机器学习纳米学位

毕业项目

Joe Huang 优达学城

2018年11月20日

I. 问题的定义

随着互联网与社交媒体的发展，恶毒评论在各大论坛网站层出不穷，如虎扑、Facebook等。对恶毒评论研究分类，可以对信息进行筛选，可以有效的清理网络环境。其中最基本的是对恶毒评论进行甄别，这里用到的主要技术是自然语言处理。

自然语言处理方向（简称NLP），是机器学习中十分热门的一个方向。据维基百科所述，NLP是计算机科学、信息工程和人工只能的重要领域，是计算机与人类语言交互的重要手段，它通常包括认知、理解和生成等部分。其中认知和理解是电脑将输入的语言变成特定的符号，生成则是将计算机数据转化成自然语言。

同其他机器学习领域类似，自然语言的起源也很早，早在上世纪50年代就被伟大计算机科学界艾伦-麦席森-图灵所提及，但却局限于当时的计算水平而无法真正意义上的普及。著名的“图灵测试”就是当时被提出，这是一项判断机器语言与自然语言的标准。2018年google IO大会上推出的google assistant，迄今最有可能通过该测试。

自然语言的处理范畴很多，包括文本朗读（Text to speech）/语音合成（Speech synthesis），语言识别（Speech recognition），语法分析（Parsing）、自然语言生成（Natural language generation）和文本分类（Text categorization）等等。较大众所熟知的多为语言识别类，而本毕业项目“恶毒评论分类”则属于文本分类范畴。

项目概述

项目名称: Toxic Comment Classification（恶毒评论分类）

恶毒评论项目源于Jigsaw(前身为Google ideas）在kaggle平台上举办的一场文本分类比赛[1]，旨在对于网络社区部分恶毒评论进行区分鉴别。

文本分类是自然语言中比较普遍的应用，如文件邮件自动分类等，目前主要有传统机器学习和深度学习模型方法等。常见的处理流程包括：训练样本预处理、特征选择、计算特征权重得到文本向量和模型训练与预测。

输入数据如下：kaggle中提供的数据包括4部分：train，test，sample_submission，test_lables。

train训练数据总量约160k，包含1个id行和7个标签数据，见下图。数据的分布为非均匀分布，并且每个数据可能对应多个标签，其中clean语句的分布最多，约为87%。toxic语句约为9.5%，obscence语句约为5.3%，insult语句约为4.9%，severe-toxic语句约为1%，identity_hate语句约为0.88%，treat语句最少，约为0.3%。

test和test_lable为测试数据及标签，总量约为150k。test数据只包含id，comment_text，output需要输出分类概率。

问题陈述

本项目需要解决的是自然语言文本分类问题，根绝训练集数据，包括toxic、obscence、severe-toxic、identity_hate和treat六个类型的文本语言，项目需要对其进行训练，以保持对测试集数据能够很好地分类，并输出概率，如下表所示。项目的重点难点在于合适的文本向量与合适的训练预测算法。

