



中国科学院大学
University of Chinese Academy of Sciences

课程作业报告

基于 text-CNN 的电影评论情感分类

作者姓名: 段宏键

学科专业: 计算机系统结构

所在单位: 中国科学院大学计算机科学与技术学院

2020 年 6 月

1. 摘要

本项目是为了实现对电影评论的情感分类。本项目设计并实现了一个 CNN 类型网络的架构（把 text-CNN 并行提取不同尺寸 gram 的特征变为串行化的提取特征，如同图片卷积一样，把输入的句子处理为正规的矩阵，然后当成单通道的图片来处理；其中有一个 embedding 层，然后是四层的卷积，ReLU，最大池化，最后跟三层的 ReLU 激活的全连接层），使用课程资源提供的预处理过的带有 label 的电影评论数据集，经过对数据集的重新划分（我认为原始数据集测试集所占的比例太小了），以及对中文文本内容的分析以及相应处理，然后经过 3 轮迭代在 2000 多条测试集的评论上达到了 84% 左右的准确率、精确率、F1 值和召回率。

1. 介绍

(1)

情感分类又称情感倾向性分析，是指对给定的文本，识别其中主观性文本的倾向是肯定还是否定的，或者说是正面还是负面的。

通常的网络内容会存在主观性文本和客观性文本。客观性文本不带有感情色彩和情感倾向，是对事物的客观性描述；主观性文本带有作者的喜好厌恶等情感倾向，是作者对各种事物的看法或想法。情感分类的对象是带有情感倾向的主观性文本。如果文本中包含客观内容与主观内容时，情感分类首先要进行文本的主客观分类。文本的主客观分类可以以情感词识别为主，利用不同的文本特征表示方法和分类器进行识别分类，对网络文本事先进行主客观分类，能够提高情感分类的速度和准确度。对于主观性文本，目前情感倾向性分析的研究工作主要思路分为基于语义的情感词典方法、基于机器学习的方法以及基于深度学习的方法。

文本情感分类与基于主题的文本分类相似但不同：基于主题的文本分类是把文本分类到各个预定义的主题上；情感分类是按照文本持有的情感、态度进行判断，并不是完全基

于内容本身。

基于机器学习的情感分类，其大致是：首先人工标注文本倾向性作为训练集，提取文本情感特征，通过机器学习的方法构造情感分类器，待分类的文本通过分类器进行倾向性分类。挖掘各种不同的特征能提高情感分类的能力。传统的特征提取方法有信息增益等。常用的分类方法有贝叶斯分类器、最大熵分类器等。使用有监督的机器学习的方法将电影评论分为正向和负向两类。文本情感分类的困难是复杂的情感表达和大量的情感歧义造成的。在基于机器学习的情感分类算法中，每篇文章被转换成一个对应的特征向量来表示。特征选择的好坏将直接影响情感分析任务的性能。

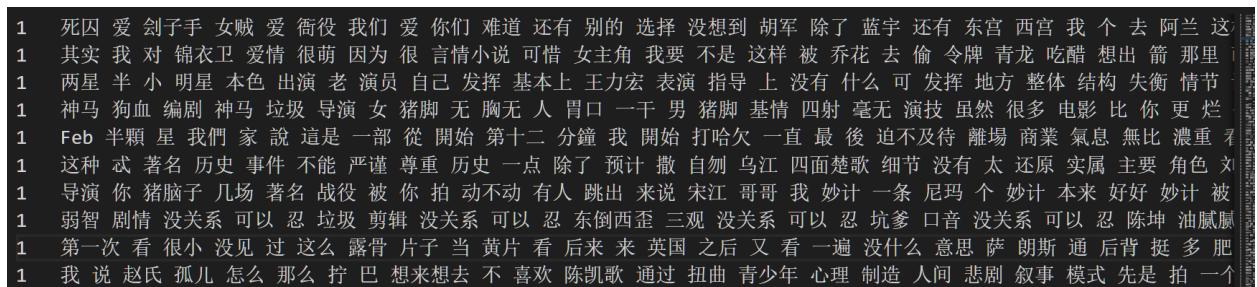
基于深度学习方法主要有两大类，一类是采用 CNN 网络结合传统语言学中的 N-Gram 模型（N 作为卷积核大小），另一种是基于循环神经网络 RNN 或 LSTM 等。深度学习方法把特征提取以及最后的分类均交给网络训练，这样能在机器学习方法的基础上进一步提高分类的准确率，但面临的问题是分类过程“**黑盒化**”，难以知道网络究竟是根据什么来进行分类的。

