# 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.

### 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);

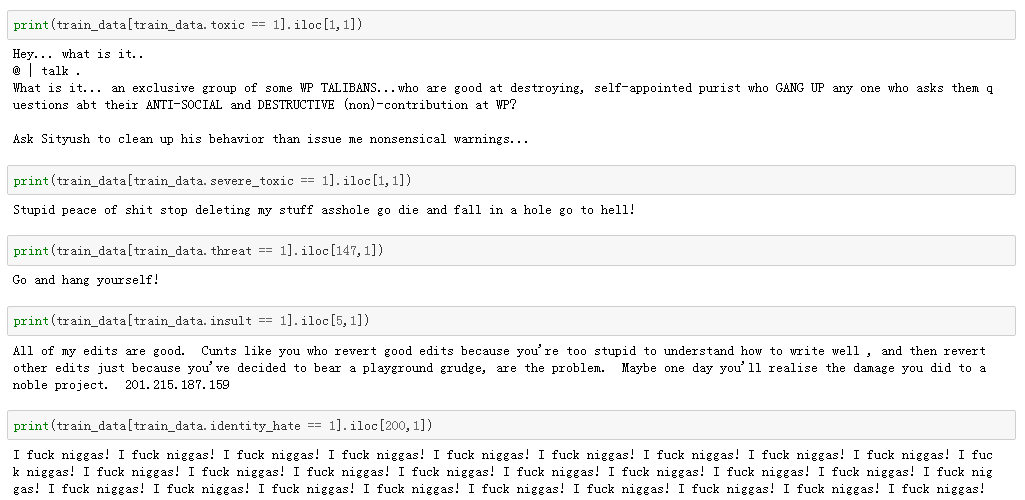
* 微元法 绘制ROC曲线,计算ROC曲线面积，即AUC的值.

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。



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。

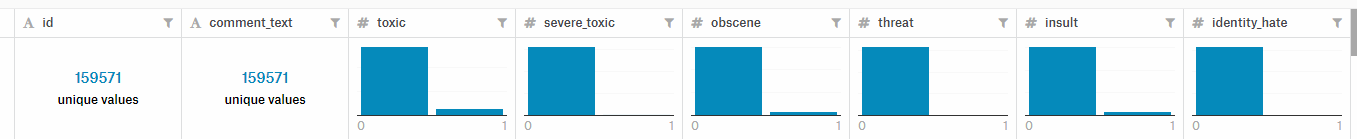


图2

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。

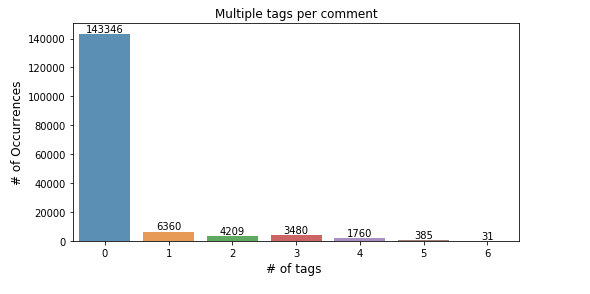


图3

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；

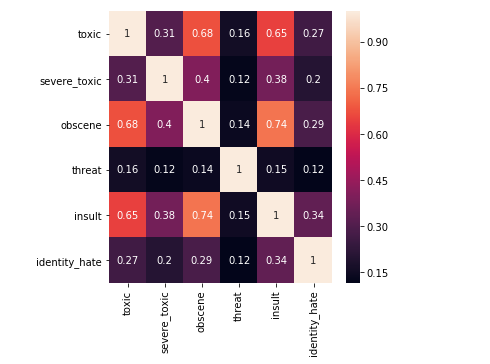


图4

1. 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。

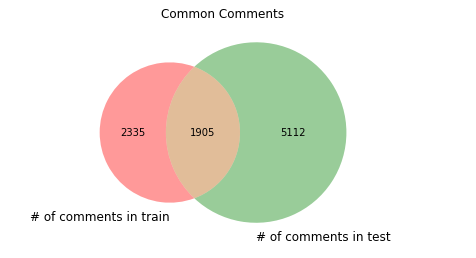


图5

### 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：



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：



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， 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；

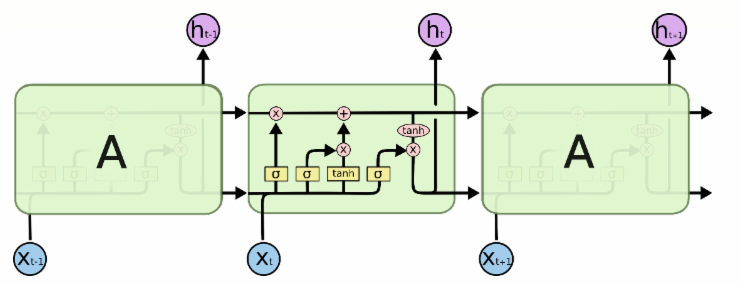
预测概率公式如下：



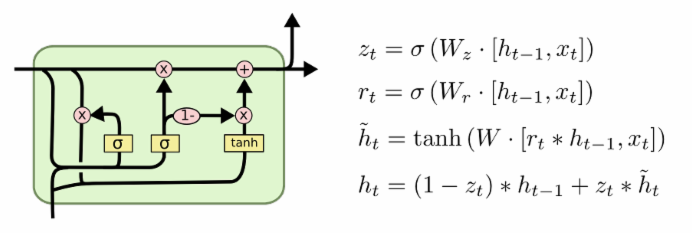
其中是需要计算的回归系数.该系数可以通过最大后验估计(MAP)来计算，也可以通过其他方法，例如梯度算法等来计算。

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.