最终需要输出的标签概率，示例：

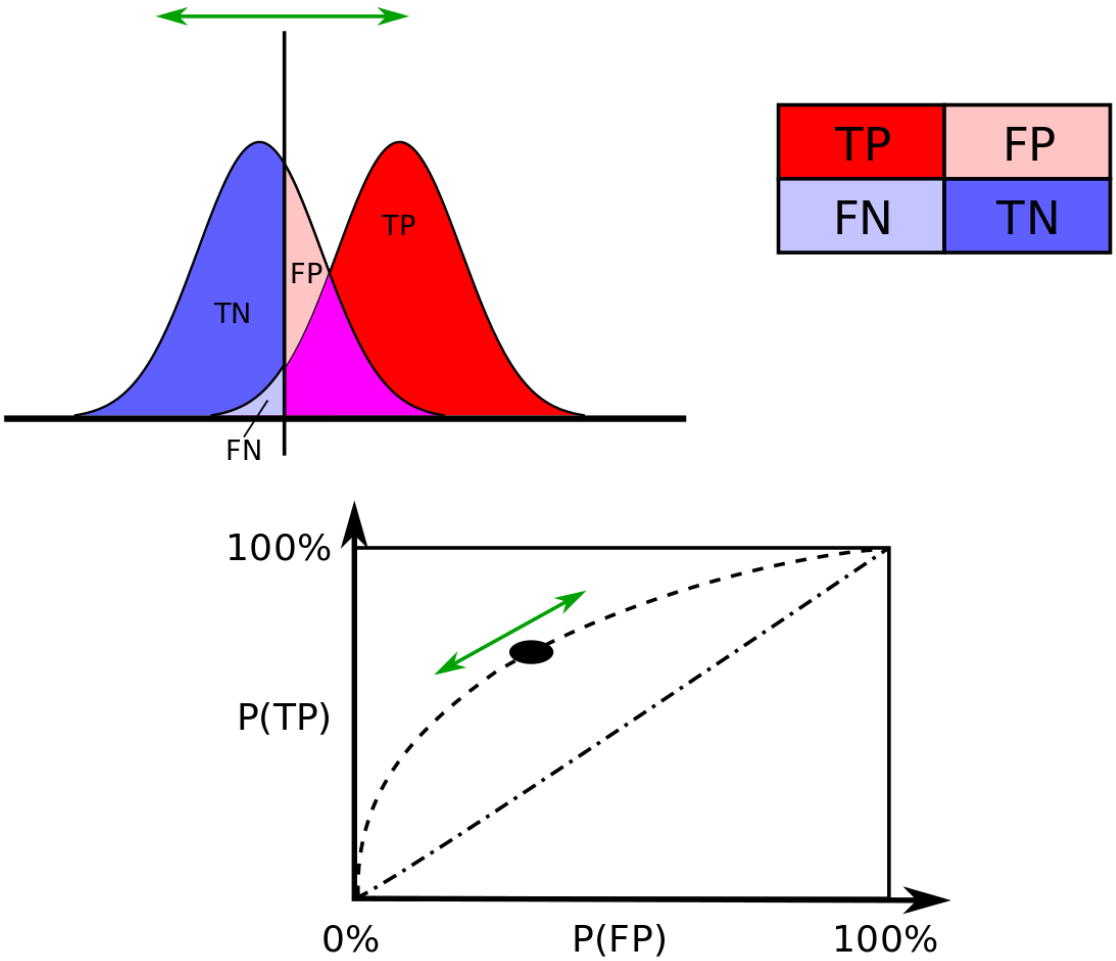
id	toxic	severe_toxic	obscene	threat	insult	identity_hate
00001cee341fdb12	0.5	0.5	0.5	0.5	0.5	0.5

评价指标

根据kaggle上的描述，采用ROC AUC评估矩阵方式，分数计算方式为每一个预测列的平均AUC值。

ROC曲线为True Positive Rate和False Positive Rate曲线图[8]。ROC曲线特性适用于当测试正负样本集变换时，ROC曲线能保持不变。对于实际数据中出现样本分类不平衡时，集正负样本比例比较大且随时间变化时，ROC曲线基本保持不变。AUC（Area Under Curve），即ROC曲线下方面积，介于0.1-1。AUC值越趋向1，分类器效果越好。

ROC-AUC概念图如下：



II. 分析

(大概 2-4 页)

数据的探索与可视化

根据kaggle提供的数据，下面展示train数据集前10行数据。

数据集包括id、comment_text、6个类别，0和1表示语句对应的标签值，一个语句可以对应有多个分类。若每个类别对应的标签值均为0，表示该语句为clean。显然，clean语句占大部分文本内容。

