Machine Learning Engineer Nanodegree

项目报告

Christiana, ChristianaShannon@126.com

1. 项目背景

自然语言处理（简称 NLP）属于机器学习的研究领域，主要聚焦于应用计算机对人类的自然语言（文字，语音等）进行描述，处理； 从研究内容来看，自然语言处理包括语法分析、语义分析、篇章理解等。从应用角度来看，自然语言处理具有广泛的应用前景。然而，NLP 的应用有着诸多困难，最主要的一点就是消除歧义问题，如词法分析、句法分析、语义分析等过程中存在的歧义问题，简称为消歧。另外，由于歧义问题的存在，合适的语言处理方法和模型也非常难以设计。

传统的解决 NLP 问题的机器学习方法都是基于浅层模型，例如 SVM 和 logistic 回归，其训练是在非常高维、稀疏的特征上进行的。在过去几年，基于密集向量表征的神经网络在多种 NLP 任务上都产生了优秀成果。这一趋势由词嵌入与深度学习方法的成功所兴起。深度学习使得多层级的自动特征表征的学习成为了可能。因此，深度学习在 NLP 领域的应用前景十分广阔。

2. 问题描述

20newsgroups 数据集是用于文本分类、文本挖据和信息检索研究的国际标准数据集之一。在本项目中，我将分别采用不同文本描述模型（词袋子模型、词向量模型）对文档进行表示，并采用不同的机器学习算法（决策树、SVM、

朴素贝叶斯等）,对数据进行文档分类 ,并在此基础上 ,探索深度学习模型(CNN、LSTM）在本数据集上的应用，最终以分类准确率为各模型评判标准。

3. 基准模型

基准模型采用词袋子模型构建文档描述 ,并采用 Decision Tree(决策树）模型进行文档分类；这是由于决策树模型简单易用，速度快，且易于理解，是分类问题中最常用的模型之一；可以作为后续模型的参照。

4. 评估指标

对于机器学习模型，模型的评估一般使用 PRF（精确率，召回率，F1 值）和 Acc 值（准确率）来评估，本项目中，评估指标采用模型在测试集上的准确率；准确率实际上是所有被正确标示的数据点除以所有的数据点，在本问题中，分类数据不存在明显的偏斜，且对于一个监督学习的多分类问题，“准确分类的比率”可以清晰地表征模型对数据的学习情况及泛化能力，准确率可表示为：

$$accuracy = \frac{\sum_{i=1}^{n} I\left(y_i == \hat{y_i}\right)}{n}$$

用混淆矩阵（Confusion matrix）表示，有：

		实际值	
		1	*0*
预测值	*1*	*True Positive(TP)*	*False Positive(FP)*
	0	*False Negative(FN)*	*True Negative(TN)*

table 1. Confusion matrix

根据混淆矩阵我们可以得到 TP,FN,FP,TN 四个值 ,根据这四个值即可计算精确率、召回率和 F1。其中精确率(Precision)为 TP/(TP+FP) ,召回率(Recall ）为 TP/(TP+FN)， F1 值是精确率和召回率的调和均值，相当于精确率和召回率的综合评价指标，即：

$$F1 = \frac{2 * P * R}{(P + R)}$$

其中 P 为精确率（Precision），R 为召回率（Recall）.

在项目中，对于传统的机器学习，我们通过 F1 指标，可以协助我们对模型的表现能力做出综合评判；另外，对于基于 text8 语料训练的 word2vec 模型，考虑采用简单的几个判例和 PCA 二维散点图来直观判别。

5. 算法介绍：

5.1 文本表征

在传统机器学习中，NLP 处理使用较多的是词袋子模型，其中又包括特征词的频率向量(Count Vector)或者加权词频向量(TF-IDF Vector).

词袋子模型的原理大致为将文本文件划分为单词词典，将每篇文档表示为统一词典长度的一维向量，词典中的每个词对应向量中的一个位置，词的数值即为其 TF-IDF 值，TF-IDF 值计算包括两个要素，一是特征词在文本中的出现频率（TF，Term Frequency，词频），即词频，词频越高，权重越大，二是特征词的文档频率（IDF，Inverse Document Frequency，逆向文件频率），包含该特证词文档越多，表明词越普通，词向量权重越小，即考虑的是特征词的加权；特征词权重函数为：

