情感分类 实验报告

张翔 2017011568 计76

2019 年 5 月 31 日

目录

1 基本思路

1.1 一些准备工作

读取所有训练用语料，建立Vocabulary（词与index的映射关系）。对于没有在预训练词向量中出现过的词，标记为unknown(<unk>)，对应的词向量采用随机值初始化。对于长度不够的句子，进行补长处理，补充部分标记为<pad>。这里我过滤了语料中频率为1的词，虽然它们比较独特，但在总体的训练集里可能成为噪声，从而影响最后的实验结果。

1.2 训练步骤

使用Word Embedding将输入的文本（此时已经转换成词对应的index序列）转换为词向量序列，然后使用某种神经网络进行处理。对于数据集中的情感标签，这里进行独热编码(One Hot Encoding), 即将最大值编码为1，其余编码为0，从而问题转化为文本情感的多标签分类(Multi-label Classification)问题。网络使用Keras实现，其内置了词向量层，从而可以实现端到端的训练。词向量采用了`https://github.com/Embedding/Chinese-Word-Vectors`下预训练好的搜狗新闻，从而最大程度接近实验所给的语料库。

2 网络结构

2.1 CNN

参考了2014年的Text CNN论文，实现了如下结构的CNN

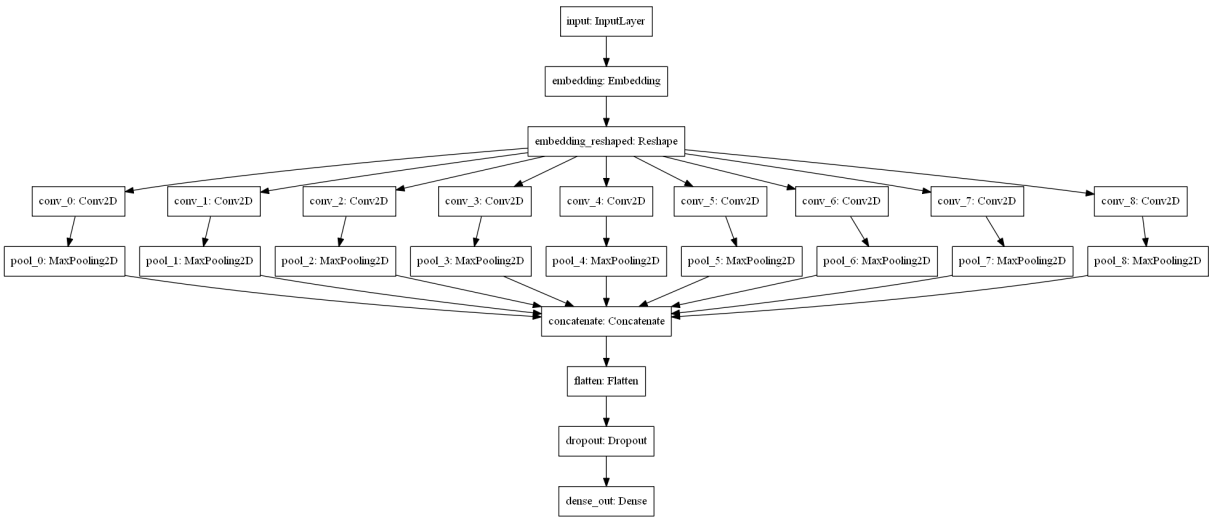


图 1: Regular CNN

输入文本经过Embedding层（使用词向量初始化，参数为预训练好的，在训练过程中不可变，即trainable设置为False）后，会转化成300维的词向量，而每个文本长度被限制在1000词，超出部分截去，不足部分补零，则实际上得到了一个1000×300维的矩阵，然后经过$n \times 300$的卷积核进行卷积，其中$n = 2, 3, 4, \cdots, 10$，每个卷积层有100 filters，输出后经过Max Pooling层。这些Max Pools的输出合并后经过一个Dropout层，然后经过全连接层分类器，最后生成8个类别输出。

