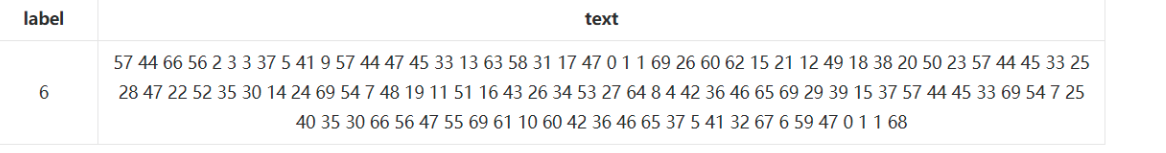
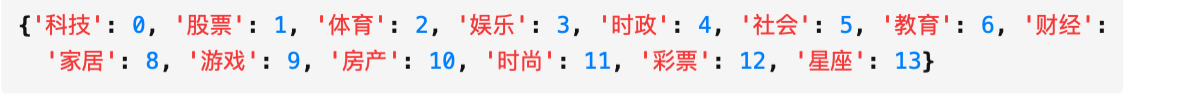
**基于TextRNN的新闻文本分类**

**一、研究内容和数据分析**

本实验主要来自天池的一个新人赛，赛题以自然语言处理为背景，要求选手根据新闻文本字符对新闻的类别进行分类，这是一个经典文本分类问题。通过这道赛题可以引导大家走入自然语言处理的世界，带大家接触NLP的预处理、模型构建和模型训练等知识点。

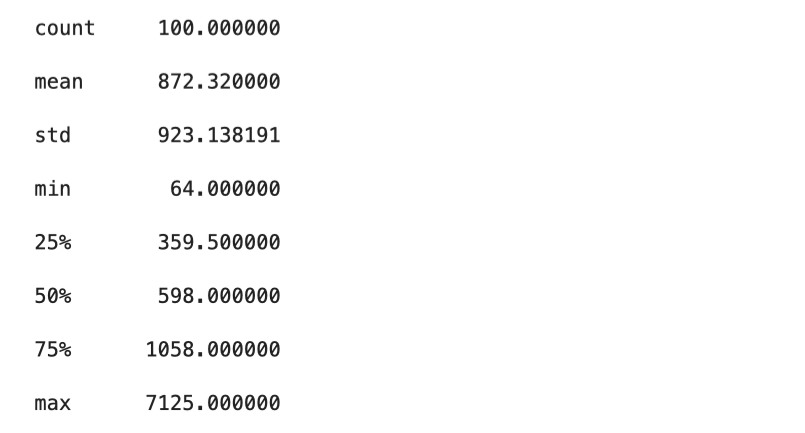
赛题以匿名处理后的新闻数据为赛题数据，数据集报名后可见并可下载。赛题数据为新闻文 本，并按照字符级别进行匿名处理。整合划分出14个候选分类类别：***财经、彩票、房产、股票、家居、教育、科技、社会、时尚、时政、体育、星座、游戏、娱乐***的文本数据。

题数据由以下几个部分构成：训练集20w条样本，测试集A包括5w条样本，测试集B包括5w条样本。为了预防选手人工标注测试集的情况，我们将比赛数据的文本按照字符级别进行了匿名处理。处理后的赛题训练数据如下：

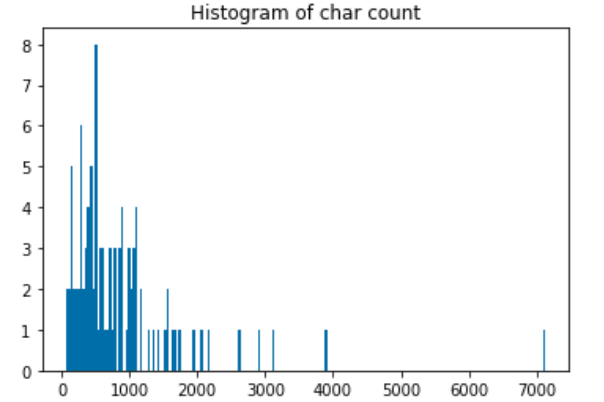
在数据集中标签的对应的关系如下：测指标如下：评价标准为类别f1\_score的均值，选手提交结果与实际测试集的类别进行对比，结果越大越好。

