要指标。



这是一个混淆矩阵热力图，从图中可以直观看出模型在哪一类样本里面表现的不是很好，以及对角线上能看到绝大多数为正确的预测，其中体育类别预测的较好，科技，教育类的分类错误比较多。

1. **结果分析和总结**

**3.1结果分析**

尽管调整参数后再验证集上的准确度变化不大，说明了想要提升准确率还需要进一步的思考。查看了几个分类错误的文本，比如科技类的文本可能涉及到很多其他类别的词汇；又或者是截断文本的时候把关键词截掉了。了解到Textcnn优缺点如下：优点：模型简单，训练速度较快，解决了词袋模型的稀疏性问题；缺点：对中短文本场景好于长文本；maxpooling可能会丢掉一些有用特征并丢掉语序。

**3.2总结**

TextCNN在此次的新闻文本分类中有不错得表现，在keras的帮忙下也能快速实现这个模型，成功运行和预测出结果还是很令人激动的。但对于中文文本分类还有其他模型还没有尝试，这段时间只侧重学习了textcnn，没能对比不同模型的预测结果与性能有点遗憾，不过未来还可以继续学习。