另外实现了一个更改过的CNN。与上述网络的区别在于增加了一个可训练的Embedding层，从而得到的1000×300维矩阵不再是单通道，而是双通道。结构如下

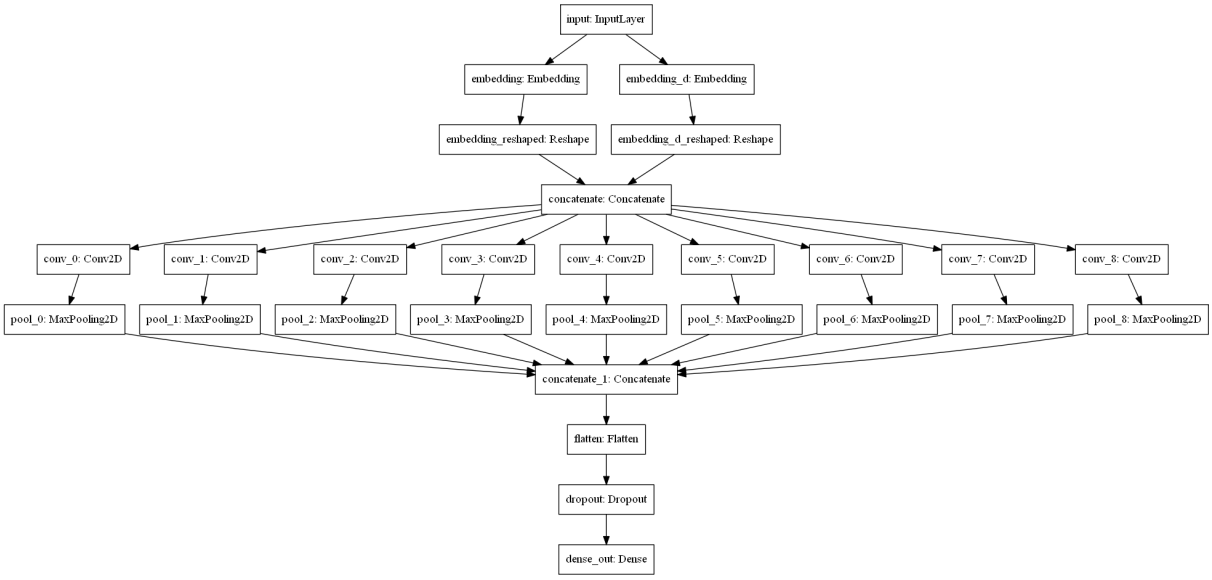


图 2: Multi Channel CNN

关于可训练的Embedding通道，有随机值初始化与使用预训练值初始化两种方式，这里均进行了测试，结果附在第5部分。

2.2 LSTM & Bidirectional LSTM

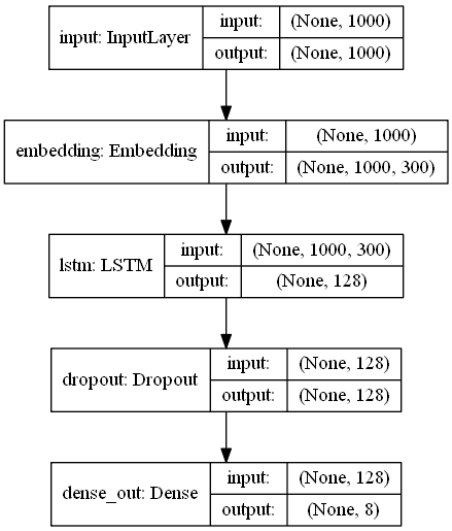


图 3: Basic LSTM

基本的LSTM实现如上图所示。这里得到的词向量经过LSTM Cell处理，然后将它们的State输出到Dropout以及全连接层，用于实现分类功能。

另外还实现了一个双向LSTM，与上述LSTM区别在于，双向LSTM从正向和反向两个方向对输入序列进行处理。这里的实现是将正向和反向得到的State等权值相加，然后给全连接层分类。网络结构如下