电影影评一般属于短评，本项目中我们要做的就是将任一条用户影评分类至 [positive,negative] 两个标签，这是个情感二分类问题。

(2)

首先，对于课程网站提供的原始数据集，我认为其对训练集、验证集、测试集的大小划分并不合理，**测试集只有 360 条左右的电影评论，并不足以说明训练模型的情况**。所以我将所有数据集汇总后，随机把所有数据（26000 多条）中的十分之一作为测试集合。

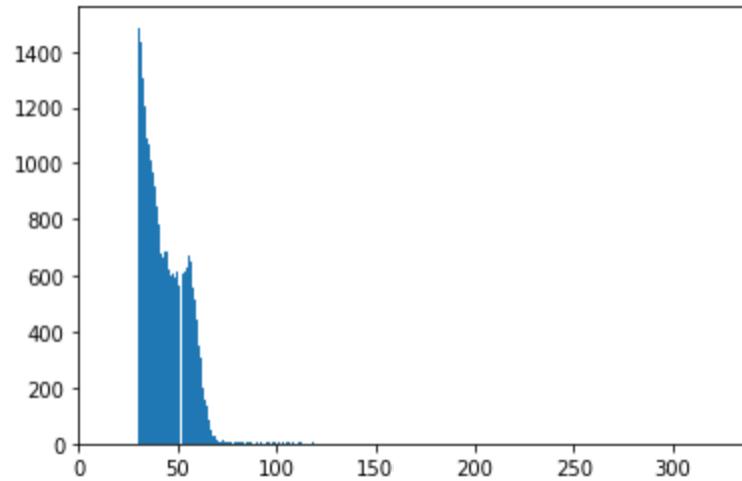
其次，数据集是中文文本，需要进行处理成为数字之后才能进入进行网络训练。



1 死囚 爱 剑子手 女贼 爱 衙役 我们 爱 你们 难道 还有 别的 选择 没想到 胡军 除了 蓝宇 还有 东宫 西宫 我 个 去 阿兰 这
1 其实 我 对 锦衣卫 爱情 很萌 因为 很 言情小说 可惜 女主角 我要 不是 这样 被 乔花 去 偷 令牌 青龙 吃醋 想出 箭 那里
1 两星 半 小 明星 本色 出演 老 演员 自己 发挥 基本上 王力宏 表演 指导 上 没有 什么 可 发挥 地方 整体 结构 失衡 情节
1 神马 狗血 编剧 神马 垃圾 导演 女 猪脚 无 胸 无 人 胃口 一 干 男 猪脚 基情 四 射 毫无 演技 虽然 很多 电影 比 你 更 烂
1 Feb 半颗 星 我們 家 說 這 是 一部 從 開始 第十二 分鐘 我 開始 打哈欠 一直 最後 迫不及待 離場 商業 氣息 無比 濃重 看
1 这种 式 著名 历史 事件 不能 严谨 尊重 历史 一点 除了 预计 撤 自刎 乌江 四面楚歌 细节 没有 太 还原 实属 主要 角色 义
1 导演 你 猪脑子 几场 著名 战役 被 你 拍 动不动 有人 跳出来 说 宋江 哥哥 我 妙计 一条 尼玛 个 妙计 本来 好好 妙计 被
1 弱智 剧情 没关系 可以 忍 垃圾 剪辑 没关系 可以 忍 东倒西歪 三观 没关系 可以 忍 坑爹 口音 没关系 可以 忍 陈坤 油腻腻
1 第一次 看 很小 没见 过 这么 露骨 片子 当 黄片 看 后来 来 英国 之后 又 看 一遍 没什么 意思 萨 朗斯 通 后背 挺 多 肥
1 我 说 赵氏 孤儿 怎么 那么 拧 巴 想来想去 不 喜欢 陈凯歌 通过 扭曲 青少年 心理 制造 人间 悲剧 叙事 模式 先是 拍 一个

除此之外，观察数据集可知，每一个评论句子的长短是不一样的，我统计了一下数据集中所有句子的长度，发现最长的句子有 679 个词，所有句子平均却只有 50 个词，分布极度不平衡，如果采用 RNN 类的网络架构，只要保证同一个 batch 内部的句子数据长度一