$$\boldsymbol{\omega_{ij} = Tf_{ij} * Idf_{ij}}$$

Word2Vec 方法则是使用 CBOW 或者 Skip-gram 方式，将文档单词表示为向量形式，将词向量“嵌入”文档，以此来表征文档，以 CBOW 为例： CBOW 模型的训练输入是某一个特征词的上下文相关的词对应的词向量，而输出就是这特定的一个词的词向量。Skip-gram 则正好相反，采用的方法一般是一个浅层的神经网络结构，分为输入层，和输出层(softmax 层)，通过 Word2Vec 将

one-hot 式的词向量化为“语义化”的词向量，建模出词之间的相似度；用图表表述为：

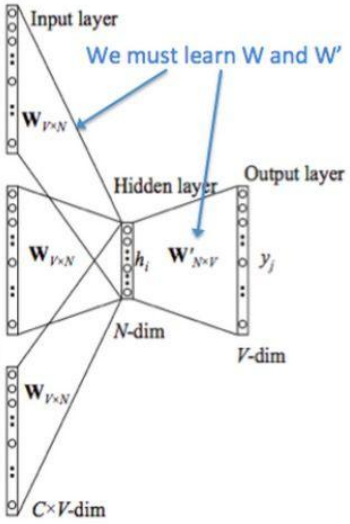

Figure 1. CPOW 模型结构图

5.2　分类算法

我采用了决策树为 baseline 分类器，并实验了朴素贝叶斯分类器（MultinomialNB）、SVM 分类器(Linear SVC), Xgboost 分类器 ,其中决策树、SVM、朴素贝叶斯在课堂上已有介绍，简要概括为：

决策树是通过一系列规则对数据进行分类的过程，分类原则是使无序的数据变的有序，即信息增益最大化，决策树计算便于使用、而且高效，理论明晰，是最常用的机器学习算法。

朴素贝叶斯分类器也称天真贝叶斯分类，是基于贝叶斯定理的特征条件独立的分类器 ,比较适合应用于文本分类中 ,项目中使用了多项式 NB ,适合于 TF-IDF。

SVM 是一个由分类超平面定义的判别分类器，分类目标是获得最大 margin（超平面距离）。SVM 的应用十分广泛，常应用于图像文字识别，数据挖掘等多方面；Linear SVC 采取线性核，算法简单，速度快，在文本分类上效果好。

Xgboost是由 Tianqi Chen 最初开发的实现可扩展 ,便携 ,分布式 gradient

boosting (GBDT, GBRT or GBM) 算法的一个库，是将弱分离器 f_i(x) 组合起来形成强分类器 F(x) 的一种梯度提升方法，相较于其他 boosting 方法，Xgboost 速度快，表现好，且不容易过拟合。

LSTM 全称为长短时间记忆网络（Long Short Term Memory networks），是一种特殊的 RNN，它能够学习长时间依赖，可以解决 RNN 中存在的梯度消失问题，在深度学习中，embedding+LSTM 是一个基础的神经网络层组合。

GRU 全称为 Gated Recurrent Unit，GRU 是 LSTM 的一个变体，仅包含更新门和重置门两个门结构 ,更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，更新门的值越大说明前一时刻的状态信息带入越多。重置门用于控制忽略前一时刻的状态信息的程度，重置门的值越小说明忽略得越多。它保持了 LSTM 的效果同时又使结构更加简单，除保持 LSTM 的特性外，GRU 简化了运算，在大数据量下有明显的运算优势。

CNN 在课程中已有介绍，限于篇幅，在此略过描述。

6. 具体实现：

6.1 数据获取

本项目中采用的数据来源于官方网站：20Newsgroup, 且 sklearn 库中对数据也已近有集成，可以通过 sklearn.dataset 调用，调用方式为：

```
from sklearn.datasets import fetch_20newsgroups
```

在深度学习中要使用 word2vec 词向量表达模型，我使用了 Mikolov 研究的 text8 语料进行词向量训练，数据来源为 text8 官网.

