文本分类实验报告

庞博琛

北京大学信息科学技术学院

1701214092

目录

[文本分类实验报告 1](#_Toc518566594)

[实验环境 1](#_Toc518566595)

[实验要求 1](#_Toc518566596)

[数据分析 1](#_Toc518566597)

[任务分析 2](#_Toc518566598)

[模型设计 3](#_Toc518566599)

[模型评估 5](#_Toc518566600)

[可视化展示 6](#_Toc518566601)

[实验过程与结果 8](#_Toc518566602)

实验环境

由于数据集大小的原因，本次实验主要在实验室服务器上运行，前期的数据预处理部分和sample实验在笔记本上完成。

使用的机器学习框架

Python3.6

Pytorch 0.4.0

可视化部分使用pytorch推荐的Visdom可视化框架，具体细节见后面的可视化部分

实验要求

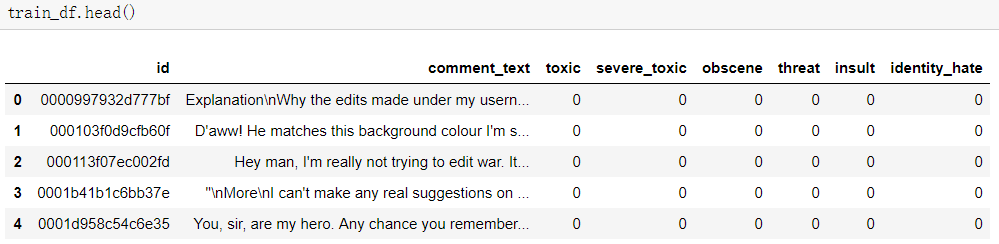
Toxic Comment Classification Challenge

1. 对Wikipedia上用户评论进行文本分析
2. 进行多种类的文本二分类任务
3. 对于每一个评论toxicity的种类有六种分别是['toxic', 'severe\_toxic', 'obscene','threat', 'insult', 'identity\_hate']

数据分析

原始数据一共有4个文件，分别是

train.csv - the training set, contains comments with their binary labels



test.csv - the test set, you must predict the toxicity probabilities for these comments. To deter hand labeling, the test set contains some comments which are not included in scoring.

sample\_submission.csv - a sample submission file in the correct format

test\_labels.csv - labels for the test data; value of -1 indicates it was not used for scoring; (Note: file added after competition close!)

由于任务原先是kaggle比赛，test.csv中没有每条样本记录在六种类型toxicity上的01类别标签记录，test\_labels.csv是比赛结束后网站给出的部分test label数据集的标签

另外数据还有一个问题，即是每种type样本数据的不平衡。

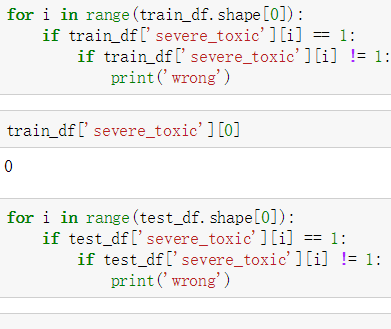
数据预处理

拼接test数据和label的两个数据文件，由于有的test数据记录没有，生成和train.csv格式相同的测试集文件

任务分析

这实验的主题是Identify and classify toxic online comments的文本分析和分类任务，需要判断六种type，['toxic', 'severe\_toxic', 'obscene','threat', 'insult', 'identity\_hate']

六种type中有两个type有很强的相关性，分别是'toxic'和 'severe\_toxic'，通过实验也可以发现当'severe\_toxic'为1时'toxic'也必然为1，二者之间有层次性，可以通过这点优化模型实现逻辑，即在判断'toxic'的基础上判断 'severe\_toxic'，或者在判断'severe\_toxic'时引入'toxic'模型的先验信息。



从模型实现的角度有两种分类的解决思路

* 每个type相互独立，分别训练多个文本二分类的分类器
* 由于每个子问题是二分类问题，将预测看作是一个整体，训练一个统一的模型直接给出评论记录在不同type上的置信度，并进行二分类任务

后续的实验部分分别实现了两种不同的思路

模型设计

文本分类可以拆分成以下三个阶段：文本embedding+feature\_extraction+分类任务。文本分类和图片分类不同，图片的数据本事就是电子化的，有明确的物理光学意义，每张图片可以看作为shape为（w，h，channel）的三维数组，数组中每个值的取值范围是0~255之间的RGB值，具有明确的实际意义。同时表示图片的三维数组可以直接作为神经网络的输入。而对于文本则需要通过embedding进行语义表达。同时本次实验中还实现了网络模型和传统分类器的混合模型