- train具体数据前10行。

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0

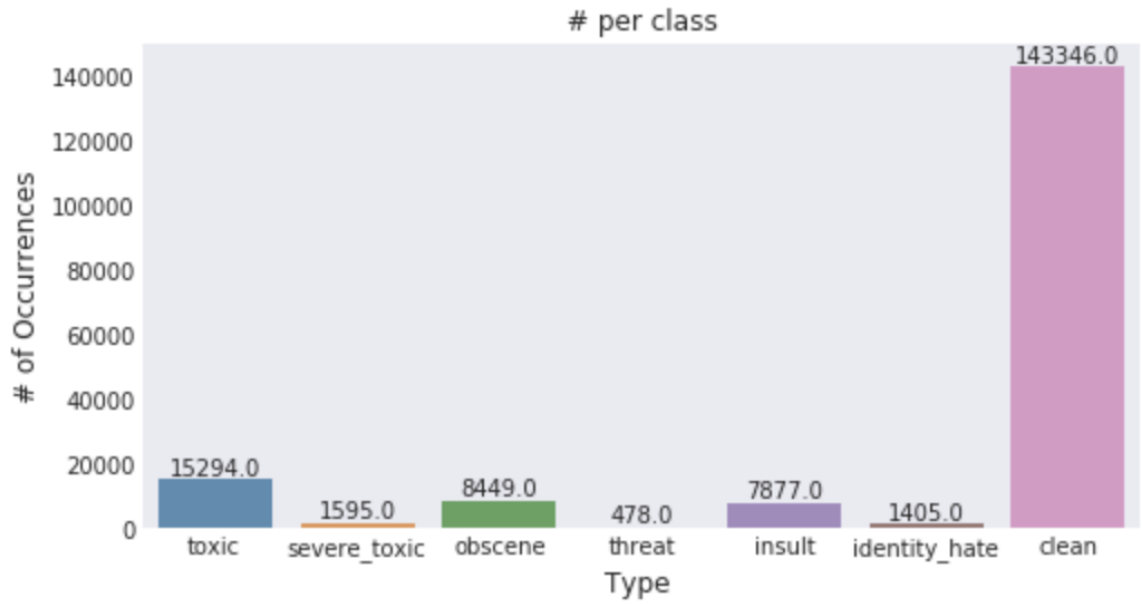
- train数据统计如下表所示：

train数据共计159571行语句，根据语句标签可以得到各个类别的概率统计值。

	toxic	severe_toxic	obscene	threat	insult	identity_hate
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

- train数据分图形展示：

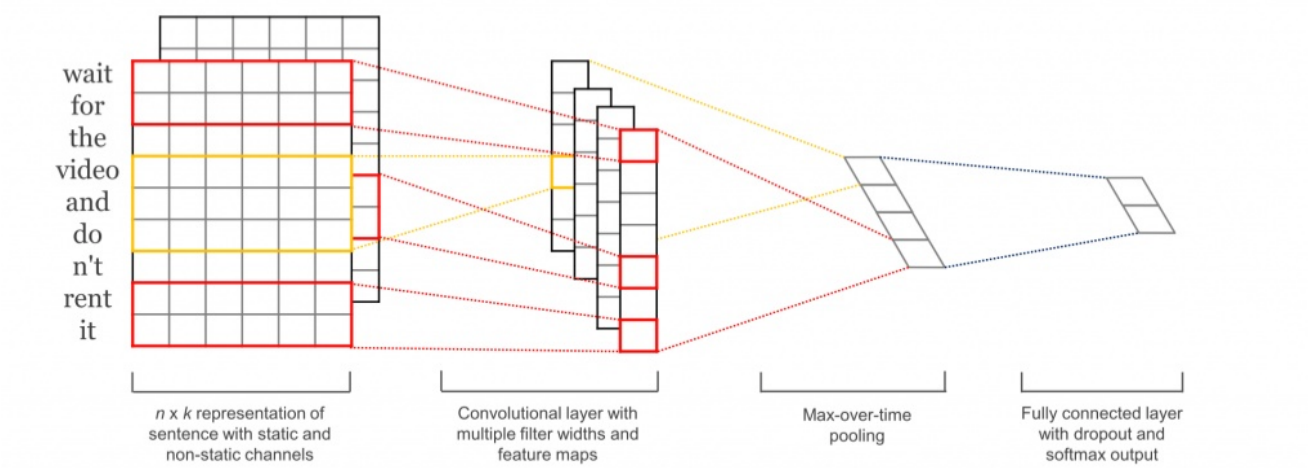
  从图中可以看出，clean语句远远大于其它语句，约为87%。toxic语句约为9.5%，obscence语句约为5.3%，insult语句约为4.9%，severe-toxic语句约为1%，identity_hate语句约为0.88%，treat语句最少，约为0.3%。



算法和技术

算法主要采用（CNN + GRU）× 3三个模型进行特征提取。本项目将主要基于keras上的框架进行分析。

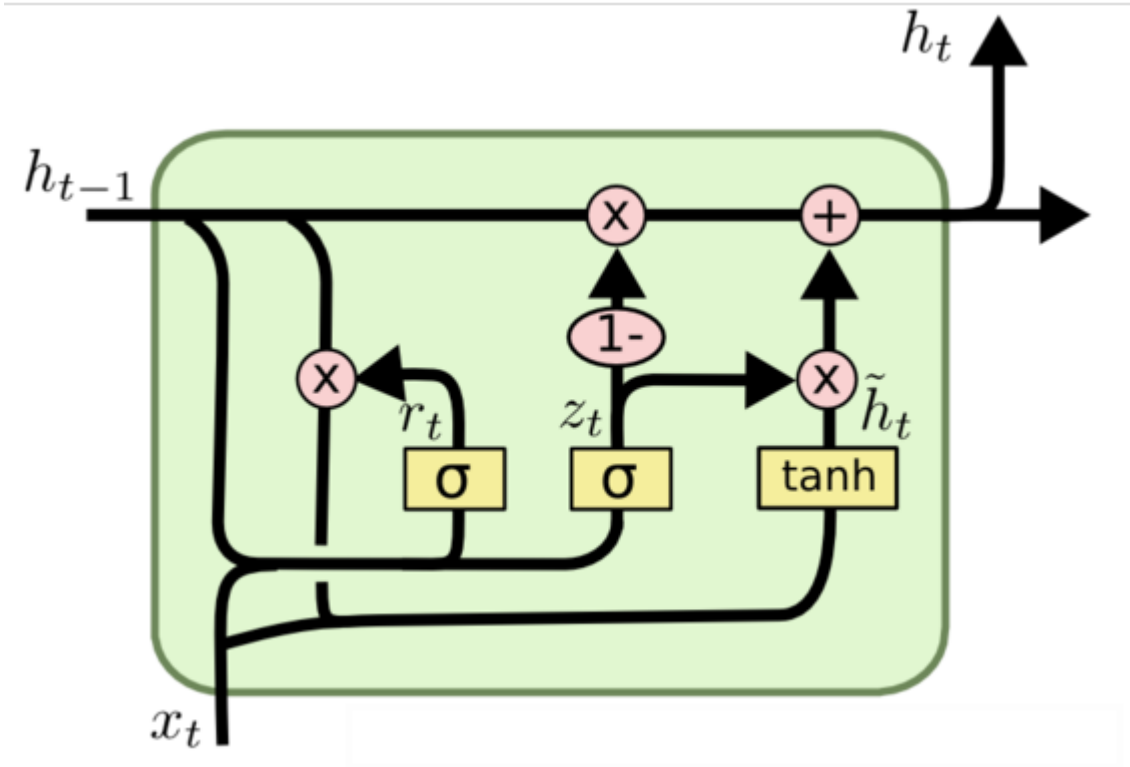
CNN，卷积神经网络。主要基于卷积运算，这是信号处理中十分常用的方式，可以对特征进行提取。在自然语言处理方面，它的优势在于能快速进行计算，在表征方面也更加有效[5]。在NLP自然语言文本分类，对于本项目包含情感的分类比较有效。



GRU即Gated Recurrent Unit，门限循环单元网络 。它是LSTM的变体，在保持LSTM效果的同时，其结果更加简单。通常认为，GRU具有较少的参数，所以训练速度快，而且所需要的样本也比较少。而LSTM具有较多的参数，比较适合具有大量样本的情况，可能会获得较优的模型。

GRU模型如下，它只有两个门了，分别为更新门和重置门，即图中的zt和rt。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。

GRU模型图示：



在算法训练完成后，需要对算法模型进行评估。应用算法对测试数据进行预测，并根据上文提及的ROC-AUC评估矩阵，对预测值进行评估。

基准模型

自然语言算法模型分类大致分为传统方法和深度学习神经网络方法。传统方法主要是从原始文档提取特征，在制定分类器进行计算。经典特征提取方法如频次法、tf-idf、互信息法、Ngram，分类器算法如LR、SVM等。神经网络特征提取方法如CNN、RNN、LSTM等。基于传统方法tf-idf和LR算法已经可以很好地得到分类结果[6]，本项目将其作为benchmark model。

tf-idf为词频-逆文档频度（Term Frequency - Inverse Document Frequency，TF-IDF）。词频为语句在某一个给定的词语在该文件中出现的次数。逆文档频率是值出现给定词语的文件总数概率对数。该概念能够很好反应词语的重要性。

LR为逻辑回归，是传统机器学习的方式。

该model处理方法比较简单，先是提取数据中的文本资料，然后对统计tf-idf，将其转化成词词向量。最后用机器学习的方法对test数据集进行预测。相比与深度学习，该方法更加简单高效，并且能得到很高的预测分数，在一定条件下实用性高。

该模型有很好的效果，在kaggle上的得分约为0.97+.

III. 方法

(大概 3-5 页)

数据预处理

数据挖掘分析，主要是采取可视化的方式对训练测试数据进行展示，如数据量大小、toxic comments与non-toxic comments的分布等。可以借助大数据词云显示等，分别展示toxic comments与non-toxic comments的较大词频词语的分布等，有助于本人了解不同分类语句之间的主要区别，也有助于下一步数据处理。

数据处理是主要包括数据预处理与主处理。

数据预处理主要是对数据进行清洗，去除一些训练测试数据中出现的空白、乱码词句、杂乱无章等非正常语句。

数据主处理主要是对语句进行处理，计算机无法识别自然语言，必须将其转换成机器语言。本项目中，需要将语句解析分成单独词字，再将词字转化成数值形式，并进行编码。这种形式称为word embedding[4]，常见手段有Glove、fastText、word2vec。杜热编码（One-Hot encoding）也是一种方式，但对任意词的余弦相似度都为0，但无法表达不同词之间的相似度。

对于自然语言处理而言，预训练词向量特别关键。本项目将采用glove.840B.300d作为预向量数据。

数据清洗

项目中处理的主要是文本内容中的评论恶毒情况，数字基本作用不大。

去除一些没有意义的字符，有'.'等。

利用NLTK模型中的stopwords，去除英文中的停止词。

```
all_text.replace({r'[^\x00-\x7F]+':''},regex=True,inplace=True)
text_raw = re.sub(r'[^a-zA-Z]', ' ', text_raw)
# 去停止词
stops = set(stopwords.words('english'))
result_text = []
result_text = " ".join([w for w in words2 if not w in stops])
```

数据清洗前后结果如下：

对比显示，有用的信息能够被基本保留。

Text[0] before cleaned: explanation why the edits made under my username hardcore metallica fan were reverted? they weren't vandalisms, just closure on some gas after i voted at new york dolls fac. and please don't remove the template from the talk page since i'm retired now.89.205.38.27

Text[0] after cleaned: explanation edits make username hardcore metallica fan revert vandalisms closure gas vote new york doll fac please remove template talk page since retire

分词处理

利用keras的Tokenizer模型，对目标文本进行分词处理处理，主要参数包括：

MAX_NUM_WORDS = 100000 MAX_SEQUENCE_LENGTH = 200 EMBEDDING_DIM = 300

该部分主要基于glove预训练数据，生成Embedd 矩阵和Embedded Layer。

词向量的原理是是将一类将词的语义映射到向量空间中去的自然语言处理技术。即将一个词用特定的向量来表示，向量之间的距离（例如，任意两个向量之间的L2范式距离或更常用的余弦距离）一定程度上表征了的词之间的语义关系。由这些向量形成的几何空间被称为一个嵌入空间。

本项目将采用glove.840B.300d作为预训练，glove数据中每行为一个单词和其对应的词向量，以空格分隔。使用词向量的目的是进行词的向量化表示，使得向量之间尽可能多地蕴含语义和语法的信息。

Glove模型如下：

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

执行过程与完善

本项目利用AWS GPU进行计算，jupyter notebook工具完成算法。

本项目算法模型框架简介如下：

1. Input层，输入维度为200，同Tokenizer分词时的最大维度保持一致。

   sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype = 'int32')

2. 本层为Embedded layer， Embedded layer由词向量组成，接收上一层Input输入。初始化由Glove提供的840B.300d，输出维度为300.

   embedded_sequence = embedding_layer(sequence_input)

3. 本层为dropout层，以降低模型的过拟合程度。

   x = SpatialDropout1D(0.2)(embedded_sequence)

4. 本层为Bidirectional包装的GRU层，输出维度为128。

   x = Bidirectional(GRU(128, return_sequences = True, unroll = True))(x)

   上一层结束后，开始将上一层输出部分构建三层并联的模型，每一路模型由一维CNN、Dropout层、池化层和GRU模型组成。

5. Conv1D为一维卷积神经网络模型，卷积输出维度128，卷积核为1， 权值初始化为normal。

   conv_0 = