实验报告：情感分类

韦俊林 (201928016029023)

2020 年 6 月 28 日

1 摘要

自然语言处理关注计算机与人类之间的自然语言交互。本次实验即是完成一个自然语言处理任务，搭建一个 Text-CNN 网络，在给定的电影评论数据集完成情感分类任务，要求测试准确率在 83% 以上。

在本次实验中，基于 pytorch 框架实现 Text-CNN 网络，使用中文维基百科预训练的词向量，经过调参，最终模型在测试集上达到 84.8% 的准确率，达到了实验要求。

2 问题描述

1. 任选一个深度学习框架建立 Text-CNN 模型；
2. 实现对中文电影评论的情感分类，实现测试准确率在 83% 以上；
3. 也可采用 LSTM 实现，实现测试准确率高于卷积神经网络；
4. 按规定时间在课程网站提交实验报告、代码以及 PPT。

3 解决方案

3.1 损失函数

本次实验使用的损失函数为 pytorch 库里的交叉熵损失函数，pytorch 库中的交叉熵损失函数是将 log_softmax 和 nll_loss 进行了结合，具体计算公式为：

$$\text{loss}(x, class) = weight[class]\left(-x[class] + \log\left(\sum_j \exp(x[j])\right)\right)$$

代码实现如下图所示：

```
criterion = nn.CrossEntropyLoss().to(device)
```

3.2 网络结构

搭建 Text-CNN 网络，网络结构如图所示：

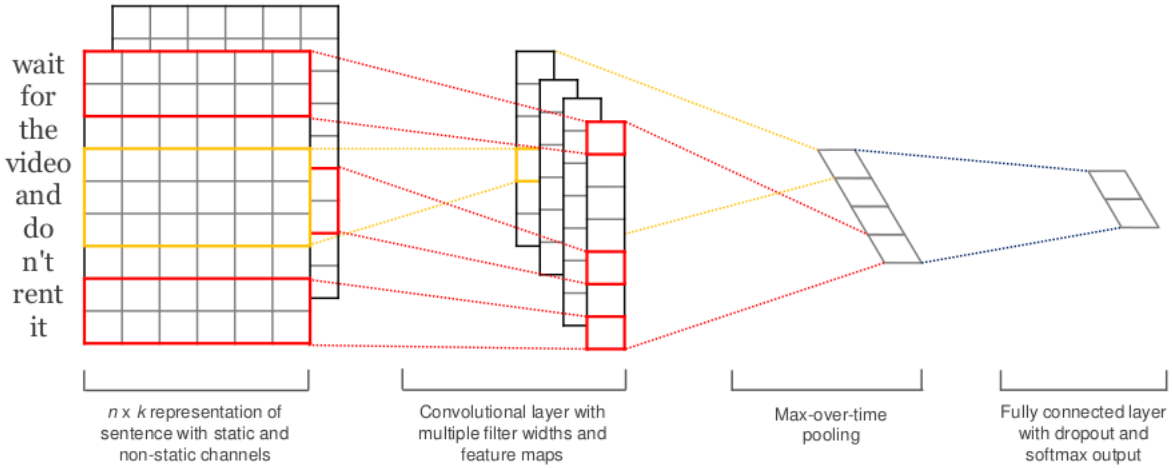


Figure 1: Model architecture with two channels for an example sentence.

详细过程是第一层 Embedding 层，然后经过一个由若干个不同卷积核的卷积核组，接着接上一个最大池化层，网络最后使用全连接层连接并输出。使用 softmax 输出每个类别的概率，全连接层采用 Dropout 算法防止过拟合。