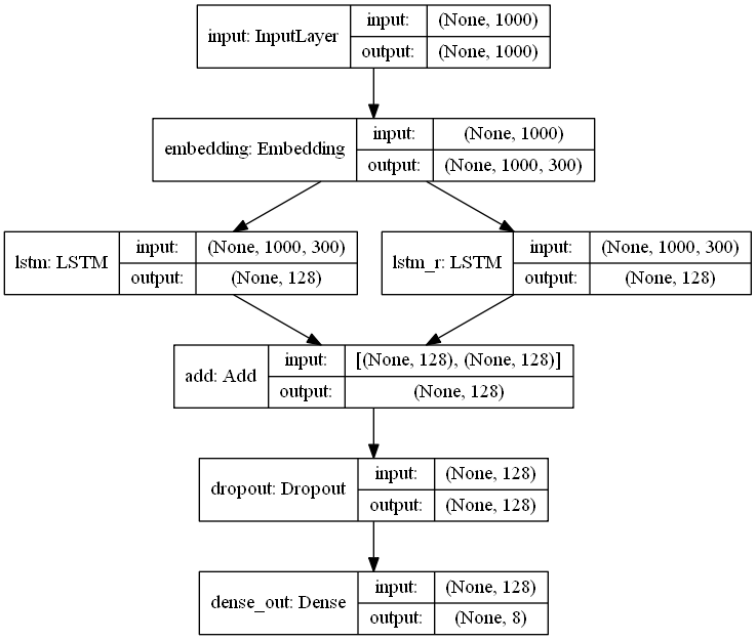


图 4: Bi-directional LSTM

2.3 MLP

这里实现了两种MLP网络作为Baseline。第一种不需要词向量，而是直接将文本中所有词对应的index排成一个1000维的向量（截长补短），作为网络的输出，经过三个全连接隐层后，直接输出分类结果。网络结构如下

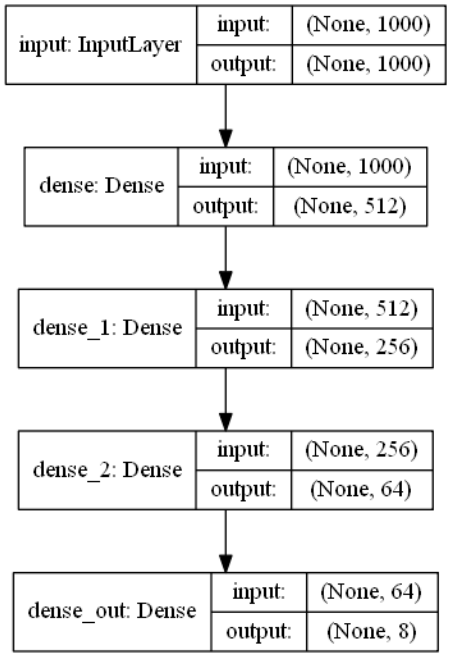


图 5: MLP Simple

第二种将1000个300维的词向量加起来，得到一个300维的向量，再给全连接层分类，结构如下

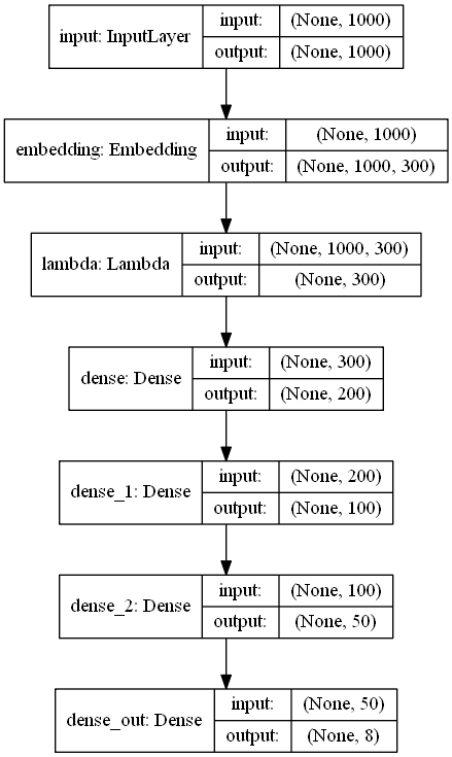


图 6: MLP

3 参数调整

3.1 输入序列的长度

使用CNN时，要求固定的输入序列长度，该值过大过小均不合适，过小使得大量信息丢失，过大则降低性能。统计训练集和测试集的语料长度可以发现，平均在400左右，但最大长度超过1000。经过调整，我发现取输入序列长度为1000时准确度较好，性能也能够接受（Batch Size为256时，大约8s可以完成一个epoch），而输入序列取400时，准确率大约有1-3%的下降。

3.2 CNN卷积核的大小

将词映射到预训练的词向量后，CNN的卷积层能够对其实现特征提取。卷积核的第二个维度与词向量维度保持一致，而第一个维度的大小n类似于n-gram的作用。经过测试，我发现$2-10$的卷积核大小抽取能力较好，而诸如15甚至更大的卷积核，不仅降低了训练速度（单次卷积成本提高），而且使得准确率下降约1%。因此，最后取$n = 2, 3, \cdots, 10$大小的核进行特征提取。

3.3 Batch Size

Batch size是每批次用于训练的数据数量，该值越小，梯度的估值越不准确，波动越大。极端情况Batch size取1，甚至会导致训练难以收敛。Batch size过大时，则需要占用较多内存，从而对硬件提出了更高的要求。这里对于训练参数较少的网络，如Regular CNN和MLP，使用了更大的Batch size(256)；对于训练参数较多的网络，如Multi-Channel CNN，或者训

练比较耗时的LSTM，使用了更小的Batch size(64)。从5实验结果中Regular CNN的对比可以看出，Batch Size较大时，效果要略好一点。

3.4 Dropout Rate

这个参数控制的是神经元的存活概率，一般在0.5 ~ 0.8之间。我在Regular CNN模型上测试了0.5, 0.65, 0.8三组参数，但没有发现明显的差别，最后统一取0.8。

4 问题思考

4.1 实验训练何时停止最合适

实验训练epoch过少会使得梯度下降不充分，从而无法发挥网络的最大性能；epoch过多则可能发生过拟合，也会导致准确率下降。这里采用的方法是将训练集85%用于train，15%用于validate，每个epoch完成后进行一次validate，从而可以实时地知道网络的训练情况。实验中我发现train准确率一般会单调递增，到某个值之后就不再上升，而是在其附近抖动，此后一段时间validate准确率可能会出现下降的情况，而这种情况说明发生了过拟合。因此我会寻找一个train accuracy较高，而validate准确率还未开始下降的epoch作为训练的终止点。这种方法相比固定迭代次数来说更加灵活，但需要实时监控训练的过程，并在恰当的时候采取措施；固定迭代次数策略比较简单，但训练可能在validate准确率下降的过程中终止，从而得到的模型是过拟合的。不同的网络需要的迭代次数不一样，固定策略也并不合适。

如下图，蓝色为Multi-channel CNN，橙色为Regular CNN，可以看出，前者准确率的上升速度远高于后者，大约在10个epoch时，前者出现了过拟合，而后者仍然在训练过程中。由此可以看出，不同网络收敛速度是不同的，应该根据验证集的准确度以及网络本身的性质对训练次数加以调整。

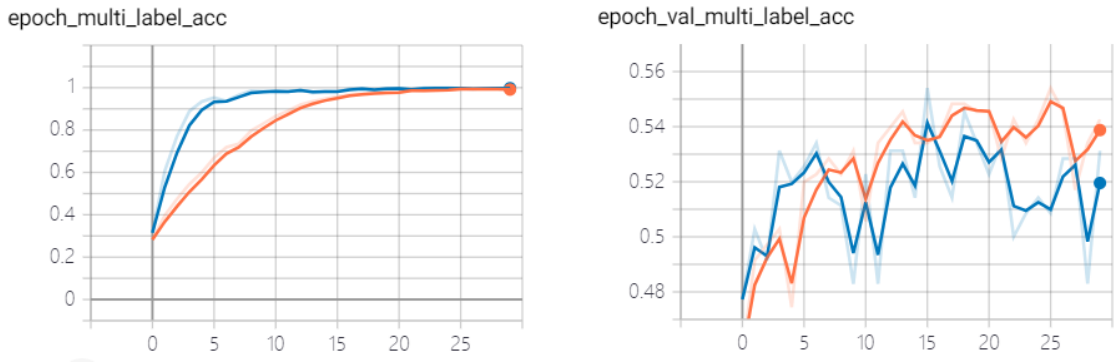


图 7: Multi-channel CNN与Regular CNN训练过程比较

4.2 实验参数如何初始化

实验中，根据经验以及相应参数的特性，对不同参数采取了不同的初始化策略。如卷积层的核函数采用[0, 0.1]的截尾高斯分布初始化，可训练的Embedding层的核函数使用[0, 0.1]的均匀分布初始化，全连接层的核函数则使用Xavier均匀初始化。对于所有的bias，使用零值初始化。

这里没有使用正交初始化。正交初始化主要是用来避免RNN的梯度爆炸或消失的问题。代价对于参数的导数正比于参数矩阵特征值的幂次λ_i^t，如果$|\lambda| > 1$，步数增加时会发生梯度爆炸，反之会发生梯度消失。理想情形下应使得$\lambda = 1$，此时参数矩阵W是单位正交阵。因此使用正交初始化可以在训练刚开始的时候避免梯度出现问题。但是本实验使用的LSTM结构本身可以较好地避免梯度消失/爆炸，从而不需要特别地进行正交初始化。

4.3 如何防止过拟合

常见方法是Early Stopping, Dropout, L2 Regularization, Data Augmentation。本实验中使用了Early Stopping和Dropout，根据Validation Set的准确率判断训练需要的Epoch数目，并对网络结构加入Dropout的操作，尽量避免过拟合的出现。