实验数据分析：在读取完成数据集后，我们还可以对数据集进行数据分析的操作。虽然对于非结构数据并不需要做很多的数据分析，但通过数据分析还是可以找出一些规律的。此步骤我们读取了所有的训练集数据，在此我们通过数据分析希望得出以下结论：赛题数据中，新闻文本的长度;赛题数据的类别分布是怎么样的，哪些类别比较多;赛题数据中，字符分布是怎么样的。

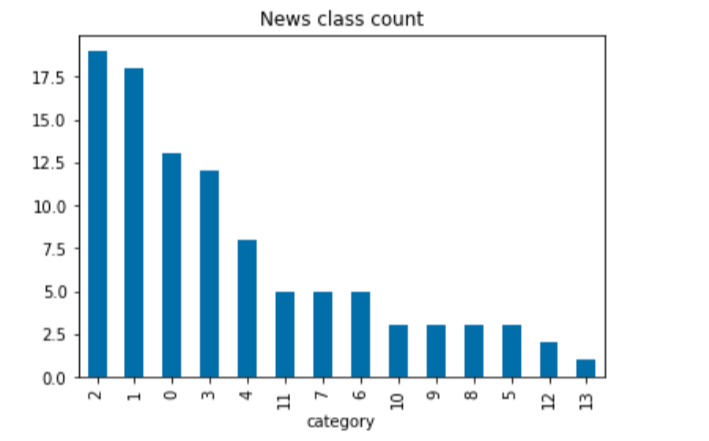
在赛题数据中每行句子的字符使用空格进行隔开，所以可以直接统计单词的个数来得到每个句子的长度。统计并如下



对新闻句子的统计可以得出，本次赛题给定的文本比较长，每个句子平均由907个字符构成，最短的句子长度为2，最长的句子长度为57921。下图将句子长度绘制了直方图，可见大部分句子的长度都几种在2000以内。



接下来可以对数据集的类别进行分布统计，具体统计每类新闻的样本个数。



从统计结果可以看出，赛题的数据集类别分布存在较为不均匀的情况。在训练集中科技类新闻最多，其次是股票类新闻，最少的新闻是星座新闻。接下来可以统计每个字符出现的次数，首先可以将训练集中所有的句子进行拼接进而划分为字符，并统计每个字符的个数。从统计结果中可以看出，在训练集中总共包括6869个字，其中编号3750的字出现的次数最多，编号3133的字出现的次数最少。通过上述分析我们可以得出以下结论：赛题中每个新闻包含的字符个数平均为1000个，还有一些新闻字符较长；赛题中新闻类别分布不均匀，科技类新闻样本量接近4w，星座类新闻样本量不到1k；赛题总共包括7000-8000个字符；通过数据分析，我们还可以得出以下结论：每个新闻平均字符个数较多，可能需要截断；由于类别不均衡，会严重影响模型的精度；

**二、实验方法和流程**

1. **文本分词**

词是最小的能够独立活动的有意义的语言成分，英文单词之间是以空格作为自然分界符的，而汉语是以字为基本的书写单位，词语之间没有明显的区分标记，在常规的中文文本分类任务中，开始一般需要对数据进行中文分词和去除停用词的操作，但是由于本次数据是匿名处理的，所以用字作为最小的处理单元，无需分词操作。