**Embedding：**

原始的文本数据为字符串类型，需要通过构建文本数据集字典进行index化，要将其转化为神经网络需要的数值类型则需要进行Word Embedding操作。Embedding操作可以将一个字符token转换为（embedding\_dim）长度的向量，即将(len\_seq)长度的文本序列转换为(len\_seq, embedding\_dim)的二维数组，常见的Word Embedding操作包括：

1） one-hot embedding：即将word转换为长度为vocabulary\_size的向量，第idx维设为1，其余位均为0

2） 隐主题embedding：结合LDA，PLSA等主题方法得到word在不同隐主题上的分布，并得到embedding vector

3） 基于矩阵方法的embedding：比如glove模型

4） 基于网络的方法：比如word2vec和神经网络方法

本次实验中主要使用了one-hot embedding 和 glove model两种embedding方式，并在lstm模型上进行了对比实验。处于数据集的考虑，得到的字典的大小约为127657，one-hot embedding辅以nn.Embedding()进行降维压缩；glove model使用预训练好的embedding模型。

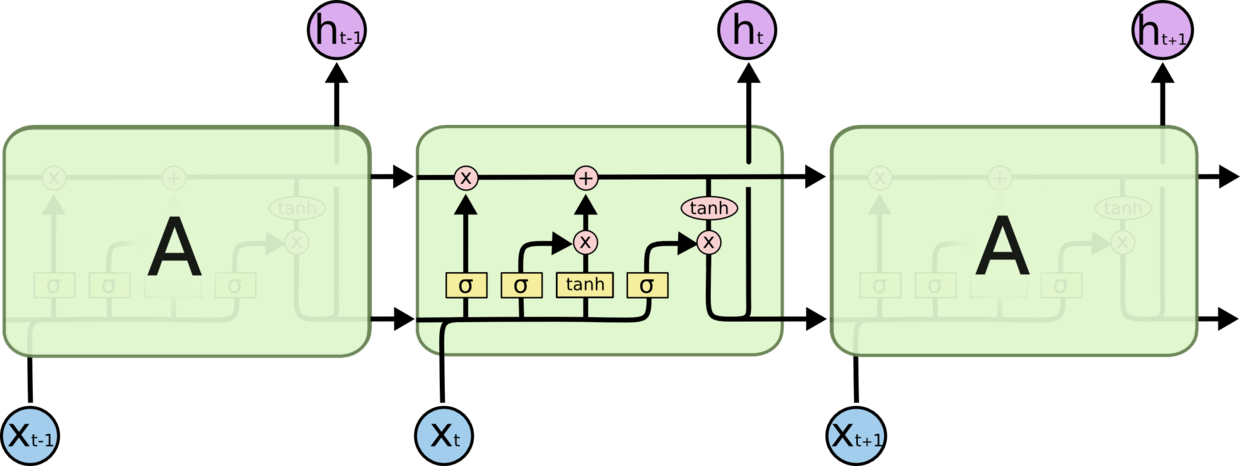
**Feature extraction:**

实验中主要实现了lstm，bilstm, cnn, self-attention四种不同的模型

经过Embedding之后，将(batch\_size, len\_seq)的评论batch字符串集转换为(batch\_size, len\_seq, embedding\_dim)的数组，可以作为神经网路的输入。

可以将不同的神经网络模型看作是文本特征提取器，发掘表示本文语义的特征向量。本次实验中使用RNN循环神经网路如lstm，bilstm，self-attention，此外还就CNN文本分类网络进行实验。

LSTM（Long Short-Term Memory）是长短期记忆网络，是一种循环神经网络。



**网络结构**

LSTM通过对网络结构的设计避免了上述长期依赖（Long-Term Dependencies）问题。LSTM和标准RNN由相似的结构，但是重复的模块具有不同的结构，不同于单一神经网络层，这里是有四个层级结构，并以一种非常特殊的方式进行交互。