4.4 CNN, RNN, MLP的比较

MLP网络比较简单，输入可以直接用词袋模型，或者将词向量简单相加，然后使用多层全连接网络进行分类。此种网络训练很快，效果不差（如Baseline MLP准确率能达到和CNN类似的效果），但没有利用上下文信息。

CNN的卷积核类似n-gram模型，能够较好地抽取特征信息，然后使用池化得到影响最大的特征，然后通过全连接网络进行分类，最后实验效果也是最好的，训练速度也还是比较快的。但是，CNN要求输入是固定大小的，从而必须对不同长度输入的句子进行截尾或者补零操作，并且网络的输入严重依赖于语料集句子的平均长度，从而对使用的灵活性略有损害。并且，CNN卷积核的大小的选取也需要一些经验。

RNN诸如LSTM等处理文本序列比较自然，因为其递归地在输入序列上操作，从而考虑了历史信息和词的顺序。但是对于较长的序列，RNN会遗忘序列开头的重要信息，从而会影响分类结果。一种解决方式是采用双向LSTM，从而可以兼顾序列头尾的一些信息，从而获得更好的效果，实验结果也验证了这一想法。另外，从训练速度来说，RNN是这三种模型中最慢的，虽然cuDNN LSTM的实现会快不少，但是它不支持mask zero功能，不方便处理带有补零信息的序列。

4.5 实现时需要注意的一些问题

1. 实现LSTM时，注意Embedding层需要将mask_zero设置为True，否则末尾补充的0将影响LSTM Cell正常运算，将出现不收敛的情况；
2. 实验中过滤频率为1的词是有正向作用的，测试该种措施可以将CNN的准确率提高~1%；
3. CNN的卷积操作实际上是一维卷积（卷积核在词向量的维度方向不滑动），如果使用Conv2D，需要注意对词向量层的输出进行Reshape，使其成为类似图片的结构，即增加第三个表示通道数的维度。

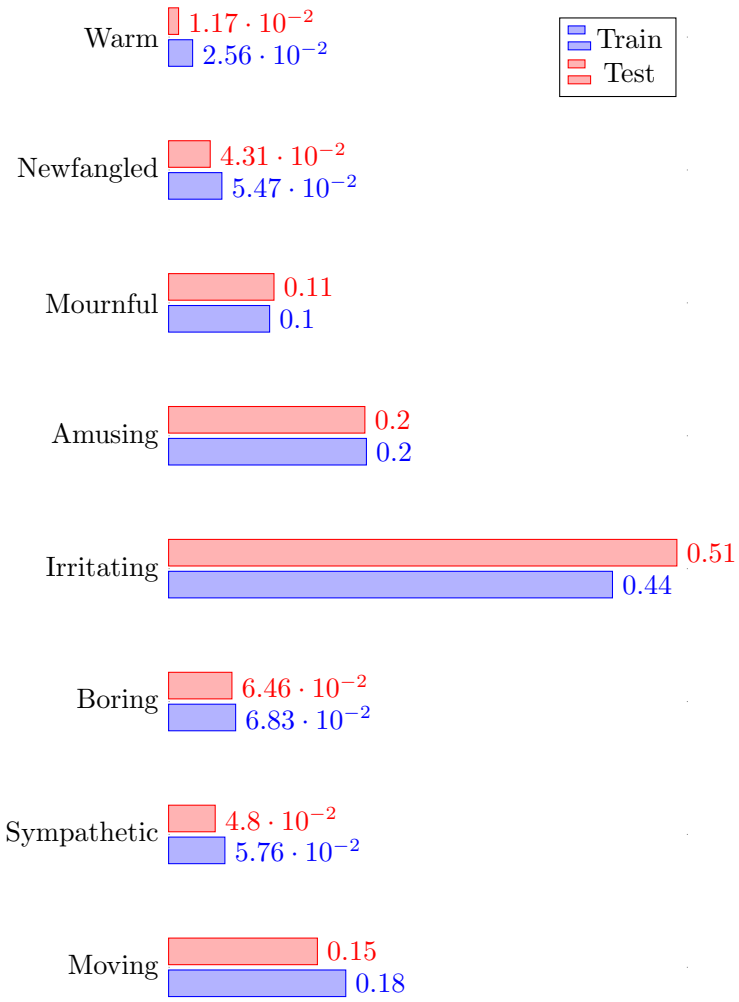
4.6 F1-Score较低的原因

实验中，我发现CNN，LSTM等模型虽然准确率能达到50-60%，但Macro F1-Score较低，只有20%-30%，而Micro F1-Score能达到50%以上，其中一个重要原因是模型预测时，