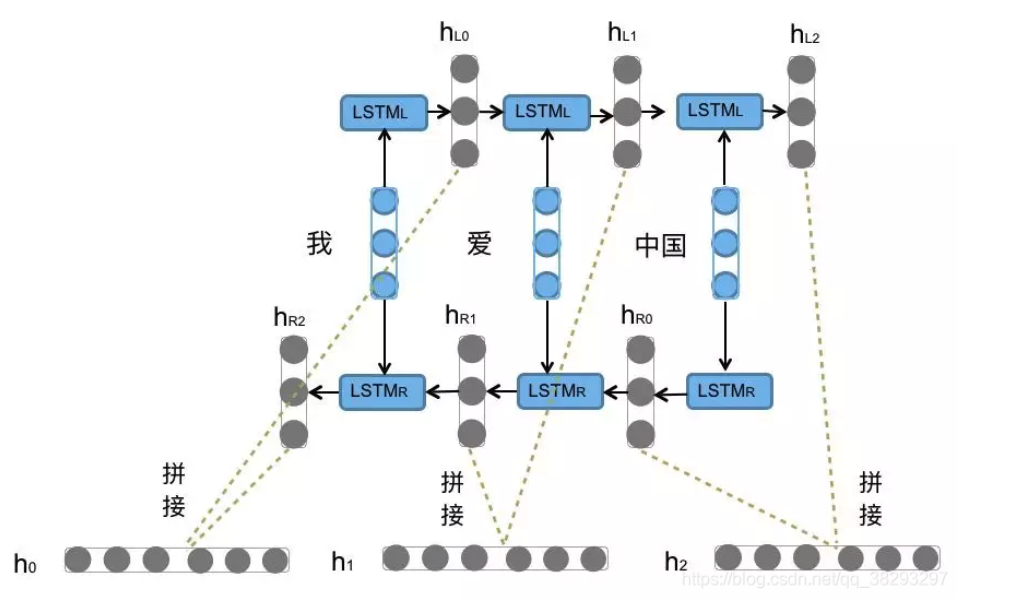
1. **word2vec**

word2vec，即词向量，将一个词用一个向量来表示。是2013年Google提出的。word2vec工具主要包含两个模型：跳字模型（skip-gram）和连续词袋模型（continuous bag of words，简称CBOW），以及两种高效训练的方法：负采样（negative sampling）和层序softmax（hierarchical softmax）。word2vec词向量可以较好地表达不同词之间的相似和类比关系。word2vec是一个NLP工具，它可以将所有的词向量化，这样词与词之间就可以定量的去度量他们之间的关系，挖掘词之间的联系。

在本次实验中，由于训练集数据量比较大，所以为了提高模型的性能，用训练集的数据预训练了一个维度为100的Word2vec词向量

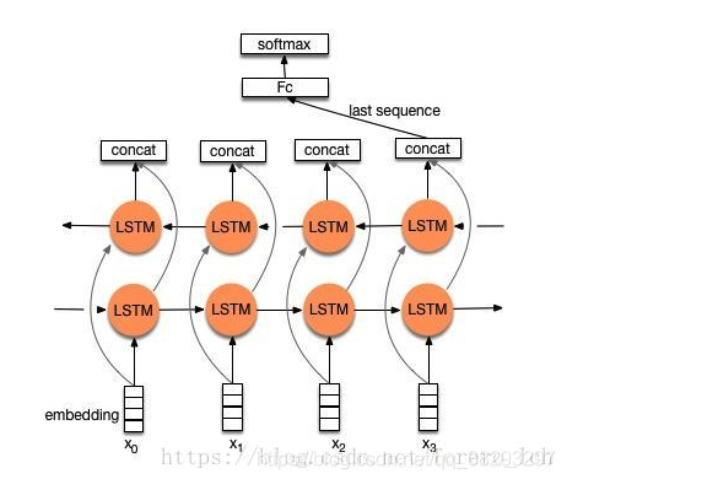
1. **Bi-LSTM**

LSTM的全称是Long Short-Term Memory，它是RNN（Recurrent Neural Network）的一种。LSTM由于其设计的特点，非常适合用于对时序数据的建模，如文本数据。Bi-LSTM是Bi-directional Long Short-Term Memory的缩写，是由前向LSTM与后向LSTM组合而成。



1. **TextRNN**

TextRNN的结构分为：1. embeddding layer, 2.Bi-LSTM layer, 3.concat output, 4.FC layer, 5.softmax,下图是一个传统的TextRNN模型结构图：



一般取前向/反向LSTM在最后一个时间步长上隐藏状态，然后进行拼接，在经过一个softmax层(输出层使用softmax激活函数)进行一个多分类；或者取前向/反向LSTM在每一个时间步长上的隐藏状态，对每一个时间步长上的两个隐藏状态进行拼接，然后对所有时间步长上拼接后的隐藏状态取均值，再经过一个softmax层(输出层使用softmax激活函数)进行一个多分类(2分类的话使用sigmoid激活函数)。但是在本次实验中使用的是对LSTM的每个时间步的输出进行Attention之后的结果作为句子或者文本的向量表示，再进行后续的处理。

**5 实验流程**

**5.1 数据处理**

数据首先会经过all\_data2fold函数，这个函数的作用是把原始的 DataFrame 数据，转换为一个list，有10个元素，表示交叉验证里的10份，每个元素是dict，每个dict包括label和text。首先根据label来划分数据行index,生成label2id。label2id是一个dict，key为label，value是一个list，存储的是该类对应的index。然后根据label2id，把每一类别的数据，划分到 10 份数据中. 最终得到的数据fold\_data是一个list，有 10 个元素，每个元素是dict，[{labels:textx},{labels:textx}. . .]。最后，把前 9 份数据作为训练集train\_data，最后一份数据作为验证集dev\_data，并读取测试集test\_data

**5.2 词表Vocab**

创建 词 和 index 对应的字典，这里包括 2 份字典，分别是：\_id2word 和 \_id2extword。其中 \_id2word 是从新闻得到的， 把词频小于 5 的词替换为了 UNK。对应到模型输入的 batch\_inputs1。\_id2extword 是从 word2vec.txt 中得到的，有 5976 个词。对应到模型输入的 batch\_inputs2。后面会有两个 embedding 层，其中 \_id2word 对应的 embedding 是可学习的，\_id2extword 对应的 embedding 是从文件中加载的，是固定的。创建 label 和 index 对应的字典。上面这些字典，都是基于train\_data创建的。

**5.3 模型**

5.3.1 数据准备：

上上一步得到的 3 个数据，都是一个list，list里的每个元素是 dict，每个 dict 包括 label 和 text。这 3 个数据会经过 get\_examples函数。 get\_examples函数里，会调用sentence\_split函数，把每一篇文章分割成为句子。  
然后，根据vocab，把 word 转换为对应的索引，这里使用了 2 个字典，转换为 2 份索引，分别是：word\_ids和extword\_ids。最后返回的数据是一个 list，每个元素是一个 tuple: (label, 句子数量，doc)。其中doc又是一个 list，每个 元素是一个 tuple: (句子长度，word\_ids, extword\_ids)，在迭代训练时，调用data\_iter函数，生成每一批的batch\_data。在data\_iter函数里，会调用batch\_slice函数生成每一个batch。拿到batch\_data后，每个数据的格式仍然是上图中所示的格式，下面，调用batch2tensor函数，batch2tensor函数最后返回的数据是：(batch\_inputs1, batch\_inputs2, batch\_masks), batch\_labels。形状都是(batch\_size, doc\_len, sent\_len)。doc\_len表示每篇新闻有几句话，sent\_len表示每句话有多少个单词。

batch\_masks在有单词的位置，值为 1，其他地方为 0，用于后面计算 Attention，把那些没有单词的位置的 attention 改为 0。

batch\_inputs1, batch\_inputs2, batch\_masks，形状是(batch\_size, doc\_len, sent\_len)，转换为(batch\_size \* doc\_len, sent\_len)。

5.3.2 模型网络：

Embedding层：

batch\_inputs1, batch\_inputs2都输入到WordLSTMEncoder。WordLSTMEncoder包括两个embedding层，分别对应batch\_inputs1，embedding 层是可学习的，得到word\_embed；batch\_inputs2，读取的是外部训练好的词向量，因此是不可学习的，得到extword\_embed。所以会分别得到两个词向量，将 2 个词向量相加，得到最终的词向量batch\_embed，形状是(batch\_size \* doc\_len, sent\_len, 100).

WordLSTMEncoder层：

使用一个双层的LSTM对于单词进行编码，这里采取的是BiLSTM + Attention 对句子进行编码。

SentEncoder层

用2层的BiLSTM对WordLSTMEncoder + Attention层编码后的句子向量进行处理，隐藏层大小默认为256，目的是最终得到一个新闻doc的向量表示，但是由于加了attention机制，所以这个网络的输出为bilstm的output而非最后

的hn，常规的做法是把最后一层的bilstm的hn进行拼接得到整个句子的向量表示，但是会造成新闻中部分信息的丢失，预测效果降低，所以这里为了后续的attention层处理方便，返回的是整个output。

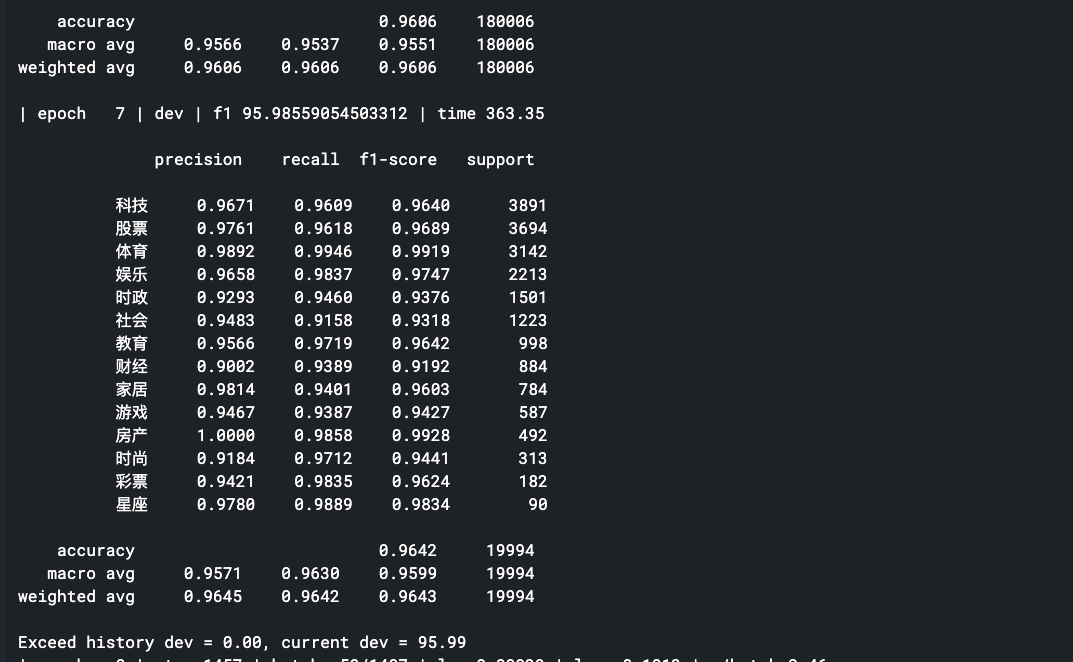
Attention层：

Attention的输入是sent\_hiddens和sent\_masks，首先sent\_hiddens经过线性变换得到key值，维度不变，所以key的shape是：(batch\_size, doc\_len, num\_directions \* hidden\_size)，然后key和query相乘得到最后outputs，query的维度在这里是512，所以outputs的shape为(batch\_size, doc\_len)

FC层：

最后经过FC层，得到分类概率的向量。

6 实验结果



**三 课程感悟**

学习了《机器翻译》这门课，课堂上的学习和实践，使我对自然语言处理有了比较系统的认识，课程的内容比较完善，但是由于课时原因没有能够讲解更多的知识，我研究生阶段的研究课题主要是知识图谱和自然语言处理方向，通过这门课程的学习，我将会逐渐的学习更多有关的知识，不断提高实践能力。