致即可，而对于 CNN 类的网络架构，所有的句子长度都要相同才行。所以需要对句子长度做一个处理。

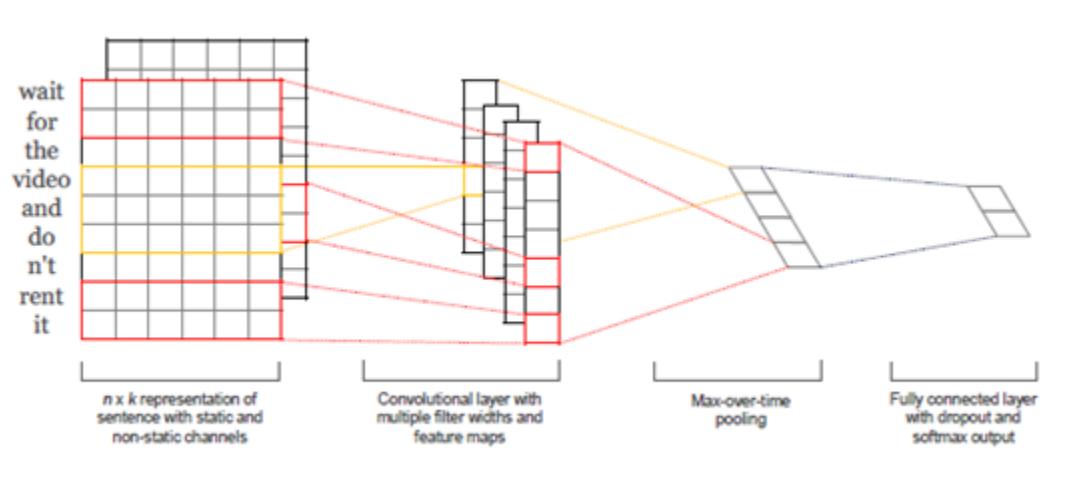


横轴：句子长度； 纵轴：对应长度的句子个数

(3)

为了解决上述问题，需要在训练之前对数据进行了大量的预处理。网络模型采用了 text-RNN 模型的变体，使用多个卷积层来模拟对不同层级 n-gram 的串行提取特征。同时每次卷积之后均使用 ReLU 和最大池化操作来减少特征维度，突出重要特征。最后使用三个全连接层来根据这些特征进行分类。

