Toxic Comment Classification Report

Definition

Project Overview

恶毒评论分类项目源自 Kaggle 于 2017 年 12 月 19 日发起的竞赛[1]. Jigsaw and Google 希望应用技术来处理最难的地缘政治性挑战任务，即在线审查各种数字的网络攻击，以求减少网络暴力行为。他们提供了免费的机器学习工具来标识恶毒评论，但是依旧存在一些待解决的问题。比如，目前的模型不能让用户选择他们需要区分的恶毒评论的类别。Kaggle 提出的这次竞赛，需要参赛者建立模型，来对恶毒评论进行分类，主要分为以下六类：toxic、severe_toxic、obscene、threat、insult、identity_hate. 待分类及训练使用的评论数据集源自 Wikipedia.

恶毒评论分类问题，属于 NLP(Nature language processing)范畴。NLP 是关注计算机与人类(自然)语言交互的领域[2]。关于 NLP 的研究始于 1954 年，时至今日，已持续有 64 年的研究历程。近年来，有大量的研究表明深度学习在很多自然语言问题上表现的令人满意，比如语言建模，文本解析，机器翻译等[3]。神经网络在 NLP 中的应用始于 2001 年，从最初的自然语言模型，到现在的预训练语言模型，一共经历了多任务学习、词嵌入、NLP 神经网络、seqtoseq 模型、注意力机制、记忆网络等 8 个高度相关的里程碑[4]。Spiros V Georgakopoulos 在 2018 年 2 月 27 日发表的文章[5]中，提供了 CNN 用于恶毒评论的分类的强化学习。Siyuan Li 应用词嵌入技术和 RNN(循环神经网络)于多标签文本分类，以区分各种形式的网络恶毒攻击评论[6]。2013 年 google 公司公布的 word2vec 与训练词向量，以及 Facebook 于 2016 年开源的一个词向量计算和文本分类工具 Fasttext，均得到了广泛的应用。所有的这些研究成果将给本次的恶毒评论分类提供理论基础和应用指导.

恶毒评论分类，根据分类的类别来看，主要是 6 类：toxic、severe_toxic、obscene、threat、insult、identity_hate，显然这六大类不是完全独立互斥的，因此一段评论可能对应于多个类别，属于多标签分类问题。

Problem statement

本次工程旨在解决一个文本分类问题，即需要我们提供机器学习的模型，在 kaggle 提供的训练评论基础上，训练&学习，然后将 kaggle 提供的测试评论按照 toxic、severe_toxic、obscene、threat、insult、identity_hate 六大类别区分开来。

解决一个文本分类问题，主要的流程如下：首先数据预处理，然后做文本的特征提取，最后使用分类算法或者深度学习来对文本进行分类。

数据预处理也是做文本分类的关键一步，合适合理的训练数据，是模型训练的必要条件。而关于数据处理部分，资料显示，主要有以下几种情况待处理，其一，去掉特殊字符，包括标点，stopwords，不涉及表征恶毒文本的词(&/@/ 0-9/表情符号☺/日期/网址); 其二，用正则表达式将诸如 fuckkkkkk 的词语替换成 fuck; 其三，用 TextBlob 改正一些错误的词语；以

及数据增强。

文本的特征提取有多种模型，比如词集模型，词袋模型，词向量模型，TF-IDF 模型。词集模型指的是独一无二的单词构成的集合。词袋模型是在词集的基础上增加频率的维度，词集只关注有和没有，词袋还会关注单词出现的次数。词袋模型是对文章进行特征化的常见方式。TF-IDF(Term frequency-inverse document frequency)模型是词袋基础上衍生出来的一种文本提取算法，本次建模也将使用 TF-IDF 做文本的特征提取。

根据课题的要求，需要分别使用一种分类算法和一种深度学习模型来对文本分类。

分类算法各有不同，常用的分类算法有 Bayes、Decision Tree、SVM、KNN、LR(Logistic regression)等。各种不同的分类算法有各自的优缺点，Bayes 算法比较使用属性之间相互独立的情况，显然这并不适合现有的测试数据；Decision Tree ，KNN 不适合样本分布不均衡的数据集，故而不考虑使用；SVM 比较难以调参。故本次将选 LR 做分类算法。

深度学习来做两种分类模型。深度学习模型建模，需要使用到一些预训练词向量模型，常用的有 freetext、word2vec、GloVe。我们将选用其中两个预训练词向量，应用在本次的模型中。同时适合文本分类的深度学习模型也比较多，例如 TextCNN (Text Convolutional Neural Networks)[6]、LSTM(Long Short-Term Memory)、Attention、GRU 等。TextCNN 是利用卷积神经网络对文本进行分类的算法，这种算法会要求文本信息的长度一致性，不太适合本次数据集的特征。RNN(Recurrent Neural Network)是 LSTM 的基础，他们都是依赖先前的相关信息来预测当前信息，随着依赖信息和预测信息之间的间隔的增加，RNN 的学习能力就会逐渐衰退，故而不太适合很多复杂的学习环境，LSTM 因此而生。LSTM 是一种 RNN 的特殊类型，可以学习长期依赖信息。后期学者对 LSTM 有各种变体的研究，其中 GRU 是 LSTM 的一种改动较大的变体，由 Cho, et al.(2014)提出。它混合了细胞状态和隐藏状态，最终的模型比标准的 LSTM 模型要简单，属于比较流行的变体。本次深度学习模型可能将选用 GRU。

预期结果，需要符合课题的要求，结果达到 kaggle private leaderboard 的 top 20%，从 leaderboard 的结果来看，AUC 大约要达到 0.9863 以上。这里的 AUC 是 kaggle 要求的统一的评价指标，详见下一节。

Evaluation Metrics

模型建立起来之后，如何判断一个模型的好坏就成了最后的关键。由于 kaggle 要求采用 AUC 做为模型的评价标准，因此本次也将计算模型的 AUC 值。AUC (Area under the ROC curve), ROC 曲线一定程度上反应分类器的分类效果，但不够直观，从而引入了 AUC，AUC 实际是 ROC 曲线下的面积。他能直观反应 ROC 曲线表达的分类能力。

- AUC =1，代表完美的分类器
- 0.5<AUC<1, 代表分类器优于随机分类器
- AUC<0.5 代表分类器差于随机分类器

AUC 的计算流程：

- 得到结果数据，数据结构是：(输出概率，标签真值);
- 对结果数据按输出概率分组，得到(输出概率，该输出概率下真实正样本数，输出概率下真实负样本数)。便于分组统计，阈值划分统计;
- 对结果数据按照输出概率进行从大到小排序;
- 从大到小把每个输出概率作为分类阈值，统计该分类阈值下的 TPR & FPR。其中 TPR=TP/(TP+FN), FPR=FP/(FP+TN);
- 微元法 绘制 ROC 曲线,计算 ROC 曲线面积，即 AUC 的值.

AUC 越大说明模型的排序能力越强，从概率的角度解释，AUC 对正负样本的比例不敏感，即

如果训练样本的正负比相差较大(本次训练数据就具有这样的特点)，AUC 的值不会受到影响。而相比于其他评估指标，比如 F1-score，召回率等，负样本采样相当于只将一部分真是的负例子排除，从而模型并不能准确的识别出这些负例，所以下采样后的样本来评估会高估准确率，从而影响到 F1 的值[7]。

Analysis

Data Exploration

数据集提供主要的文件有两个，分别为 Train.csv 和 Test.csv. Train.csv，训练集，用于训练的数据集，一共含有数据 159571 条，每条样本数据有 8 列，分别为 id、comment_text、toxic、severe_toxic、obscene、threat、insult、identity_hate，其中 id 为每条评论的唯一标识，comment_text 即为评论，toxic，severe_toxic，obscene，threat，insult，identity_hate 是恶毒评论的几种分类，也是后期需要预测的值。分类列的值为 1 表示该评论属于对应的恶毒评论，值为 0 则相反。Test.csv，测试集，用于测试的数据集合，一共含有数据 153164 条，与 train 的数据量比约为 0.96. 数据集共两列，id 和 comment_text.我们将需要对 text.csv 中的每条 comment_text 样本按 6 种不同的恶毒评论做预测，sample_submission.csv 提供预测结果的模板。从文本结果来看，每条 comment 的文本长度不一，从下图列举的不同的 comment，就足以显示出各有不同。

```
print(train_data[train_data.toxic == 1].iloc[1,1])
```

```
Hey... what is it..
@ | talk .
What is it... an exclusive group of some WP TALIBANS...who are good at destroying, self-appointed purist who GANG UP any one who asks them q
uestions abt their ANTI-SOCIAL and DESTRUCTIVE (non)-contribution at WP?

Ask Sityush to clean up his behavior than issue me nonsensical warnings...
```

```
print(train_data[train_data.severe_toxic == 1].iloc[1,1])
```

```
Stupid peace of shit stop deleting my stuff asshole go die and fall in a hole go to hell!
```

```
print(train_data[train_data.threat == 1].iloc[147,1])
```

```
Go and hang yourself!
```

```
print(train_data[train_data.insult == 1].iloc[5,1])
```

```
All of my edits are good.  Cunts like you who revert good edits because you're too stupid to understand how to write well , and then revert
other edits just because you've decided to bear a playground grudge, are the problem.  Maybe one day you'll realise the damage you did to a
noble project.  201.215.187.159
```

```
print(train_data[train_data.identity_hate == 1].iloc[200,1])
```

```
I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuc
k niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck nig
gas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas! I fuck niggas!
```

从上图可知，文本长短不一，有高度重复无意义文本，有 ip 地址，特殊符号，大小写，这些都将是需要在训练前做一定的数据处理。

Exploratory Visualization

Kaggle 的赛题上提供了对 train.csv 数据各类别值为 1，或 0 的统计，如图 2。从图 2 的结果来看，值为 1 的评论相对于值为 0 的评论，数据量相差较大，因此训练用的数据集合属于极度不均衡的数据集。且数据集的正样本量极少，也就是说标签值为 1 的数据量远少于标签为 0 的 comment。

图 2

将测试数据中的评论对应的标签数进行统计，可以得到图 3，由图可以知一条样本数据可能被归于多于一个标签。

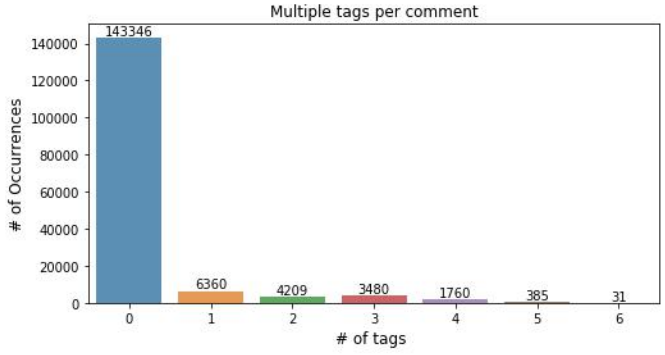


图 3

比较训练集，测试集，我们可以得出数据集合几个特点：

(1) 训练集/测试集 = 1.04;

(2) 训练集没有缺失数据;

(3) 文本长短不一，含有一些特殊字符及与情绪表达无关的符号；

(4) 热力图图 4 显示了训练集中各类别之间的关系，显然 insult 和 obscene 的评论相关性最高，其次是 insult/obscene 的评论与 toxic 的相关性较高；

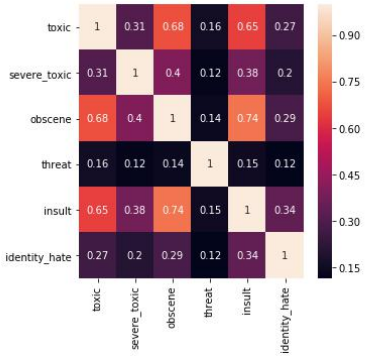


图 4

(5) 图 5 显示训练集与测试集的评论交集状态；两者中包含的词汇有一定的重合，而且 test 的词汇远多于 train 的词汇。

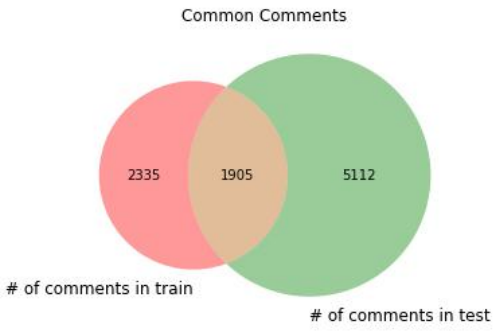


图 5

Algorithms and Techniques

Benchmark

1. 文本特征提取模型分两种。

1.1 传统的 TF-IDF(Term frequency-inverse document frequency)模型。

TF-IDF 是词袋模型基础上衍生而来的另一种文本特征方式，公式如下：

$$TF-IDF(w,d)=TF(w,d)*IDF(w,d)$$

其中 TF(w，d)是词条 t 在文章中出现的概率，IDF(w，d)是语料库总文档数与出现词条 t 的文档总数+1 的比值的对数。主要思想是，如果某个词或者短语在一篇文章中出现频率 TF，词频高，并且在其他的文本中少出现，则认为此词条或者短语具有很好的分别能力，IDF 是指如果包含词条 t 的文档越少，n 就越小，IDF 越大，则说明词条 t 具备很好的类别区分能力。因此本次将会使用 TD-IDF 模型作为文本特征提取的模型。

1.2 配合深度学习使用的预训练词向量模型 freetext 和 glove 或者 word2vec。

据资料显示，word2vec 先于 freetext 问世，两者有很多异同。相同点是两者模型结果类似，均采用 embedding 向量的形式，得到 word 的隐向量表达；类似的优化方法，例如 hierarchical softmax 优化训练。不同之处， word2vec 的输出层对应的是每一个 term，计算某个 term 的概率最大，而 fasttext 的输出层对应的是分类的 label； word2vec 的输入层是 context window 内的 term；而 fasttext 对应的是整个 sentence 的内容，包括 term，也包括 n-gram 的内容。两者皆属于浅层网络，在文本分类任务中，往往能取得和深度网络相媲美的精度，却在训练时间上比深度网络快许多数量级。

另一种 Glove 词向量，它的核心是在于使用此向量表达贡献概率比值。Glove 用词向量表达贡献词频的对数，fastText 用子词向量之和表达整词。

2. 分类模型也分两种。

2.1 传统的 RL 模型. RL 又被称为广义线性回归，常用于分类问题，LR 的实质其实是使用一个 Sigmoid 函数：

$$S(t)=\frac{1}{1+e^{-t}}$$

从上述公式中可以看出该函数具备以下特点：

- 自变量范围是$(-\infty,\infty)$， 因变量(值域) [0，1]；
- 连续可导函数；
- t 为任何一个值，所得到的值[0，1]之间，可以被认为是一个概率值，适合做多分类问题；

预测概率公式如下：

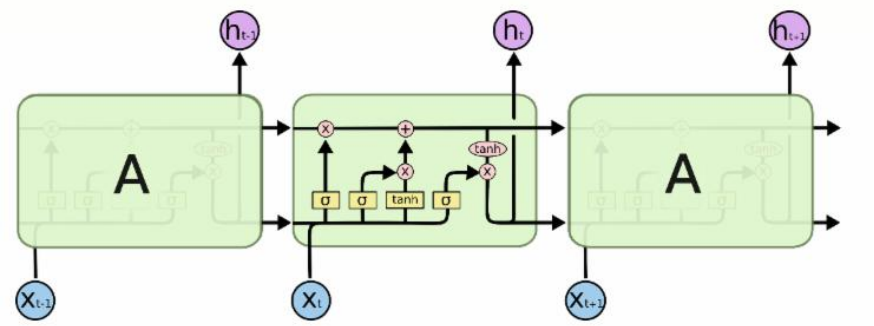
$$P(Y_i=K-1)=\frac{e^{\beta_{K-1}X_i}}{1+\sum_{k=1}^{K-1}e^{\beta_k\cdot X_i}}$$

其中 β_k 是需要计算的回归系数.该系数可以通过最大后验估计(MAP)来计算，也可以通过

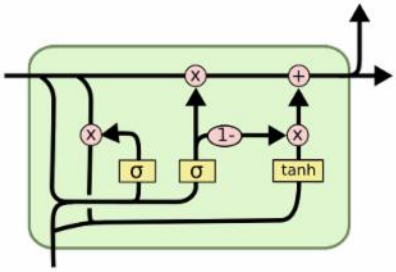
其他方法，例如梯度算法等来计算。

2.2 深度学习模型 – GRU[7]

GRU 是 LSTM 的变体之一，LSTM 的核心思想是通过刻意的设计来避免长期依赖问题。记住长期的信息在实践中是 LSTM 的默认行为，而非需要付出很大代价才能获得的能力。LSTM 的结构如下：



LSTM 引入了一个叫做细胞状态的连接，这个细胞状态用来存放想要记忆的东西，同时在里面加入三个门：忘记门(控制是否遗忘，在 LSTM 中以一定的概率控制是否遗忘上层的隐藏细胞状态)、输入门(负责处理当前序列位置的输入)、输出门(决定什么时候需要把状态和输出放在一起输出)。GRU 就是将忘记门和输入门合成了一个单一的更新门。同时混入了细胞状态和隐藏状态，以及一些其他的改动。最终的模型比标准的 LSTM 要简单。虽然 GRU 比 LSTM 少了一个状态输出，但是效果几无变化.



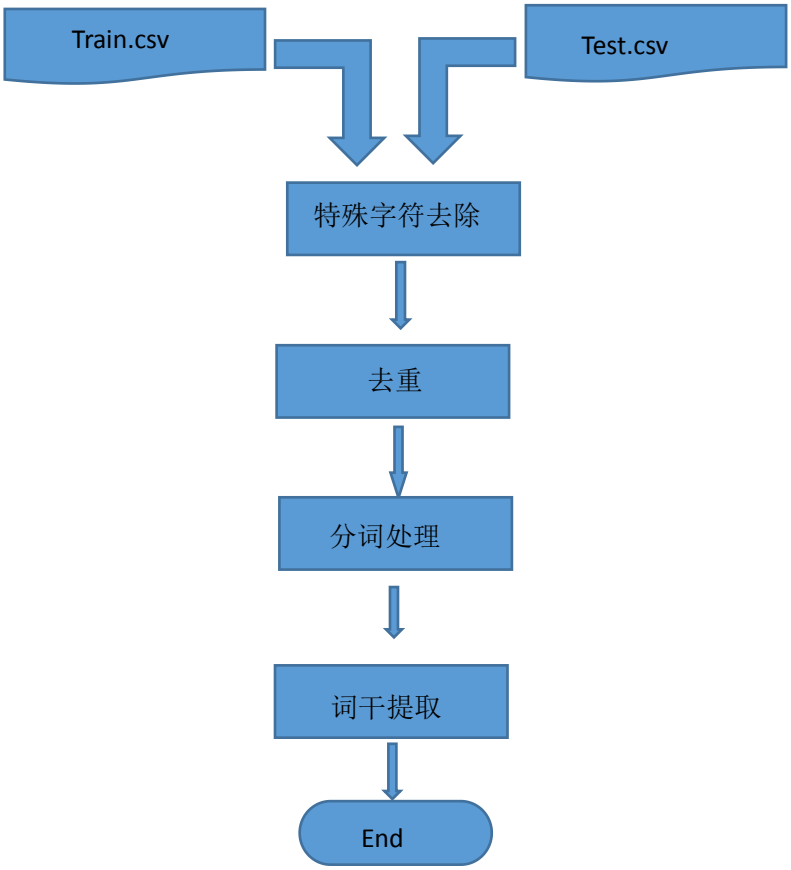
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

LSTM 的流行变体有很多种，Greff, et al.(2015)对这些流行的变体进行了比较，效果的差异性并不大。

Methodology

Data Preprocessing

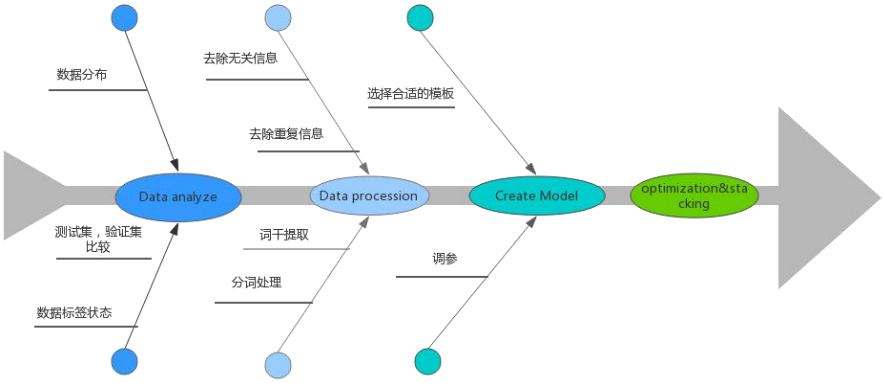
根据之前的数据分析，本次数据处理将采用下述流程进行处理。事实上，除了流程图中显示的几种数据预处理的内容之外，还有很多可能有助于提高 AUC 的方法。比如多国语的翻译，从 kaggle 比分板上显示的第一名提供的算法思路来看，他有用到将文本翻译：英语--法语/德语等--英语，用以进行数据增强，从理论上来说，这样的处理确实有助于增强文本量，但是应该也会带来一定的过拟合现象，不过因为时间有限性，这一点我也没有做进一步的处理。再者便是对错误字符的修正，对于这一点，我个人觉得是有一定意义的，但是由于没有找到合适的方法来对实现，所以未加入数据处理。



Implementation

根据课题要求，本次恶毒评论分类的实现，运行的库：keras，sklearn 等，实现脚本：Python3，硬件环境有两种，一种是 GPU(Floydhub – Standard GPU(Tesla k80 12GB Memory 61GB RAM 100GB SSD))，另一种是 CPU(Intel Xeon(R) CPU E3-1220 r3@3.1GHz, RAM 8G) . 整个实现过程将采用两个的基于预训练词向量的深度类型模型的建模以及一个的传统词袋模型建模.

大致的流程可以分为数据分析，数据处理，建模（模型训练，测试，模型评估，调参，训练，测试）以及模型融合。数据分析，将从数据的分布，多标签状态来进行基本分析。数据预处理部分，采用通常使用的文本数据的处理方式。比如特殊字符的去除，奇怪字符的去除，提取词干等。数据预处理第二部是需要对经过预处理的评论数据进行分词处理，比如利用 Tokenizer 将文本转换成一个序列的类，pad_sequences 进行序列填充。数据预处理第三部分，就是词嵌入。就是利用已准备好的预训练词向量，为深度学习的 embedding 层做准备。流程图如下所示。



创建模型根据两种不同的方式来创建，传统词袋模型主要是应用到 sklearn.linear_model 中 LogisticRegression 来创建，主要是反复训练调整参数 C，solver(优化算法)，让模型的逻辑损失尽可能的降低。主要流程如下图所示，试验过程中，分别用 analyzer 为 word 和 char 的

tfidfVectorizer 进行特征提取,LogicticRegression 分类，其中参数 C 从 0.1-0.5 调参，用 cross_val_score 交叉验证，其中参数 scoring 为 auc，验证测试过程及结果详见下一章 Results。

深度学习的模型，将会用到 Keras 的 GRU 层，首先对与训练词向量 fastext，glove 提取权重，然后将权重作为参数传入 embbeding 层，再结合 bigru，conv1d 以及 dropout，dense 层建立模型。该深度学习模型，主要调整参数有 dropout 的百分比，bigru 的 units，测试验证比，模型的 batchsize。具体的调参过程及结果见下一章 Results。

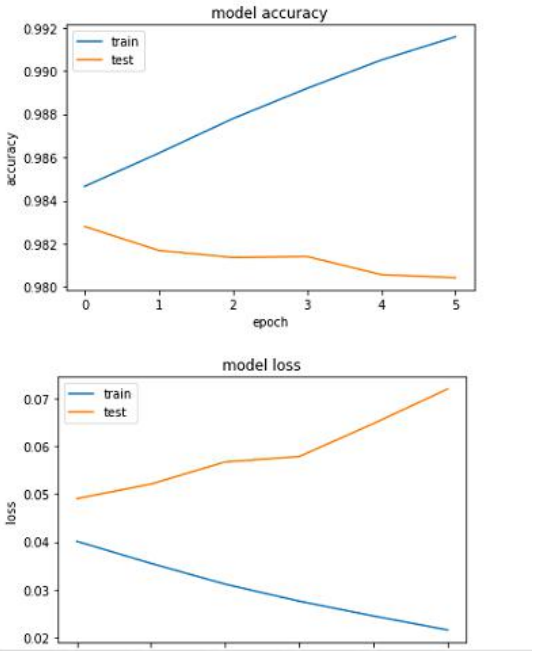
Results

Model Evaluation and Validation

1. 传统模式。TF-IDF analyzer = word，最大的 AUC score 是 0.978，对应的 tf-idf 参数是 n-grim=(1,1)，min-tf=10，max-tf=30000，LG 的参数 c=0.5，cv=5，测试过 bigrim trigrim，结果均没有 n-grim 为 1 的时候值要高。关于这个结论，个人觉得是跟预期有差别的，按照常理，n 约大，越能体现词义，但是从预测结果看，n 越小预测却越准确。Analyzer = char ,最大 AUC scroe 是 0.9804，tf-idf 参数是 n-grim=(2,5)，min-tf=50，max-tf=30000，LG 的参数 c=0.5，cv=5，两者组合后，最大 AUC score 是 0.9840。CV 的值从 5 改至 20 后，AUC score 上涨 3 个万分点，最后结果可以达到 0.9843。cv 测试过 30 的值，预测结果没有 cv 为 20 时的高。下图为 AUC score 达到 0.9845 的 submission.csv 截图，详见附件中 submission.csv 文件。

	id	toxic	severe_toxic	obscene	threat	insult	identity_hate
137113	e5254ce9fabd9893	1.0	0.972202	1.0	0.086464	0.999999	0.540957
39402	41600a159afb4a20	1.0	0.943378	1.0	0.070153	0.999989	0.406263
122641	cccdd198baa36464	1.0	0.956472	1.0	0.203682	0.999978	0.666133
126357	d320f8749c6d9820	1.0	0.971767	1.0	0.199575	0.999842	0.277631
141347	ec4774caac6adef4	1.0	0.937374	1.0	0.083501	0.999960	0.489629
135375	e248b2a56473f1a3	1.0	0.961801	1.0	0.592078	0.999964	0.900754
71810	77a740cd70e2959f	1.0	0.959117	1.0	0.058676	0.999693	0.145779
69838	74412dadacac9976	1.0	0.902598	1.0	0.117471	0.999799	0.285364
146815	f56c9f2f5c599ec2	1.0	0.988230	1.0	0.159440	0.999997	0.616608
72990	799638b93f93a0c4	1.0	0.990347	1.0	0.561841	0.999989	0.961152

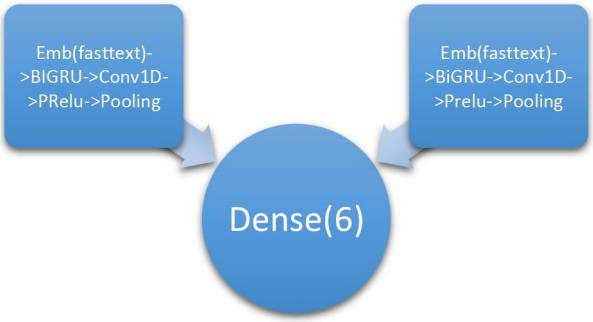
2. Deep learning. 模型一，采用深度学习模型，不引入词向量时，测试的结果如下图。模型过拟合。



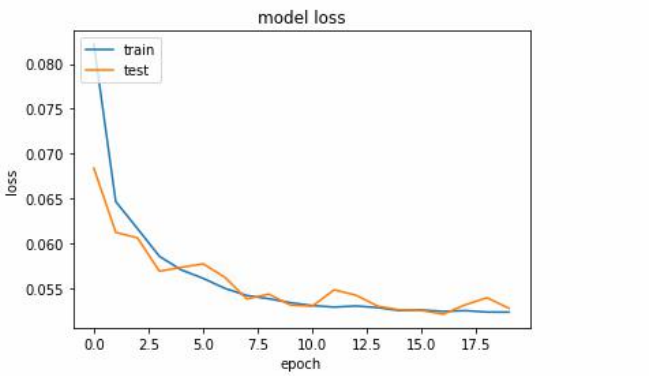
模型一结果

模型二，单层 Embedding(fasttext/glove)+dropout(0.5)+BiGRU+dropout(0.3)+dense(0.12),batchsize 为 35 左右，测试结果是 AUC 的值 0.984。

模型三，两个模型二，最后结合在一起，大致模型如下：



预测结果如下图所示，据下图显示的结果来看，有明显的收敛，且 test 和 train 的 loss 都随着 epoch 的增加在不断递减，故该模型没有过拟合，也没有欠拟合。

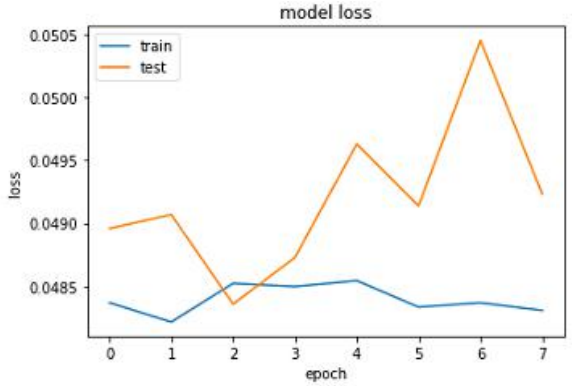


模型三结果

20 个 epoch 时，已有 3 次 loss 无法下降故而停止，batchsize 35，最终 AUC 0.9479.

模型三改进一，把模型其中一个 embbedding 层改用 glove 的权重，其他参数不变，最后的 auc 结果是 0.9463，没有模型三的 auc 值高，说明还是 fasttext 更加适合在这次的数据中使用。

模型三调参二，增大 batchsize 为 256，测试验证比改为 0.3，其他参数不变， 结果第一个 epoch 就可以达到 0.95 以上，但是 7 个 epoch 就无法再下降 loss(earlystop 设置的 5)，最后结果停在 0.9537，model loss 与 epoch 的关系图如下：

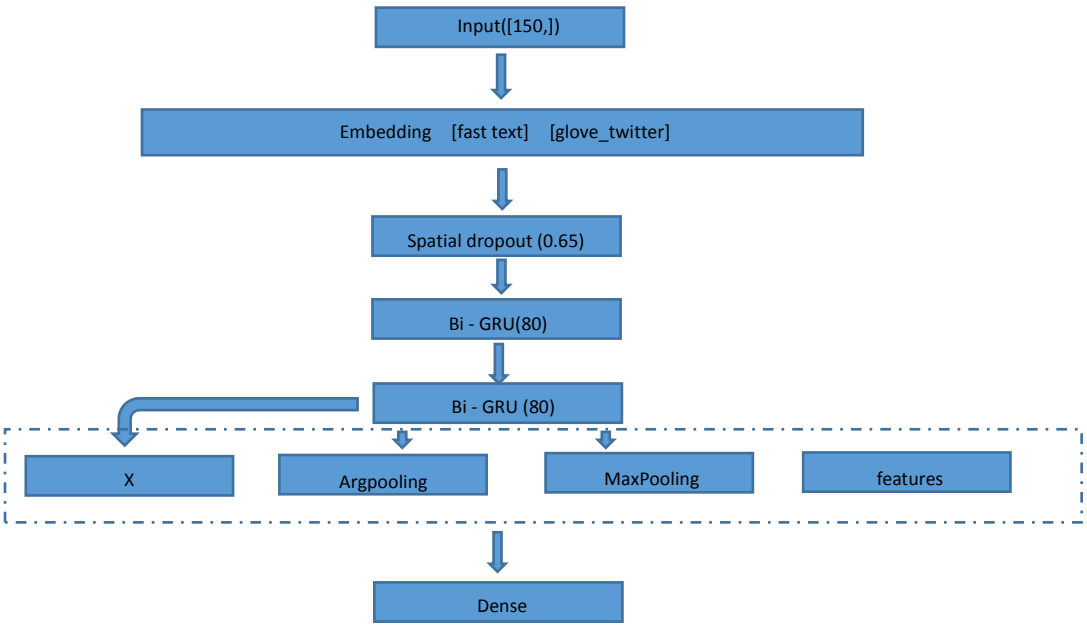


模型三结果二

虽然 auc 的值有提高到 0.9537，但是从训练的结果看，模型有点过拟合；

模型四，大致跟模型三类似，主要的改进有三点，其一在于 embedding 层的 weight-matrix 的区别。Weight_matrix 利用了通过 fasttext 和 glove-twitter 两个预训练词向量生成，而之前的模型都是单一的使用 fasttest 来提取 weight_matrix；其二，在于将 train 数据进行了 Kfold 交叉验证；其三，模型中用到 feature 是根据 test 提取的一些特征，'caps_vs_length', 'words_vs_unique'。具体模型如下图所示。运算过程可参见 log.txt，运算结果见 test_submission.csv。最终 AUC 的值为 0.98791.

```
In [52]: print('Total CV score is {}'.format(np.mean(scores)))
Total CV score is 0.9879136506450571
```



Conclusion

传统的词袋模型(tf-itf)方式建立的模型，最后的 AUC 的值为 0.9843，虽然结果不是特别的占优势，但是胜在运算量不大，运算时间不长。

深度学习算法，所建立的模型效果目前达到的 AUC 最大值是 0.9879，满足标准 top 20%，从 kaggle 里提供的一些 top 1%的方法来看，我有以下几点改进，或许能提高 auc 的值。

1. 数据增强，比如翻译，或者爬取更多的测试数据；对于翻译，我觉得有利有弊，毕竟是机器翻译，必然会引入一些误差，或许不合适的翻译，会导致模型的 AUC 的值不升反降；
2. 伪标签的半监督学习以及模型的融合。
3. 增大 max-features，maxlen，batchsize。

从目前我所试验的结果来看，深度学习模型从拟合的结果上来看是优于传统方式的，但是从测试时间来看，深度学习模型每次的试验时间都比较长，每次一个 epoch 都需要达到半小时的时间来运行(cpu 环境)，基本每个模型运行到 10 个 epoch 左右.所以就模型 4 一次的运算时间，大致需要 30 小时左右。如果硬件环境好，还是深度学习的优势比传统词袋模型更有优势。

Reference

[1]https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/rules

[2]https://en.wikipedia.org/wiki/Natural_language_processing#cite_note-choe:emnlp16-8.

[3]Goldberg， Yoav (2016). A Primer on Neural Network Models for Natural Language Processing. Journal of Artificial Intelligence Research 57 (2016) 345–420.

[4]https://mp.weixin.qq.com/s?__biz=MzI3ODgwODA2MA==&mid=2247485804&idx=1&sn=0c60

148b909fbdc7c2e7e4a27df0f71b&chksm=eb501dffdc2794e99b53fa5bf0cfd3217330d4bb0d925375f1878d6bf17f075dd2f9b84ae97c&mpshare=1&scene=1&srcid=1103Yni8GqzCXyICmZu43m0o&pass_ticket=nq841biAAM3IBsHnTR61koAmmG4sNPAKyla2u4YnvJOBX4H6XBYg5LhtF28pnSsP#rd
[5]Spiros V Georgakopoulos(Feb 2018). Convolutional Neural Networks for Toxic Comment Classification.
[6]Siyuan Li(2018). Application of Recurrent Neural Networks In Toxic Comment Classification.
[7]https://tracholar.github.io/machine-learning/2018/01/26/auc.html