本次实验网络的一些细节参数如下：

在本次实验中，Embedding 使用中文维基百科预训练好的词向量，因此 Embedding 层的词向量是 50 维；

卷积层同时包括卷积核大小为 3, 4, 5 的卷积核，并且每个核大小的卷积有 256 个。因此，卷积层是由 3×256 个卷积核组成的卷积核组；

Droput 层参数 keep 的比例为 0.3；

全连接层权重使用 xavier_normal 初始化，偏置全置 0。网络实现代码以及 terminal 输出如下：

```
  init.constant(self.fc.bias, 0)
TextCNN(
  (embedding): Embedding(58954, 50)
  (convs): ModuleList(
    (0): Conv2d(1, 256, kernel_size=(3, 50), stride=(1, 1))
    (1): Conv2d(1, 256, kernel_size=(4, 50), stride=(1, 1))
    (2): Conv2d(1, 256, kernel_size=(5, 50), stride=(1, 1))
  )
  (dropout): Dropout(p=0.3, inplace=False)
  (fc): Linear(in_features=768, out_features=2, bias=True)
)
```

```
# 使用预训练的词向量
self.embedding = nn.Embedding(vocab_size, embedding_dim)
self.embedding.weight.data.copy_(torch.from_numpy(pretrained_embed))
self.embedding.weight.requires_grad = update_w2v
# 卷积层
self.convs = nn.ModuleList([nn.Conv2d(num_channel, num_kernel, (K, embedding_dim)) for K in model_config.kernel_sizes])
# Dropout
self.dropout = nn.Dropout(drop_keep_prob)
# 全连接层
self.fc = nn.Linear(len(model_config.kernel_sizes)*num_kernel, n_class)
init.xavier_normal(self.fc.weight)
init.constant(self.fc.bias, 0)
```

4 实验分析

4.1 数据集介绍

一个经过分词的中文电影影评数据集，包括训练集、验证集和测试集，同时还有一个预训练好的词向量。

1) 训练集。包含 2 万条左右中文电影评论，其中正负向评论各 1 万条左右。

2) 验证集。包含 6 千条左右中文电影评论，其中正负向评论各 3 千条左右。

3) 测试集。包含 360 条左右中文电影评论，其中正负向评论各 180 条左右。

4) 预训练词向量。中文维基百科词向量 word2vec。

数据集相关代码如下，创建 word2id：

```python
def build_word2id(dataset_path, file_word2id):
    if path.exists(file_word2id):
        word2id = {}
        with open(file_word2id, encoding='utf-8') as f:
            for line in f.readlines():
                sp = line.strip().split()
                word2id[sp[0]] = int(sp[1])
        return word2id
    else:
        word2id = {'_PAD_': 0}
        for dataset in dataset_path:
            with open(dataset, encoding='utf-8') as f:
                for line in f.readlines():
                    sp = line.strip().split()
                    for word in sp[1:]:
                        if word not in word2id.keys():
                            word2id[word] = len(word2id)

        with open(file_word2id, 'w', encoding='utf-8') as f:
            for w in word2id:
                f.write(w+'\t')
                f.write(str(word2id[w]))
                f.write('\n')
        return word2id
```

基于预训练的词向量，创建本次实验数据集的词向量：

```python
def build_word2vec(pretrain_word2vec, word2id, file_word2vec):

    import gensim
    n_words = len(word2id)
    model = gensim.models.KeyedVectors.load_word2vec_format(pretrain_word2vec, binary=True)
    word_vecs = np.array(np.random.uniform(-1., 1., [n_words, model.vector_size]))

    if path.exists(file_word2vec):
        with open(file_word2vec, encoding='utf-8') as f:
            for i, line in enumerate(f):
                word_vecs[i] = np.array(line.strip().split()).astype(np.float)
    else:

        for word in word2id.keys():
            try:
                word_vecs[word2id[word]] = model[word]
            except KeyError:
                pass
        if file_word2vec:
            with open(file_word2vec, 'w', encoding='utf-8') as f:
                for vec in word_vecs:
                    vec = [str(w) for w in vec]
                    f.write(' '.join(vec))
                    f.write('\n')
    return word_vecs
```