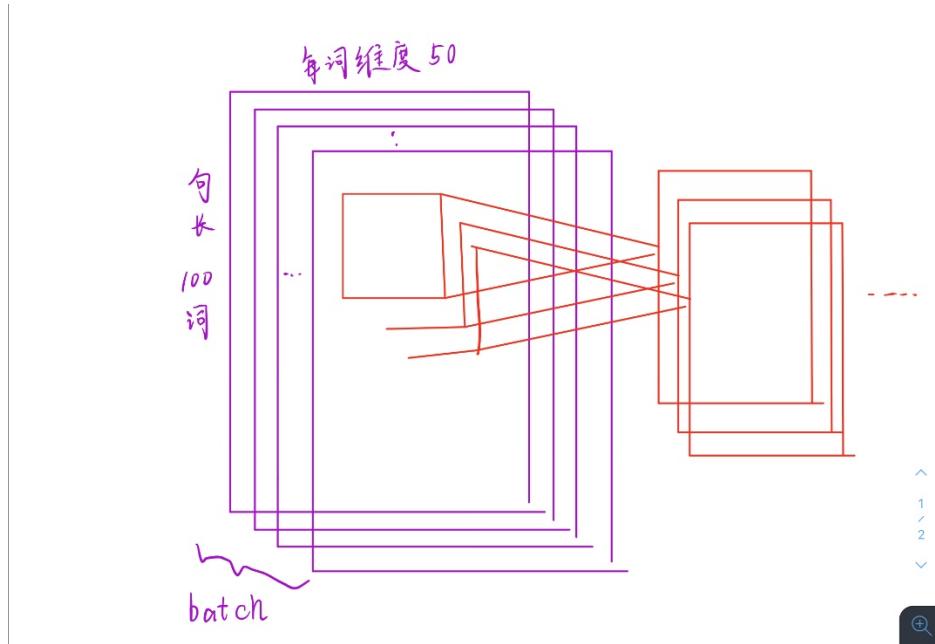
常规的 Text-CNN 模型如下：



常规的 CNN 多层卷积的结构，卷积核的宽度等于词向量的维度，经卷积后可以提取

文本的特征向量。与在图像领域应用类似，Text-CNN 可以设置多个卷积核以提取文本的多层次特征，长度为 N 的卷积核可以提取文本中的 N-gram 特征。

我的模型是对 Text-CNN 的卷积操作做了一些修改。我并没有采用常规的 Text-CNN 的卷积操作。我把输入的 [句子的长度*词 embedding 的维度] 的向量当成一个单通道的图像，在这个图像上做多个多层的卷积，然后通过多层全连接网络把这个输入的“图像”类别输出。这样做的想法是希望追求更细粒度的词与词之间的关系，同时通过不同的卷积层抽取句子中不同层次的特征，以便更好地完成情感的识别与分类任务。