出现了某些类完全预测不到的情况，而sklearn.metrics.f1_score函数会将这种类别的F1-Score算作0，从而拉低整体的F1-Score。

对数据集（训练和测试集）进行统计，可以发现情感分布如下图所示

图 8: Sentiment Distribution



容易发现，数据集的情感分布偏差较大(High bias)，约一半的情感均为“愤怒”，而“温暖”“新奇”“同情”“无聊”则非常少，并且测试集相较于训练集更加重了这种不均匀分布。最后模型预测时，对于“温暖”这一类情感，几乎检测不出，因为检测结果为所有情感概率值argmax后得到的分类，而这种数据集中出现较少的情感最后预测出的概率也往往较低，使得其难以成为最终结果。如果所有抓取的文章是随机分布的，“愤怒”出现概率如此之高，这是非常不合理的，说明数据集标注的随意性太大。由此可知，数据集的质量对于网络的训练还是有较大影响的，如此次实验中网友随意标注的数据集，难以保证标注准确性，很容易使得最后训练出来的网络具有“偏见”。

5 实验结果

Model(64 Batch Size unless otherwise stated)	**Accuracy**	**F-Score**	**Corr.**	**Epoch**
Regular CNN	61.40%	22.50%	59.73%	17
Regular CNN(256 Batch Size)	**61.76%**	23.41%	60.25%	30
Multi-Channel CNN (Pre-trained Initialization)	58.53%	26.45%	55.11%	15
Multi-Channel CNN (Random Initialization)	59.87%	20.68%	60.77%	20
Simple LSTM	53.73%	28.69%	50.51%	35
Bi-directional LSTM	57.00%	**33.96**%	56.20%	15
Baseline(MLP Simple, 256 Batch Size)	37.61%	14.92%	31.17%	150
Baseline(MLP, 256 Batch Size)	60.73%	28.25%	**61.26%**	30

可以看出，总体效果CNN最好，LSTM次之，而MLP则依赖于具体的方法，使用词向量时其效果也比较好。其中CNN种类中，多通道类型的准确率反而比普通的略有下降，而使用预训练词向量（用gensim直接在训练集语料上训练得到，质量比搜狗新闻词向量差很多）初始化的多通道CNN准确率更低。其中一个原因是训练集太小，将词向量作为可训练参数，很容易就过拟合，这也是多通道CNN只训练了15-20轮的原因；另外，使用较低质量的词向量初始化，也很可能会影响梯度的估计，从而降低准确率。Simple LSTM的表现中规中矩，但与CNN差距还是较大（约8%），不过双向LSTM则改善了LSTM存在的一些问题，从而相比Simple LSTM略有提升。

而Baseline的两种MLP模型，Simple版本不依赖词向量，属于利用词袋信息直接分类的范畴，可以看出效果并不好；利用词向量的MLP竟然能达到CNN水平的准确率，但训练时在训练集上的准确率亦是如此，无法达到90%以上。一个很可能的原因是，MLP的分类能力较大程度上依赖于词向量本身而非网络的结构，使用词向量的MLP准确率较好，很可能是依靠词向量的信息取得的结果。另外，对比F-Score，可以发现MLP和CNN还是有较大差别的，其分类能力相比CNN还是要弱一些。

6 总结

通过本次实验，我了解了NLP领域中常用网络的结构，动手实现它们并进行参数的调整，并且在实验中通过比较不同的结果，对它们各自的特性、优劣有了更深的认识，可谓受益匪浅。对于CNN，之前在自学图像分类的时候有过较多了解，而在本实验中则利用它对自然语言的信息进行处理，并且得到了较好的效果（就Accuracy和Correlation而言），而这也说明了CNN良好的特征提取能力；LSTM结构比较简单，训练过程中没有出现常规RNN的梯度爆炸或消失的情况，其对序列的处理能力还是不错的。

另外，遗憾的是本次实验中数据集标注的偏差较大，输入任意文字时，输出情感为“愤怒”的概率较大，这使得这个模型在其他环境中难以利用。如果能够使用标注质量更高的数据集，可能效果会更好一些。

7 参考资料

1. Chinese Word Vectors https://github.com/Embedding/Chinese-Word-Vectors
2. Kim, Y., 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
3. Long short-term memory. Wikipedia. https://en.wikipedia.org/wiki/Long_short-term_memory