6.2 数据探索与预处理

对每篇文档来说，包括了正文，脚注和引文。由于正文，脚注和引文可能也包含文档分类的关键词，所以我将其全部作文文档内容进行处理。接下来我对数据集进行了一定的可视化，首先观察 20 类新闻的分布情况：

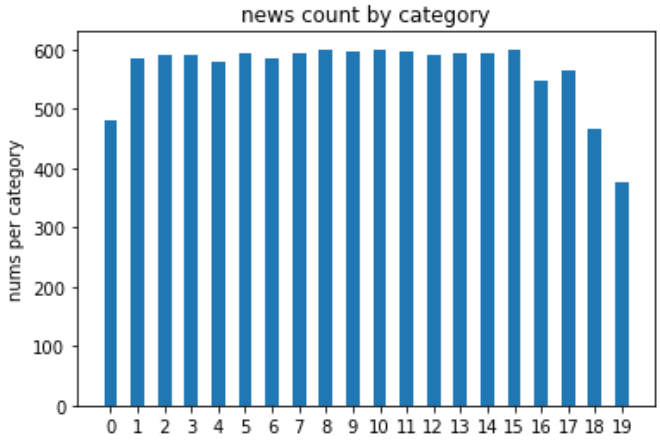

Figure 2. 20 类新闻的分布情况

20 类新闻的数量大部分基本相同，均在 550-600 左右，仅有三组偏少，分别为 480，465，377；数据集比较均匀分布，不需要再考虑偏差取样的问题。

另外，文档长度等价于其包含的信息大小，有必要观察一下文档长度是否有明显的异常值：

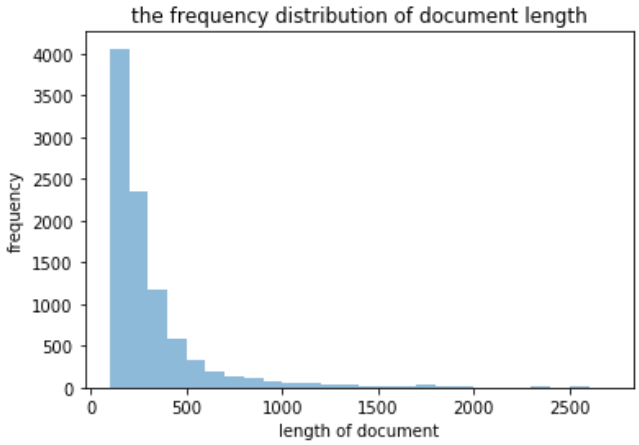

Figure 3. 20 类新闻的文档长度分布情况

绝大多数文档长度在 0-1500 单词之间，文档最短长度 18，最长长度 16197，

中位数长度 191，不需要再进行额外特征处理。

在文本的预处理中，我主要考虑了 4 个问题，分别是标点符号的处理、大小写的处理、停用词的处理及词干提取。

首先，标点符号在文档分类中明显没有用处，应当去除，同样的，英文单词大小写大部分情况并不影响词意表达，将文档中所有单词都转化为小写字母。

对于停用词的处理，可以先观察一下文档中的高频词分布：

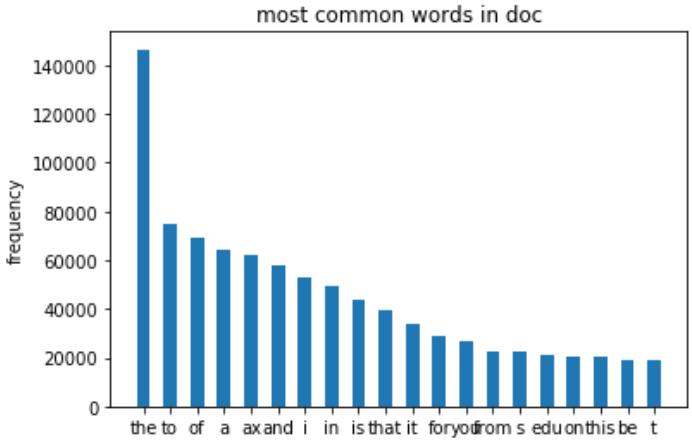

Figure 4. 20 类新闻的词频分布情况

文档中常见词分布情况为: 高频词为各类介词(the,of,to, in)及连词和助词(and, is, i, be) 等，符合常见高频词分布，最高频词为('the', 146532)，没有异常的高频词，可使用常规停用词词典。

最后再考虑词干提取的问题，在英文中，同一词常存在不同形式，例如 gun 和 guns ,eat 和 eating 等 ,理想状态下可以将其处理为同一词 ,我使用了 NLTK 库的 Snowball 方法，对文档进行词干提取，但是提取的结果对词语的完整性产生了破化，产生了诸如’ hungr’ ,’ desir’ 等的残缺词，同时在随后的分类预测中 ,所获得的准确率反而低于不做词干提取的情况 ,因此 ,不再进行词干提取。