2. 解决方案

由于进行本实验时安装 torchtext 包，导致我本地的 pytorch 环境奔溃了，所以本实验全部在 google Colab 上完成。

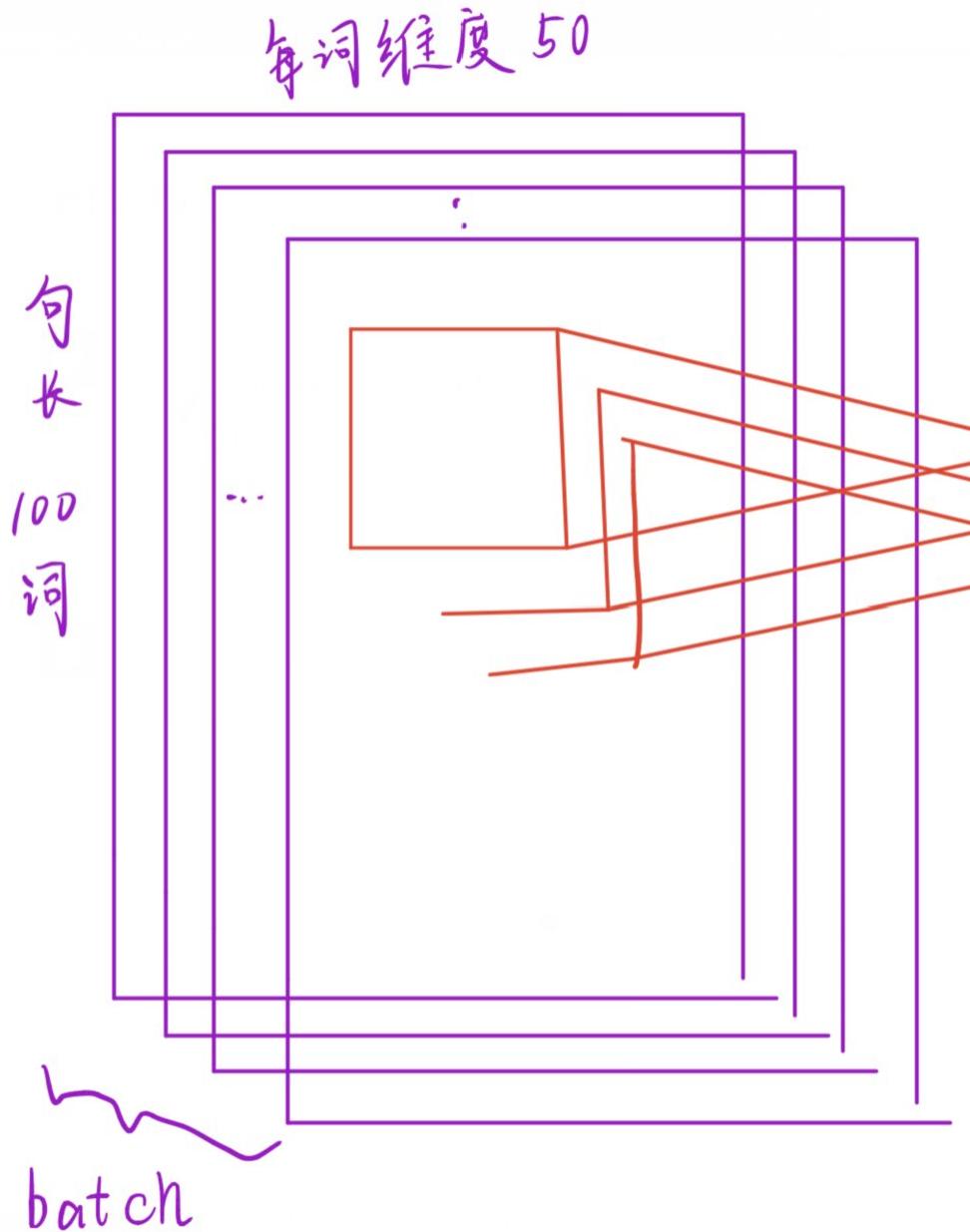
整个实验过程遵循以下几个步骤：

数据准备阶段：

把所有的评论数据以及标签加载进来，对于标签不正常的数据直接放弃（只有几条标签不正常数据，所以放弃这些数据基本无影响）。对评论数据集中的中文词汇构建词汇与

数字 index 相互转化的词表{word: id}与{id: word}形式。然后根据预先训练好的 word2vec 构建训练语料中所含词语的 word2vec，以便作预训练数据输入 embedding 层中。

根据上文“介绍”中第二小节对句子长度的分析，把每个句子的长度设置为 100，这个大概是平均句子长度的两倍，且大多数句子长度都小于等于 100，所以这样设置是合理的。不足长度的句子后续会被填充 0，大于长度的句子会被截取。



网络设计：

网络不是采用 text-CNN 结构，而是把每个句子的数据当成一个图片来进行卷积处理，

这样做的目的是希望能找到更细粒度的关系，同时采用多层卷积来提取高层次间的特征以及寻找特征间的关系。

网络结构如下所示：

```
textCNN(  
    (embeding): Embedding(58463, 50)  
    (conv1): Sequential(  
        (0): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
        (1): ReLU()  
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    )  
    (conv2): Sequential(  
        (0): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
        (1): ReLU()  
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    )  
    (conv3): Sequential(  
        (0): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
        (1): ReLU()  
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    )  
    (conv4): Sequential(  
        (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
        (1): ReLU()  
        (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
    )  
    (fc1): Linear(in_features=2304, out_features=512, bias=True)  
    (fc2): Linear(in_features=512, out_features=100, bias=True)  
    (out): Linear(in_features=100, out_features=2, bias=True)  
)
```

训练优化方法与损失函数如下：

```
EPOCH=3  
LR = 0.001  
optimizer = torch.optim.Adam(cnn.parameters(), lr=LR)  
#损失函数  
loss_function = nn.CrossEntropyLoss()  
#训练批次大小  
batch_size=100;
```

损失函数的设计采用交叉熵损失函数，优化方法是 Adam。

设置了学习率为 0.001，学习率采用固定模式。一共训练迭代了 3 轮（更多轮会过拟合）。

测试部分的设计与训练部分很相似，只是加载的数据集变为了测试数据集，同时去掉了误差反向传播的过程。

测试中除了自己累加计算准确率外，还使用 sklearn 工具来计算准确率、精确率、F1 值、召回率、混淆矩阵。

3. 实验结果和分析

实验配置如下：

深度学习平台：anaconda + pytorch（具体环境包依赖可以参见实验目录下的“环境依赖.txt”）

数据集：课程资源提供的电影评论数据并按照训练集与测试集 9: 1 重新划分。

训练环境：google colaboratory(训练使用的 runtime 是 GPU 类型)

测试环境：google colaboratory，自己的笔记本电脑环境崩掉了

训练过程中会定时输出 loss 值，每轮训练均之后均会进行验证，以防止过拟合的情况出现。

网络模型训练过程中的 loss 变化如下：

```
TRAIN:EPOCH [3/3],batch[100/233],train loss is: 0.11233361065387726, now accuracy is: 0.97
TRAIN:EPOCH [3/3],batch[110/233],train loss is: 0.22163204848766327, now accuracy is: 0.89
TRAIN:EPOCH [3/3],batch[120/233],train loss is: 0.17236606776714325, now accuracy is: 0.93
TRAIN:EPOCH [3/3],batch[130/233],train loss is: 0.17959044873714447, now accuracy is: 0.92
TRAIN:EPOCH [3/3],batch[140/233],train loss is: 0.27104651927948, now accuracy is: 0.93
TRAIN:EPOCH [3/3],batch[150/233],train loss is: 0.1784396767616272, now accuracy is: 0.92
TRAIN:EPOCH [3/3],batch[160/233],train loss is: 0.1748771071434021, now accuracy is: 0.96
TRAIN:EPOCH [3/3],batch[170/233],train loss is: 0.3362520635128021, now accuracy is: 0.89
TRAIN:EPOCH [3/3],batch[180/233],train loss is: 0.10871785879135132, now accuracy is: 0.98
TRAIN:EPOCH [3/3],batch[190/233],train loss is: 0.277923047542572, now accuracy is: 0.89
TRAIN:EPOCH [3/3],batch[200/233],train loss is: 0.18454867601394653, now accuracy is: 0.95
TRAIN:EPOCH [3/3],batch[210/233],train loss is: 0.2693216800689697, now accuracy is: 0.92
TRAIN:EPOCH [3/3],batch[220/233],train loss is: 0.22866123914718628, now accuracy is: 0.92
TRAIN:EPOCH [3/3],batch[230/233],train loss is: 0.10473629087209702, now accuracy is: 0.98
```

可以看到第三轮训练的时候，对一个 batch 为 100 的句子的准确率就比较高了，经验证更多轮次的训练会导致过拟合而是得测试误差增大。

三轮训练之后的测试结果（包括准确率、精确率、F1 值、召回率、混淆矩阵）如下：

```
sklearn: now accuracy_score is 0.8309090909090909, precision_score is 0.8342592592592593, recall_score is 0.823583180987203, f1-score is 0.82888684
sklearn:confusion_matrix is
[[927 179]
 [193 901]]
TEST:in batch[23/26] accuracy is: 0.88, total accuracy is: 0.8330434782608696

sklearn: now accuracy_score is 0.8330434782608696, precision_score is 0.8377659574468085, recall_score is 0.824607329842932, f1-score is 0.83113456
sklearn:confusion_matrix is
[[971 183]
 [201 945]]
TEST:in batch[24/26] accuracy is: 0.81, total accuracy is: 0.8320833333333333

sklearn: now accuracy_score is 0.8320833333333333, precision_score is 0.8367346938775511, recall_score is 0.8234309623430962, f1-score is 0.8300295
sklearn:confusion_matrix is
[[1013 192]
 [211 984]]
TEST:in batch[25/26] accuracy is: 0.92, total accuracy is: 0.8356

sklearn: now accuracy_score is 0.8356, precision_score is 0.84, recall_score is 0.8271704180064309, f1-score is 0.8335358444714459
sklearn:confusion_matrix is
[[1060 196]
 [215 1029]]
TEST:in batch[26/26] accuracy is: 0.91, total accuracy is: 0.8384615384615385

sklearn: now accuracy_score is 0.8384615384615385, precision_score is 0.8457907159716759, recall_score is 0.827559661277906, f1-score is 0.83657587
sklearn:confusion_matrix is
[[1105 196]
 [224 1075]]
```

Accuracy 是当前 batch (100 个句子) 中的准确率, total accuracy 是自己计算的总的准确率, 接下来是 sklearn 工具计算的准确率、精确率、召回率、F1 值与混淆矩阵, 在 2600 多个测试句子上, 总的准确率能达到近 84%。

4. 结论

这个实验我探究了把文本句子处理为标准矩阵之后, 使用 CNN 中处理图像类似的方法来处理句子的 embedding 矩阵, 发现这种处理方法在很少的训练轮次下就能有比较好的识别率, 但是这种把句子当成图像处理的网络的上限在哪里并不知道。相对于 RNN 类型网络的结构, CNN 类型的网络可能在网络设计的层以上没有 RNN 类型的网络自然, 但 CNN 类型的网络由于可以并行化的计算, 所以训练时间会非常短, 而且在较少的轮次就能取得相对还行的效果。同时, 本实验对句子的处理以及网络设计也说明了, 除了图像数据会形成各个层次的特征, 可以用卷积提取外, 文本数据经过 embedding 成为类似图像的矩阵数据之后, 也会有各个层次的特征, 可以使用 CNN 网络结构中的卷积操作进行提取。

更进一步的想法是把文本句子处理为正规的矩阵数据, 这个矩阵数据可以是多层的, 其中每一层的词 embedding 来自于不同预训练样本的 word2vec 数据, 同时加入对矩阵的类似图像的处理 (旋转、翻译、裁剪等等), 然后采用 CNN 网络架构训练和测试。

5. 参考文献

[1] 课程资源《实验四 电影评论情感分类实验指导书》

[2] Kim Y .2014--《Convolutional Neural Networks for Sentence Classification》