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。

Train.csv

特殊字符去除

去重

分词处理

End

Test.csv

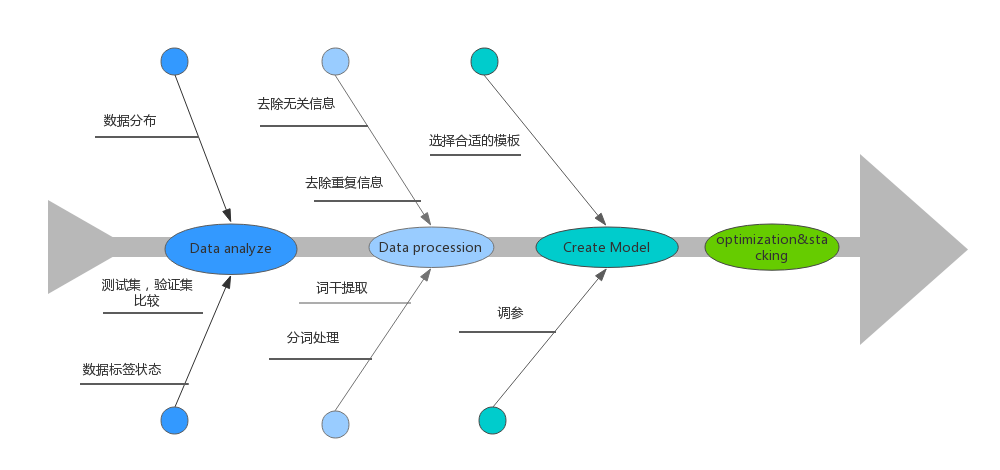
词干提取

### 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，硬件环境是 Floydhub – Standard GPU(Tesla k80 12GB Memory 61GB RAM 100GB SSD) . 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。接下来就是创建模型。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。

深度学习的模型，将会用到Keras的LSMT, GRU层，以及加入之前提到的预训练词向量fasttext、glove做embbeding层。经过反复的调参训练，计算AUC，以期达到模型的最优。在实际的测试过程中，考虑不使用预训练词向量，测试的结果过拟合，所以需要加入预训练词向量，流程图如下图所示。

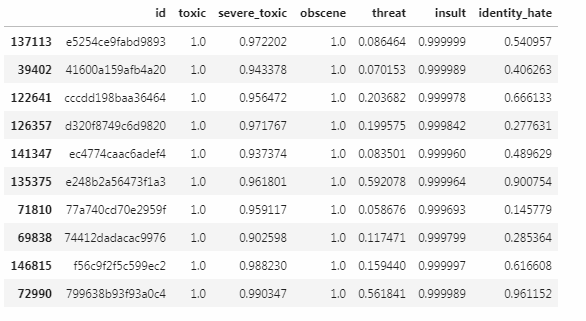


### Refinement

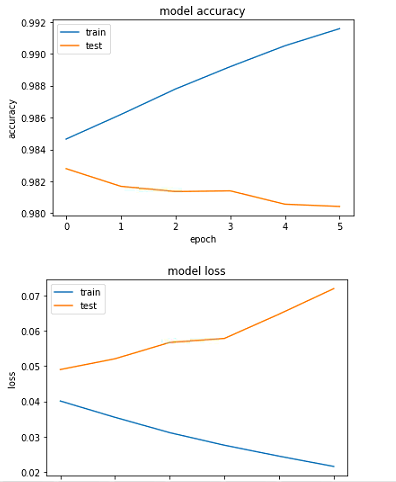
## Results

### 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5个万分点，最后结果可以达到0.9845。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。



1. 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。



1. 采用深度学习模型，引入词向量。第一次建立的模型就是单层Embedding(fasttext/glove)+dropout(0.5)+BiGRU+dropout(0.3)+dense AUC 0.984左右；第二次建立的模型是Embedding(fasttext/glove)+dropout(0.5)+BiGRU+dropout(0.3)+BiGRU+dropout(0.3)+dense ,epoch设为10，结果运行到第三次，连续两次val\_loss都没有降低，且每个epoch需要运行74min，val\_acc为0.9837。模型运行时间过长, 而运行效果并不尽如人意。第三次建模是利用了fasttest预训练词向量，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。20个epoch时，已有3次loss无法下降故而停止，batchsize 35，最终AUC 0.9479. 同时还试过一个embbedding glove，一个embbedding fasttext，最后的auc结果是0.9463，说明还是fasttext更加适合在这次的数据中使用。



## Conclusion

无论是传统的方式，还是利用了深度学习算法，我所建立的模型效果目前达到的AUC最大值是0.985，距离top20%还有一定的距离，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. 错字更正；
2. 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；
3. 深度学习模型，增大batchsize，由于运行环境的限制，我的batchsize达到50，memory就会溢出了，所以在深度学习这部分，我一直将batchsize设置在35左右测试，我也在floydhub云平台上测试过，batchsize越大，一个epoch的测试的时间越长，而结果也并没有非常明显的提升。
4. 伪标签的半监督学习以及模型的融合。

从目前我所试验的结果来看，深度学习模型略优于传统方式，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1小时的时间来运行，基本每个模型运行到20个epoch，loss就没有下降了。或许我模型建立的不够好，还需要进一步的学习和提升。

## 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=0c60148b909fbdc7c2e7e4a27df0f71b&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