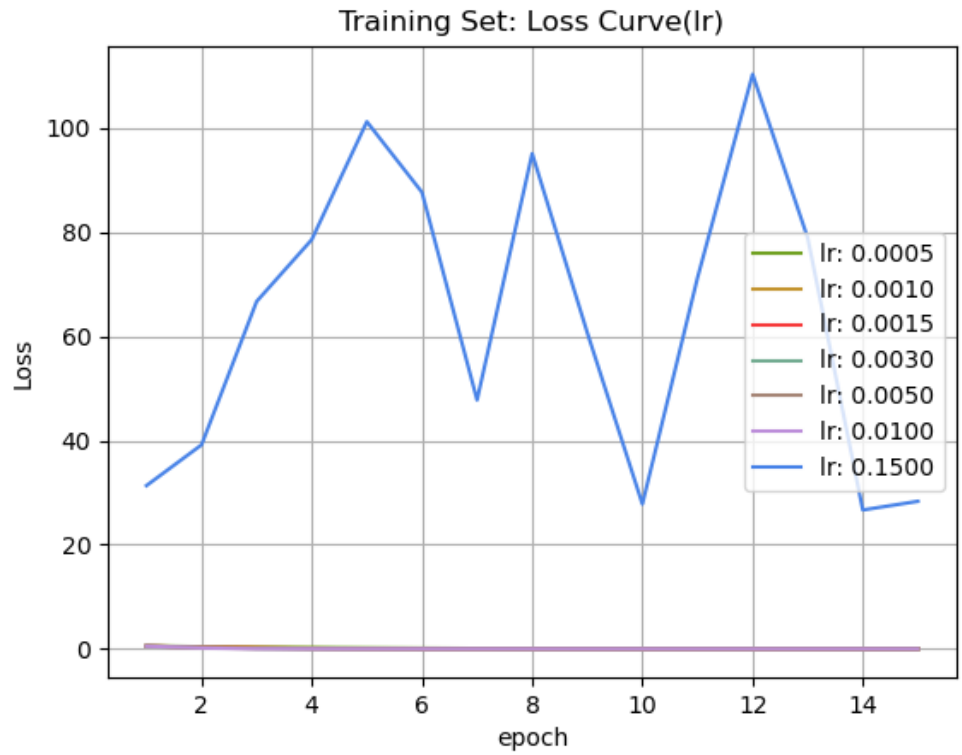
导入数据集，返回相应的 X, y：

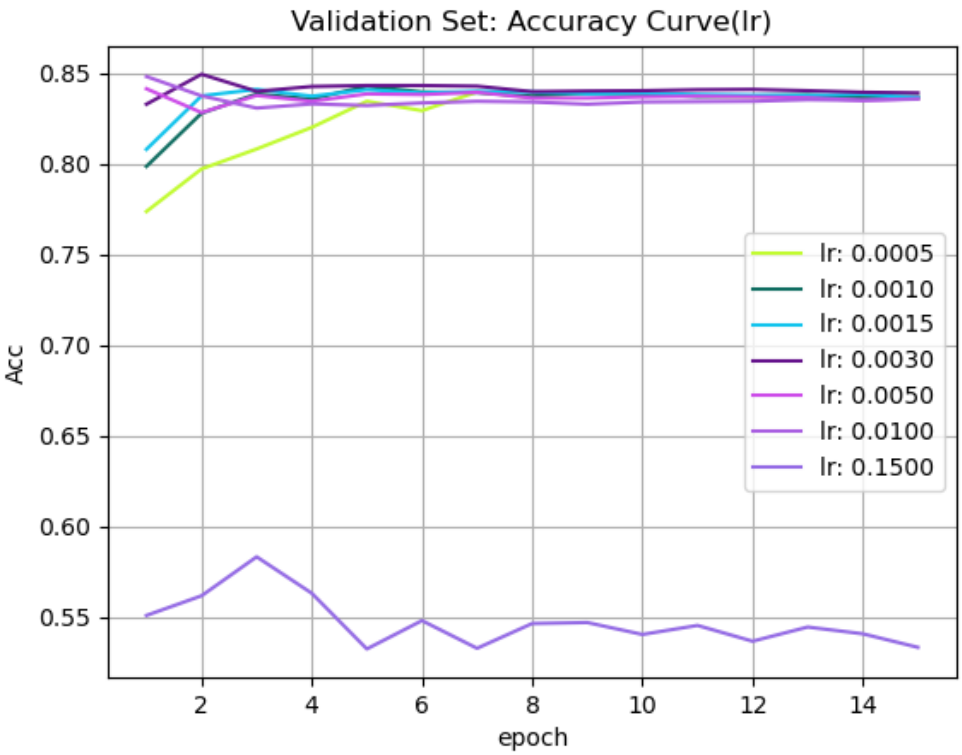
```python
def load_dataset(data_path, word2id, max_len):
    '''
    return numpy: X, y
    '''
    X, y = [], []
    with open(data_path, encoding='utf-8') as f:
        for line in f.readlines():
            data = line.strip()
            if data == '': continue
            data = data.split()

            st = [word2id.get(word, 0) for word in data[1:]]

            st = st[:max_len] if len(st)>max_len else (max_len-len(st))*[0]+st

            X.append(st)
            y.append(int(data[0]))
    return np.array(X), np.array(y)
```