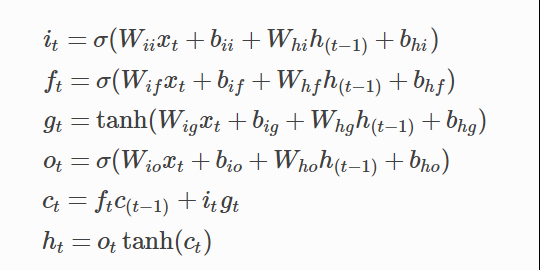
LSTM精心设计了“门”结构来给细胞状态去除或者增加信息。LSTM 拥有三个门，来保护和控制细胞状态。门是一种让信息选择式通过的办法，模拟了信息记忆个遗忘的过程。实现中"门"结构包含了一个 sigmoid 神经网络层和一个 pointwise 乘法操作。sigmoid层输出0至1之间的数值，代表每个部分由多少量可以通过。0代表全部不通过，1则指全部通过。

**BiLSTM**则是双向LSTM网路，是对LSTM的一种扩展

CNN网络结构在前面的实验中已经实现过。有区别的一点是CNN网络需要固定输入数据的长度（比如在图像分类中，需要在数据预处理中将图片转换为大小相同的尺寸），因此文本数据预处理时需要将序列进行截取或者padding的操作。

此外GRU也是常用的RNN网络结构，本次实验没有深入实现。

此外，LSTM模型可能有许多变种，本次实验中使用的LSTM结构为



文本特征（文本表示）的进一步思考

**Self-attention**

在本文中，神经网络可以看作是提取文本表示或者文本特征的一种方法。对于长度为seq\_len的一段评论，CNN网络中后面的layer可以看作是文本表示向量；但是对于RNN网络比如lstm，长度seq\_len的文本会产生seq\_len个hidden\_state，可以使用最后一个作为文本向量，或者使用seq\_len个hidden\_state向量的均值作为文本向量。

从加权机制的角度来看上述lstm的两种方法，第一种为最后一个的权值为1，其余为0；第二种则是每个权值都为1/seq\_len。

因此引入self-attetion机制解决上述问题，关于attention有很多种定义方法，在本次实验中我将其定义并实现为seq\_len个hidden\_state的动态赋权机制。通过self-attention生成更为合理的文本表示。

**特征分类器**

本次实验将文本分类任务看作是embedding，feature extraction，feature classification三部分任务。神经网络中最后的线性层就可以看作是一个线性分类器。

此外还有一种做法：即将神经网络和传统机器学习方法结合在一起，将网络的中间层作为传统分类器的输入。本次实验在这方面也进行了尝试。

由此本次实验实现了mixed\_model，将lstm模型和svm模型结合在一起，在分类器部分使用了在任务分析种提到的第二种思路，使用lstm模型得到的文本表示向量训练6个不同的lstm模型进行二分类任务。另外考虑到文本表示向量的数据含义，我没有选择GBDT-like模型。

本次实验选择LSTM神经网络模型和SVM分类器进行组合，将文本序列的lstm层的输出的均值作为序列文本的特征向量，随后将其作为SVM分类器的输入进行分类。

中间遇到一个问题：即将网络模型的中间层提取出来。

由于我使用的是pytorch框架，主要有以下三种方法：

1. 在forward函数中将网络结果和中间层数据一并返回，这个是最简单的做法，但是会改变原有的模型
2. 新建一个模型，以lstm层为输出层，这个方法很灵活
3. 使用hook方法，结合全局变量机制完成这个任务。

这次实验中我尝试了方法3）这个新的方法，即模型的forward\_hook方法，由于hook函数中不能return，因此使用全局变量完成文本序列特征向量提取的过程。通过和使用方法1）进行对比，我觉得对于提取中间层输出这一个任务，方法1）更加安全合适一些。