6.3 文本模型构造

在文本模型构造上，我在机器学习和深度学习分别采用 TF-IDF 词袋子模型

和 word2Vec 词向量。

依赖于 gensim 库，我使用了 CBOW 模式进行训练，我设定向量维度为 128，我引入了 text8 语料用于词向量模型训练，对模型进行简单判别，显示词向量模型构造成功，可以使用。

```
he is to his as she is to her
big is to bigger as heavy is to heavier
shanghai is to china as paris is to france
```

6.4 分类算法

在机器学习中，考虑到项目为高维数据多分类问题，为了提高效率，SVM 模型中的 kernal 选取了线性核，转化为了 Linear SVC;我使用了 Gridsearch 和 5 折交叉验证对分类器参数进行调整，最后将几个分类器合并成集合分类器，采用柔性构造模式，实验了一下集合分类器的性能。

在深度学习算法中，我引入了已训练好的 text8 word2vex 作为 Embedding 层的权重，并将 Embedding 设置为 trainable=False.经过 LSTM/GRU 训练后，输出对于数据集文档的 20 分类预测。

6.5 模型评估

首先，我采用决策树（Decision Tree）分类器作为基准分类器，在默认参数下验证集**准确率为 0.6634，F1 score 为 0.66。**

默认参数的 MultinomialNB, Linear SVC, XGBClassifier 验证集上的**准确率、F1 score 分别为(0.8825, 0.9046, 0.8065), (0.88, 0.90, 0.81)。**

使用网格搜索 Gridsearch 分别对三个分类器进行优化，分别获得**准确率、F1 score 为：(0.9046, 0.9055, 0.8445), (0.90, 0.91, 0.85)**，可以看出，相对于默认参数，准确度和 F1 score 有了微小的提升，用图表展示：

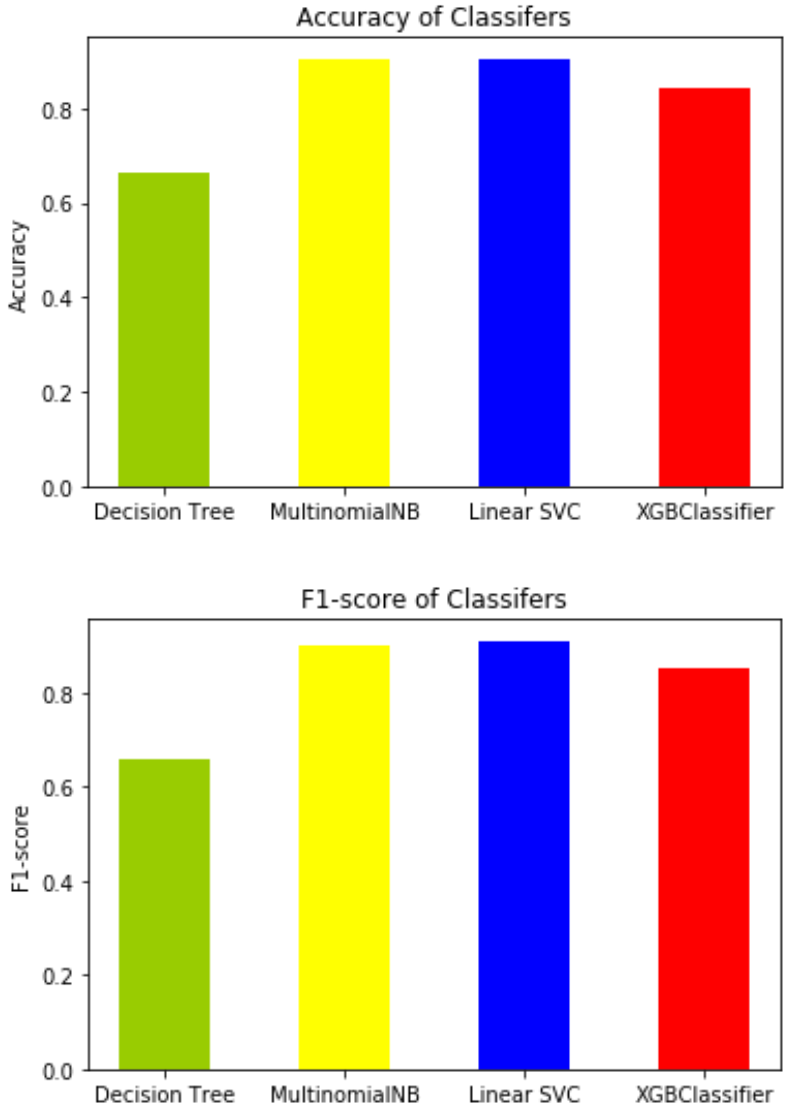

Figure 5. 分类器验证集 acc，F1-score

可以看出，三种分类器在验证集上准确率都超过了基准分类器，且都超过了 80%。

将调优后的三种分类器应用于数据的测试集上，基准分类器决策树**准确率、F1 score** 为：**0.5769 和 0.58**;三种分类器结果依次为：**(0.8132, 0.8247, 0.7661), (0.81, 0.82, 0.77)**，展示为图表：

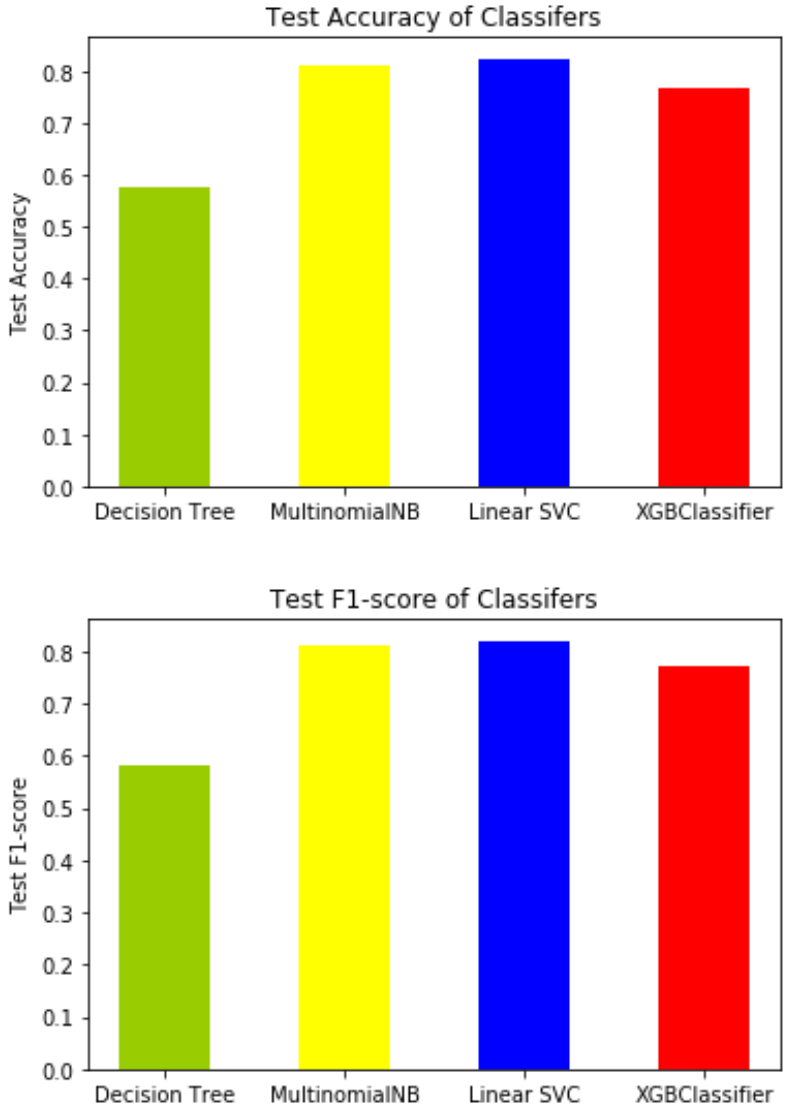

Figure 6. 分类器测试集 acc，F1-score

在项目中 ,我们发现 Linear SVC 的表现最好 ,在测试集上准确率达到了 82.47%，MultinomiaINB 次之，而 Xgboost 的表现要弱于这两个分类器，考虑到 Xgboost 在很多项目中常常与其他分类器一起集合使用，因此我构建了一个柔性的 VotingClassifier 进行分类预测；三个分类器权重均为 1，获得测试集**准确率和 F1-score 分别为 0.7402，0.74。**

综合起来看，机器学习传统分类器在本数据集的 20 分类问题上表现最好的是 TF-IDF 词袋子模型+ Linear SVC 分类器，测试集准确率达到 82.47%，使用

heatmap 进行观察，可以看出在各分类上准确率均超过了 66%.

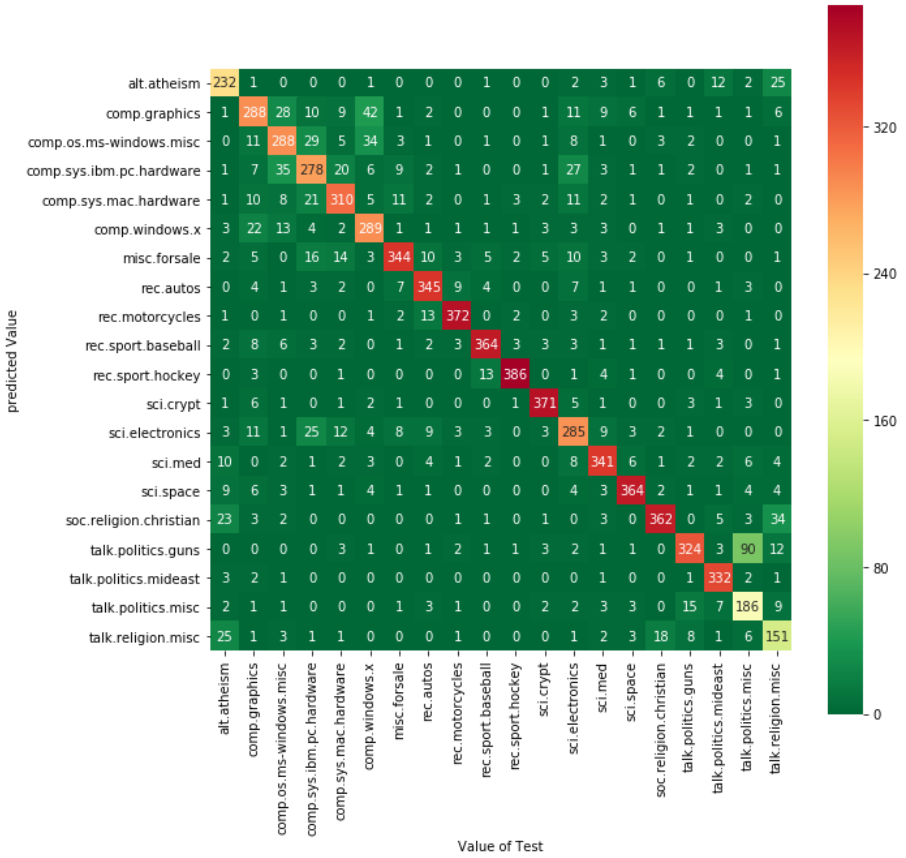

Figure 7. Linear SVC 分类器 HeatMap

在神经网络模型搭建上，我先尝试使用 Keras 文本分类 example 中提供的三层 CNN 模型，在**测试集上准确度为 0.6723，loss 为 1.243**；在此基础上，我将 CNN 层由三层简化为一层（embedding- Conv- Maxpool- LSTM/GRU- Dense - output），最终在 20epoch 训练下，**1-CNN-LSTM 在测试集上准确率为 0.7280，loss 为 0.8217**； 将 LSTM 替换为 GRU，**1-CNN-GRU 在测试集上准确率为 0.7341，loss 为 0.7878**；在此基础上，我将 CNN 层全部去掉，仅构建基本的 Embedding+LSTM/GRU（embedding- LSTM/GRU- Dense - output），在 20epoch 训练下，**LSTM 模型在测试集上准确率为 0.8214 ,loss 为 0.5723**， **GRU 模型在测试集上准确率为 0.8538，loss 为 0.4867**，模型均在 20 epoch 内 val-loss 值逐渐稳定。如图：

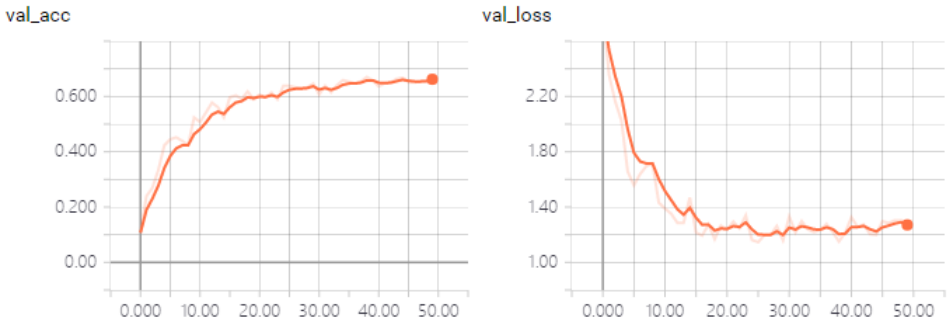

Figure8.1 3 层 CNN 模型 acc 和 loss

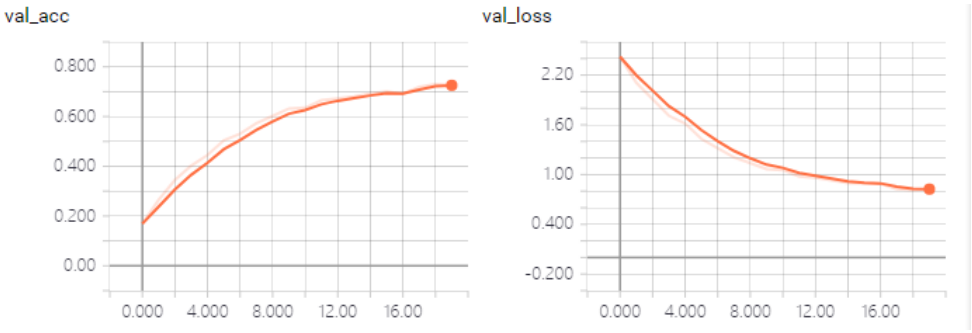

Figure8.2 单层 CNN+LSTM 模型 acc 和 loss

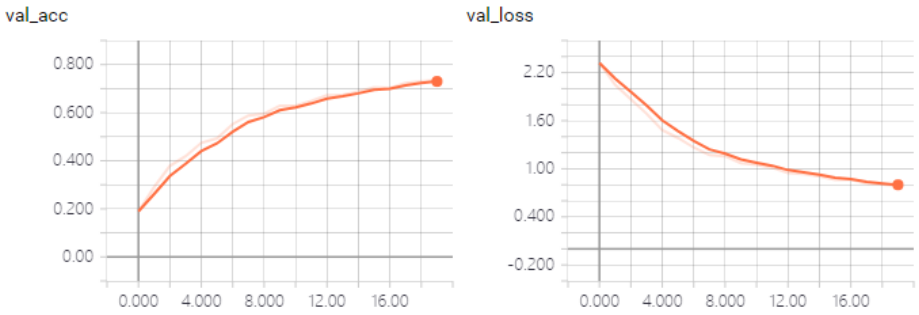

Figure8.3 单层 CNN+LSTM 模型 acc 和 loss

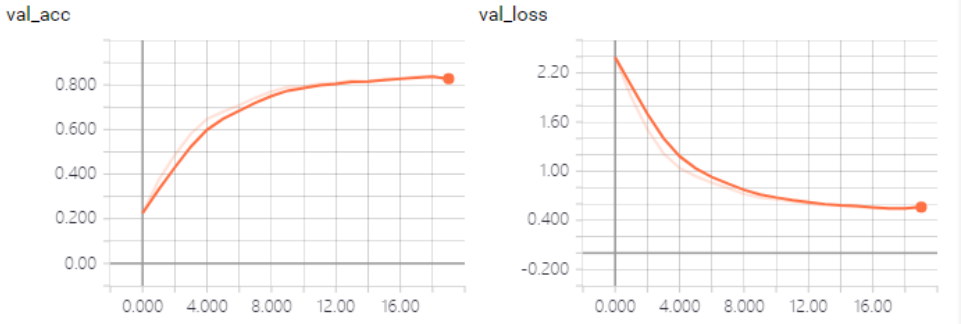

Figure8.4 Embedding+LSTM 模型 acc 和 loss

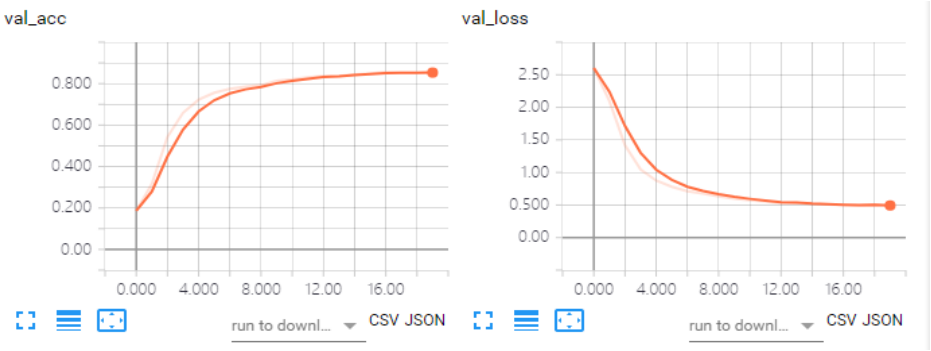

Figure8.5 Embedding+LSTM 模型 acc 和 loss

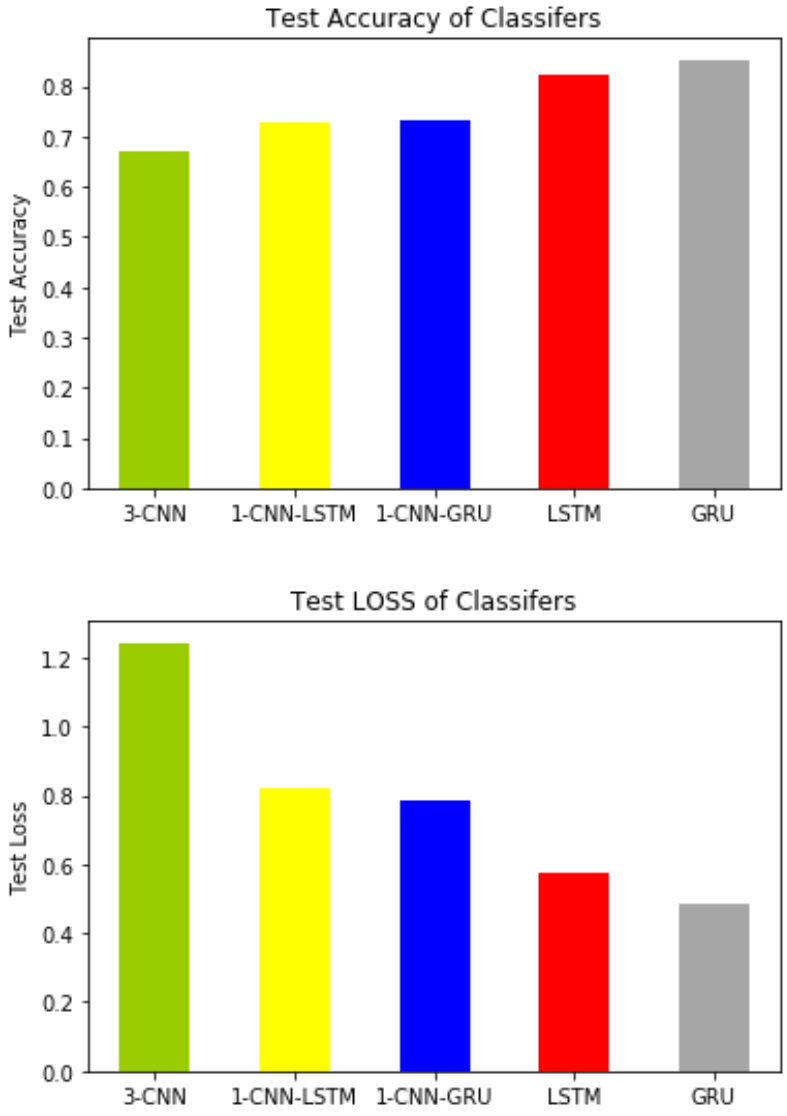

Figure 9. 深度学习模型测试集 acc，Loss

7 问题与提升

在本项目中，传统机器学习算法获得 82.47%的准确率，使用深度学习模型，

获得 85.38%的准确率；可以提升的问题包括两个，第一个是考虑机器学习模型的集成使用，是否存在进一步提高的空间；第二个可能的后续提升是将文本分类问题扩展到中文，尝试中文文本分类。

注：点击引用文献可访问原链接（并不是没有给出链接）

Ref.

1. *Deep Learning, NLP, and Representations*
2. *自然语言处理怎么最快入门？*
3. *深度学习（Deep Learning），自然语言处理（NLP）及其表达（Representation）*
4. *How to Develop Your First XGBoost Model in Python with scikit-learn*
5. *理解 LSTM 网络*
6. *Understanding LSTM Networks*
7. *Distributed Representations of Sentences and Documents*
8. *Distributed Representations of Words and Phrases and their Compositionality*
9. *20 Newsgroups*
10. *精确率、召回率、F1 值、ROC、AUC 各自的优缺点是什么？*
11. *Tf-idf*
12. *CS 224D: Deep Learning for NLP*
13. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*