4.2 实验结果

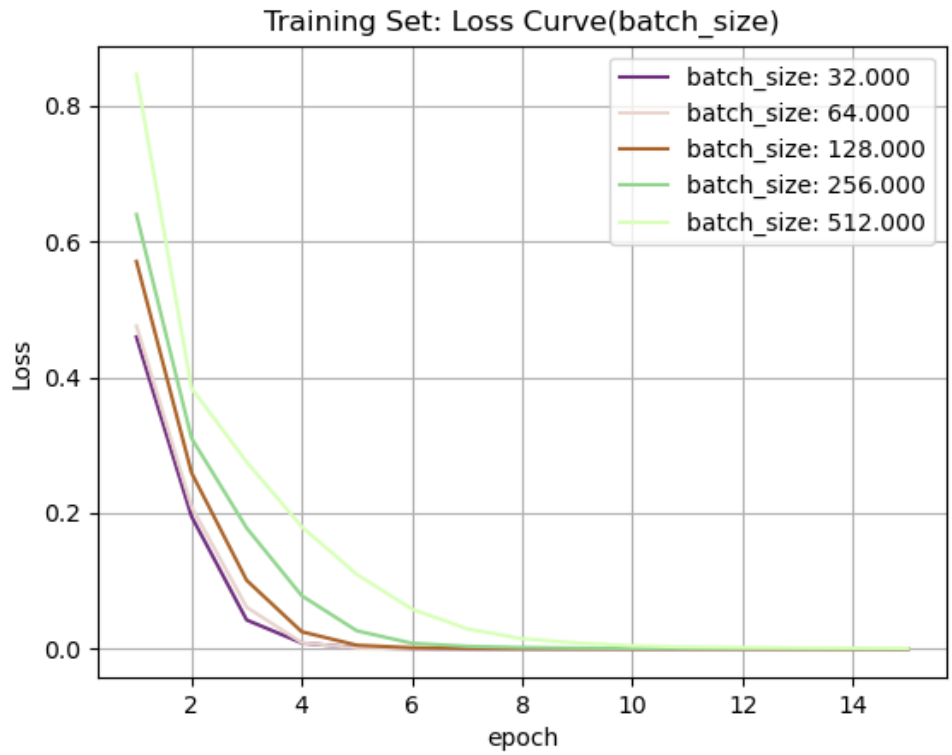
本次实验经过了大量的调参实验，这里只展示调学习率 (lr) 和批量大小 (batch_size) 的实验过程和结果，调参的每个模型经过 15 次迭代，最终训练模型经过 30 次迭代。

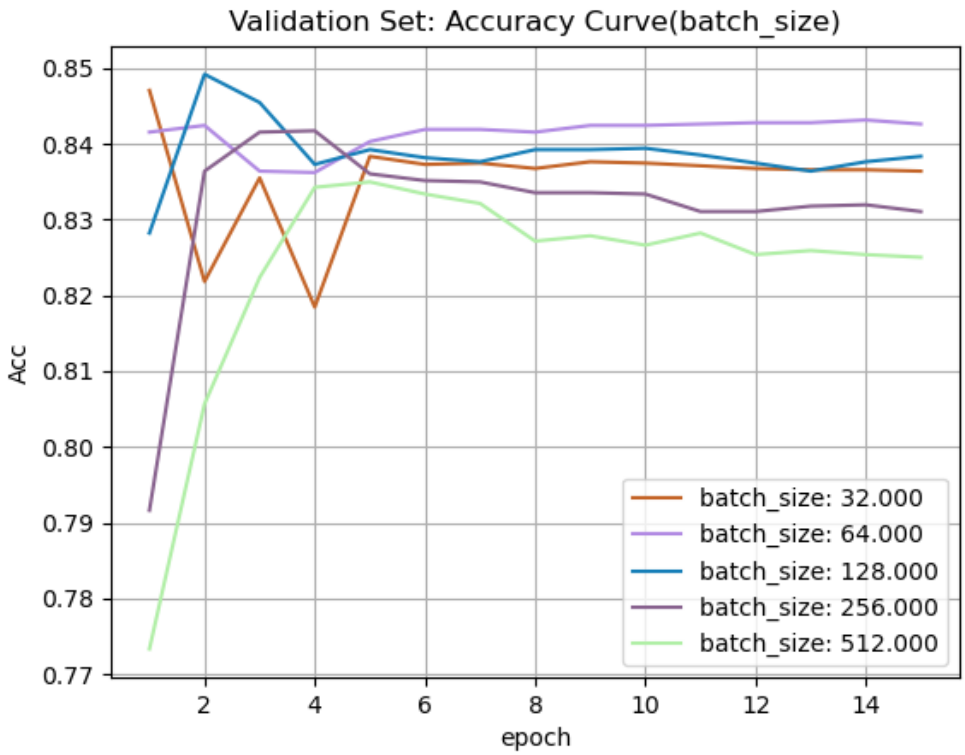
使用不同学习率，训练过程中的损失曲线以及验证集准确率：



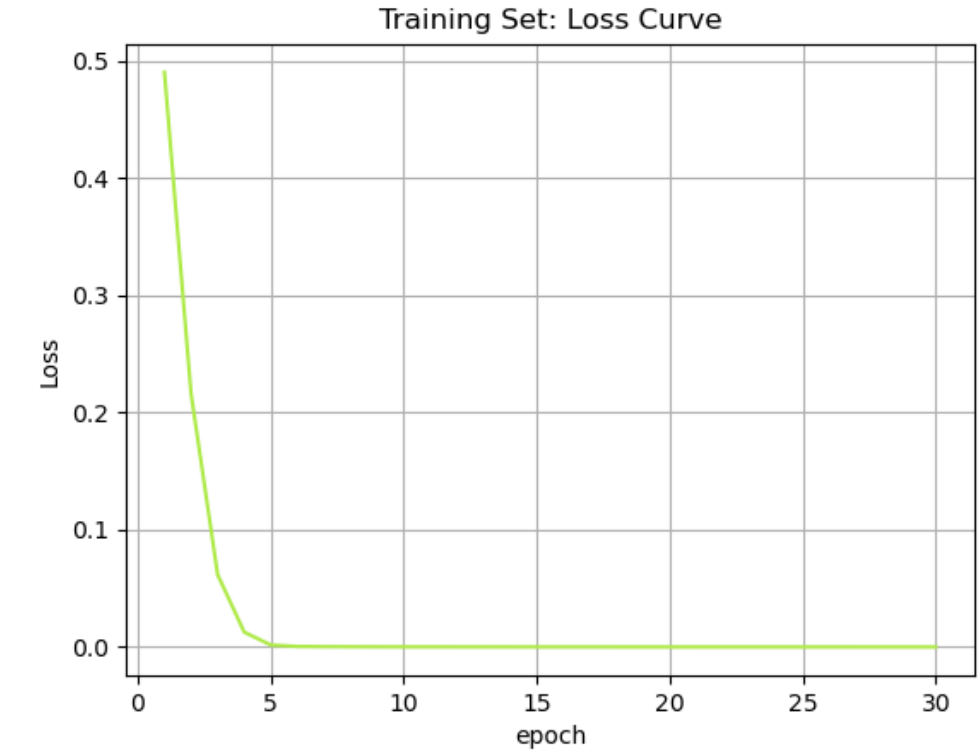


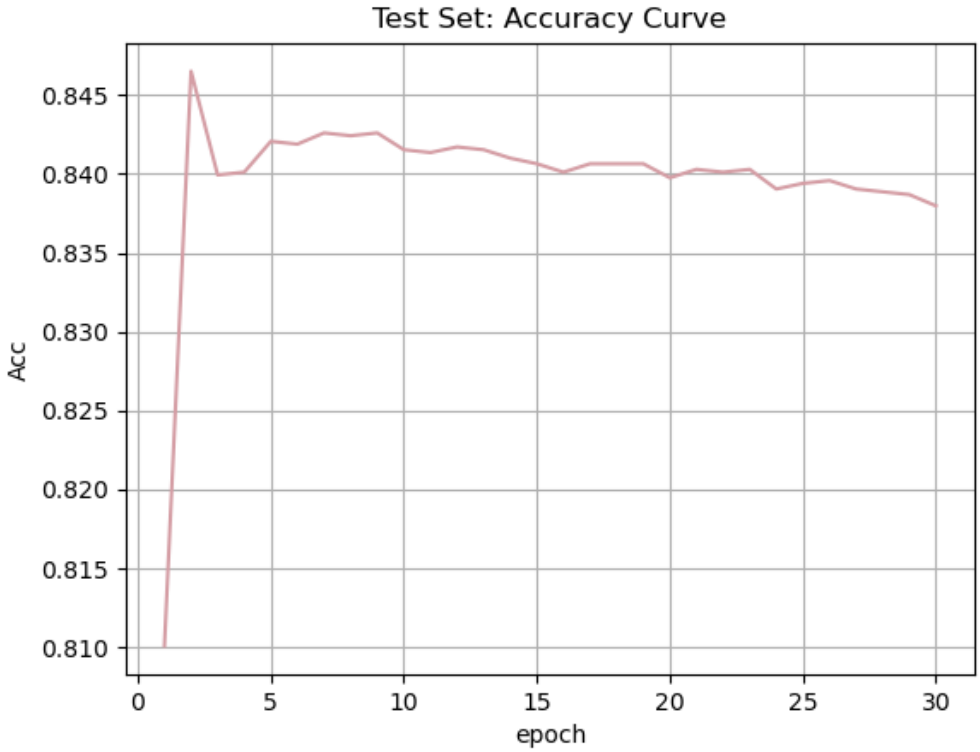
使用不同小批量大小，训练过程中的损失曲线以及验证集准确率：





计算验证集上的平均准确率，选取最优的参数，进行最后的模型训练，训练过程损失曲线以及准确率：





上述的实验过程以及结果都是在训练集以及验证集上测试得到。训练出最终模型后，使用测试集测试模型各项指标，包括混淆矩阵指标、准确率、精确率、召回率、特异率、F1-分数。terminal 输出结果：

```
epoch:[16/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[17/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[18/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[19/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[20/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[21/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[22/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[23/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[24/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[25/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[26/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[27/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[28/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[29/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
epoch:[30/30]   lr = 0.0030, batch_size = 64, train loss = 0.000, test acc = 0.
final model
num_epochs = 30, lr = 0.0030, batch_size = 64, total time = 1315.916 sec
Confusion Matrix:
TP = 152        FP = 21
FN = 35 TN = 161
Accuracy = 0.848        Precesion = 0.879
Recall = 0.813  Specificity = 0.885
F1 Score = 0.844
```

实例验证，从测试集中随机抽取五条数据，实验结果：

```
line: 148
content:        如果 一部 sp 我 给 星 一部 剧场版 水准 只能 星 推理 剧情 太弱 出场 人物 也 显得 不够 丰满 而且 沉睡 毛利 桑 竟然 未 出现 片尾曲 一年 比 一年 差 总之 比较 失望
true label: 1   pred label: 0
line: 36
content:        不 好看 沉闷 就算 演技 差 话 说 年龄 这种 事 不是 穿 校服 马尾 可以 掩盖 李冰冰 还是 修炼 不够 全智贤 一 比 整个 人 弱 掉 HJ 酱油 也 打 毫无 亮点 最 重要 不能 统一 语言 英文 韩文 上海 话 四川话 普通话 杂交 后期 配音 很难
true label: 1   pred label: 1
line: 151
content:        想 说 太 多 反而 显得 假大空 从头到尾 每个 人 流血 方式 一样 镜头感 过 强 显得 摇 拍 范冰冰 永远 一样 表情 谢霆锋 眼神 从来 没有 凌厉 过 憋 出来 前奏 太 长 转折 太 少 显得 相当 不 自然 亮点 小 和尚 音乐 外国人 炮轰 寺庙 而 里面 中国 人 缠斗 不休
true label: 1   pred label: 1
line: 345
content:        曾经 那么 幼稚 曾经 回不去 曾经 但是 想要 谢谢 你 回不去 曾经 留下 最 美好 回忆 你 青春 过程 结局 能够 说 一句 祝你幸福 嘿嘿 总 有些 电影 虽然 故事 很 简单 却 也 最 温暖
true label: 0   pred label: 0
line: 223
content:        一部 好 电影 需要 导演 编剧 演员 用心 专注 导演 专心 雕琢 策划 每 一个 细微 小 情节 设置 异样 波澜 曲折 如 绵里藏针 一般 渐渐 感动 每 一个 观众 如 帮助 一样 这是 一部 导演 编剧 演员 用心 作品 只言片语 细微 情节 中 可 窥见 其 专注 宏大 叙事 借 一个 微小 故事 表达 已有 折射 阳光 伟大
true label: 0   pred label: 0
```

5 总结与分析

之前实验任务的卷积神经网络都是在计算机视觉领域的应用，本次将神经网络应用与文本分类。相比于之前的卷积神经网络，结构没有太大的变化，甚至更加简单了（相比于第二次实验）。

个人感觉本次实验难点在于：第一，相比于第三次实验数据集完全处理好，本次实验经过了从中文文本映射到词向量的过程，更加熟悉了自然语言处理的基本流程以及一些技术细节。第二，虽然 Text-CNN 模型不复杂，但为了达到实验所要求的准确率，调参也是本次实验的重点与难点。