模型评估

使用accuracy和AUC两个指标进行模型评估。其中AUC采用pytorch.AUCmeter的计算标准。

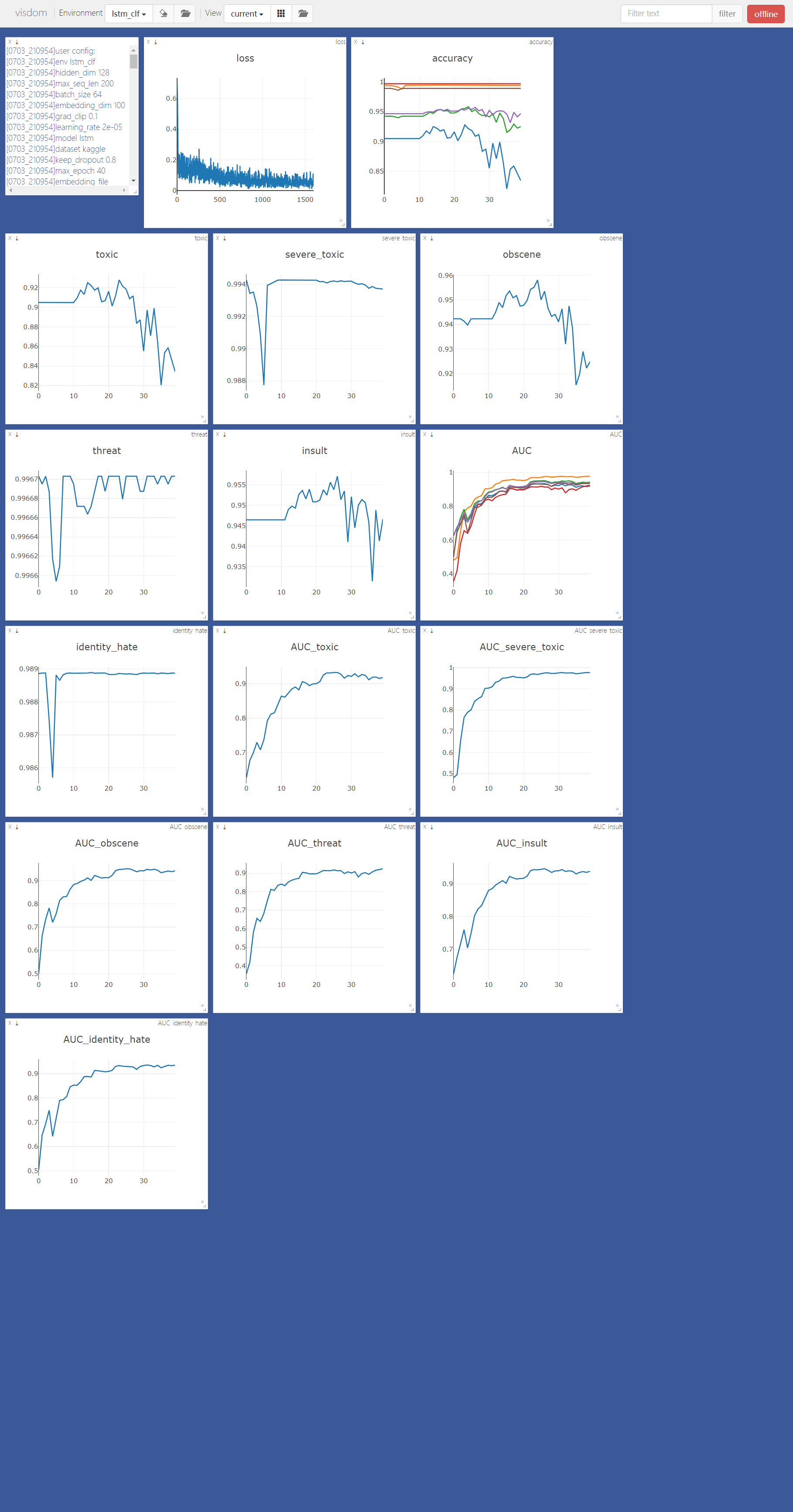
可视化展示

可视化展示包括以下几个部分

* Loss
* Log
* Accuracy on six type
  + Group view
  + Single view
* Auc of model
  + Group view
  + Single view

Take lstm\_clf as example

Next page



实验过程与结果

数据预处理，对原始的文本文件进行预处理，其中特殊的一步就是word embedding，将字符串转换为网络能够处理的数值形式。一开始我使用one-hot编码，但是由于模型比较简单，使用one-hot无法取得较好的结果，因此随后使用pretrained glove model进行训练。

之后构建测试数据集和训练数据集，并结合pytorch框架构建iterator,每次返回(batch\_size, seq\_len)的数据进行实验

模型加载，根据不同的模型设置参数，进行训练

训练过程可视化：和之前的实验相同，使用visdom对中间的实验过程进行可视化。

模型评估：在测试集评估模型性能。

分类模型的扩展：将神经网络和传统分类器结合起来，进行对比。

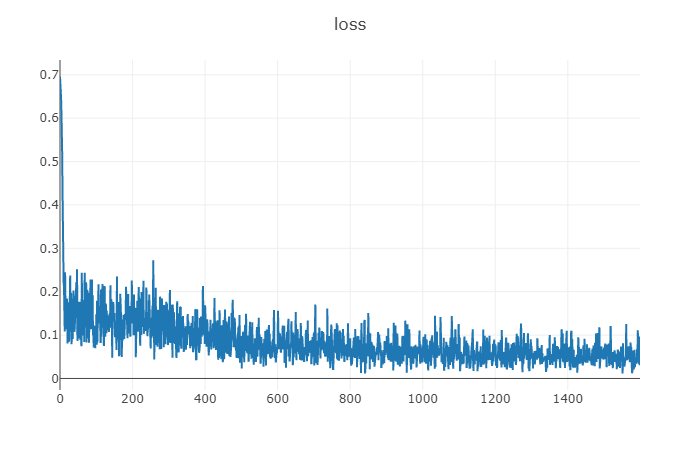
实验过程中进行6组实验

1. lstm\_clf : glove+lstm+fc
2. bilstm\_clf : glove+bilstm+fc
3. cnn\_clf : glove+cnn+fc
4. attention\_clf : glove+self-attetion+fc
5. lstm\_clf\_one\_hot : one-hot+lstm+fc
6. lstm\_mixed\_clf : glove+lstm+multi-svm

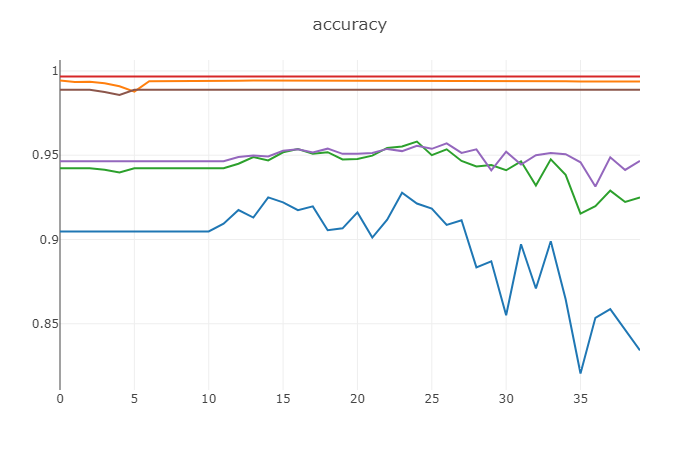
实验过程

**lstm\_clf : glove+lstm+fc**

使用glove预训练模型进行word embedding，使用预训练的weights，lstm模型直接输出评论文本在不同type上置信度，判断是否属于该type



模型在每个type的accuracy



['toxic', 'severe\_toxic', 'obscene','threat', 'insult', 'identity\_hate']

蓝：'toxic'

橙：'severe\_toxic'

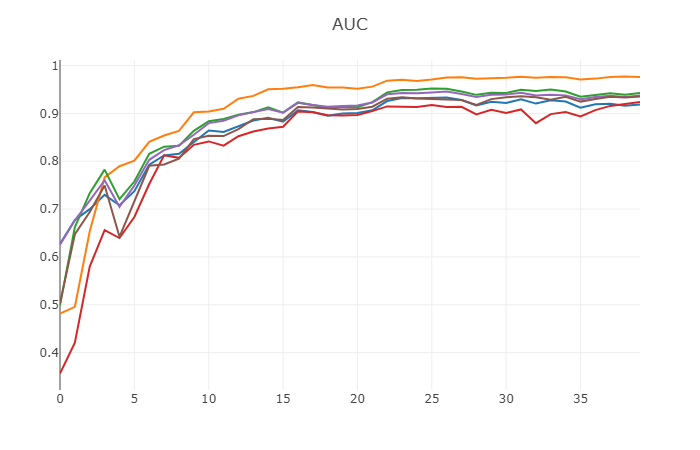
绿：'obscene'

红：'threat'

紫：'insult'

棕：'identity\_hate'

考虑到数据分布的不平衡，计算AUC

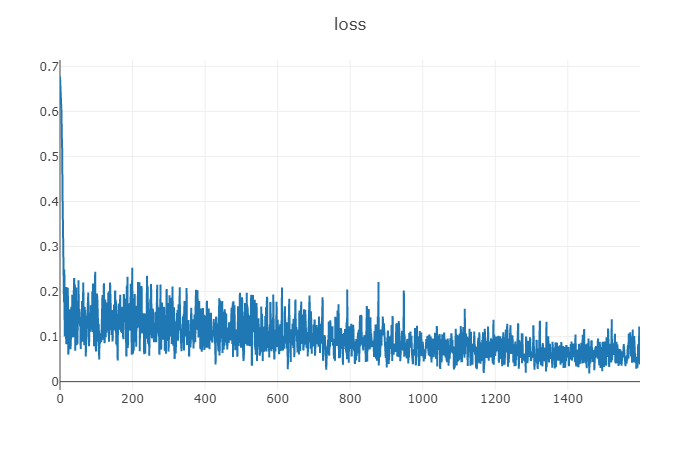


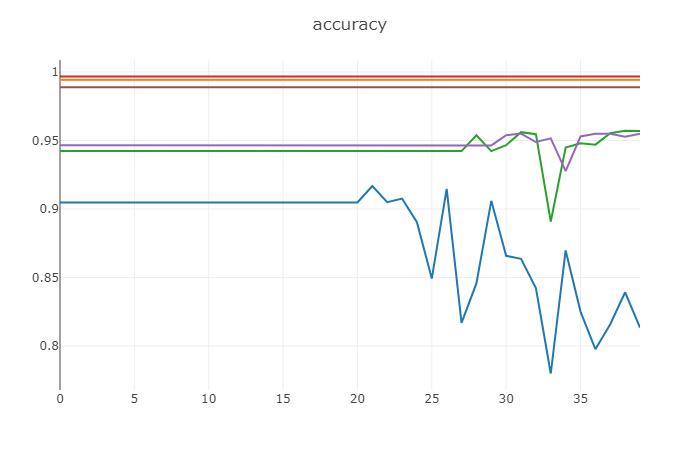
**lstm\_clf\_one\_hot : one-hot+lstm+fc**

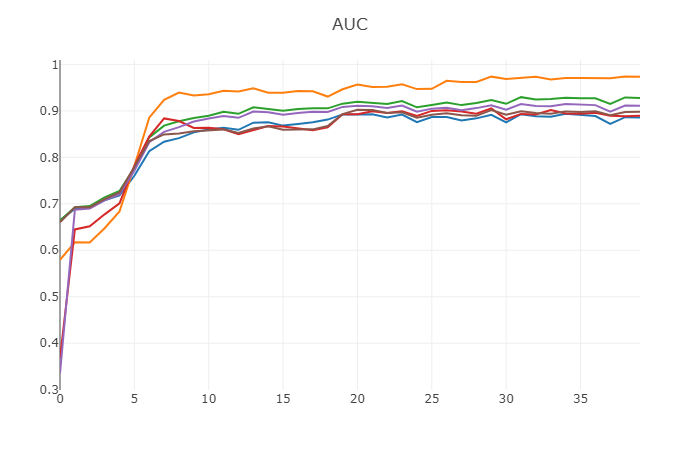
实验中发现one-hot embedding是一种性能很差的word embedding 方法

Vocab的size很大，因此模型增加了额外的参数量，训练过程很慢，小样本集上20min per epoch, 因此后续没有继续采用one-hot方式。

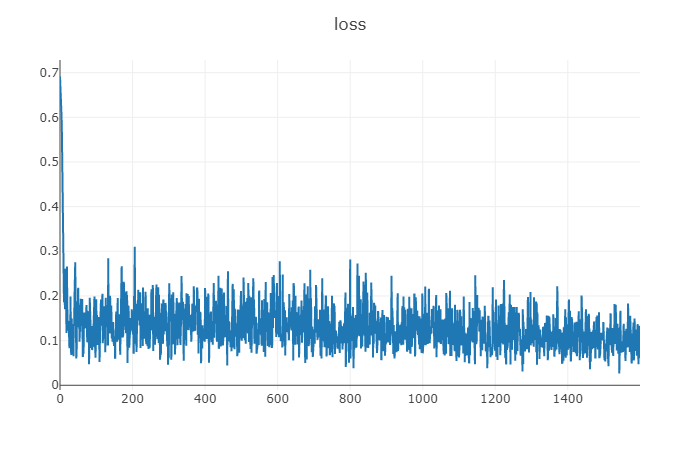
**bilstm\_clf : glove+bilstm+fc**

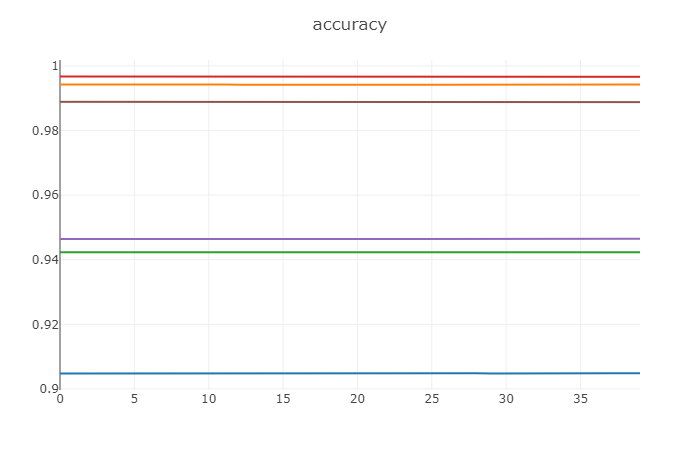




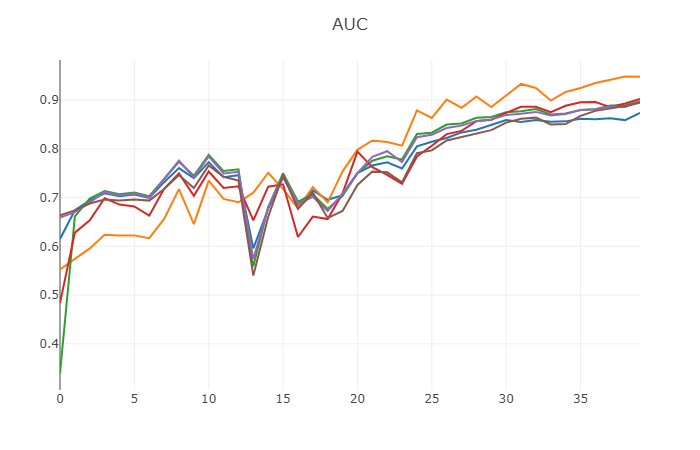


**attention\_clf : glove+self-attetion+fc**

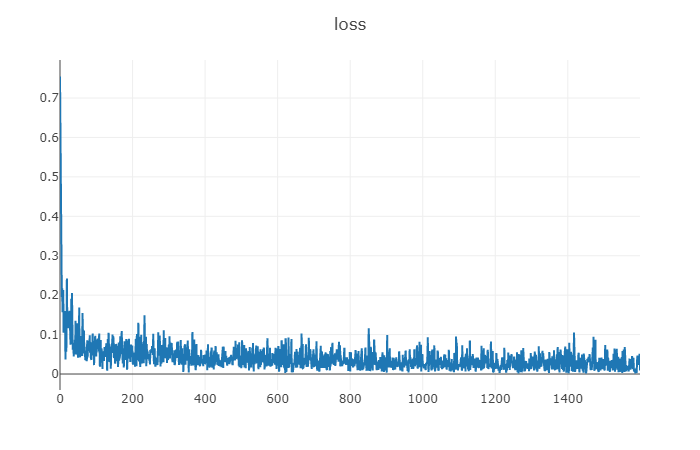


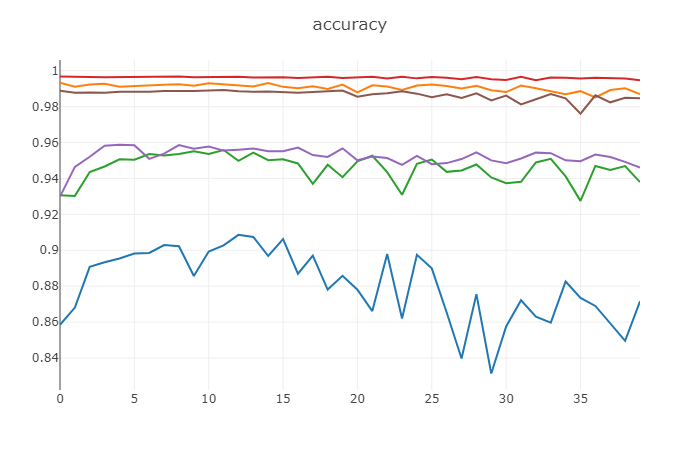


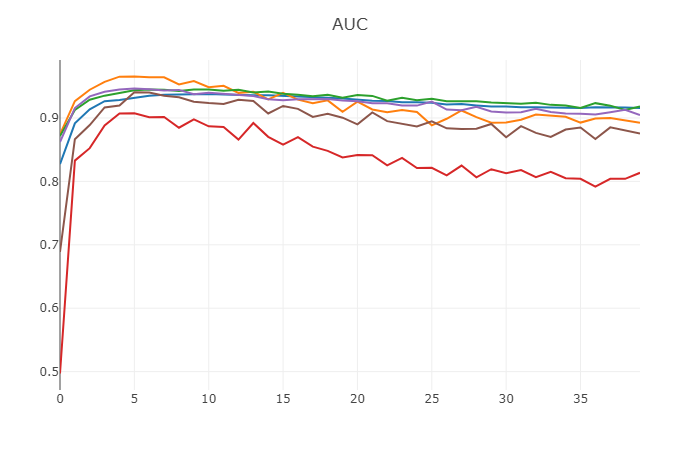
可以看到attention模型非常稳定



**cnn\_clf : glove+cnn+fc**







**Mixed\_model**

将神经网络的模型和传统的分类模型结合在一起，实验效果是否有提升。

具体的做法为：

本次实验选择LSTM神经网络模型和SVM分类器进行组合，将文本序列的lstm层的输出的均值作为序列文本的特征向量，随后将其作为SVM分类器的输入进行分类。

中间遇到一个问题：即将网络模型的中间层提取出来。

由于我使用的是pytorch框架，主要有以下三种方法：

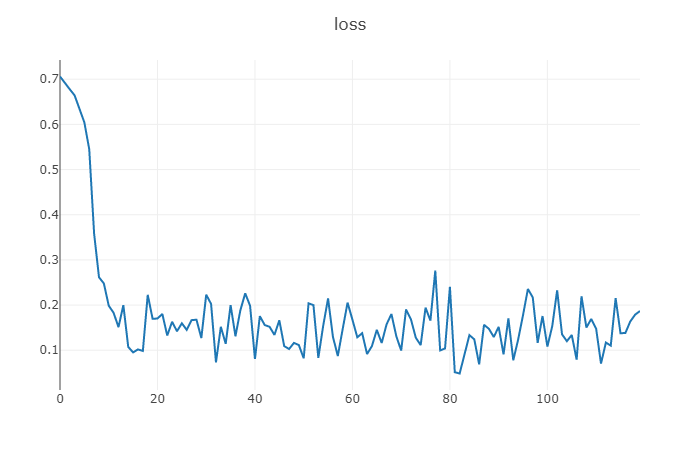
1） 在forward函数中将网络结果和中间层数据一并返回，这个是最简单的做法，但是会改变原有的模型

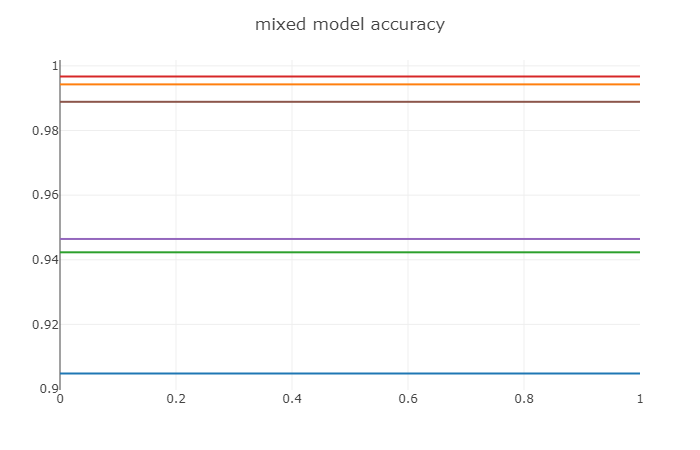
2） 新建一个模型，以lstm层为输出层，这个方法很灵活

3） 使用hook方法，结合全局变量机制完成这个任务。

这次实验中我尝试了方法3）这个新的方法，即模型的forward\_hook方法，由于hook函数中不能return，因此使用全局变量完成文本序列特征向量提取的过程。通过和使用方法1）进行对比，我觉得对于提取中间层输出这一个任务，方法1）更加安全合适一些。

训练过程中的loss情况





实验过程中，发现模型训练时间较长，平均25min per epoch

**实验结果对比**

模型准确率(六类)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| lstm\_clf | bilstm\_clf | cnn\_clf | attention\_clf | mixed\_clf |
| 0.96800 | 0.96418 | 0.96521 | 0.96223 | 0.96223 |

TODO

优化模型的评估方式，改进loss函数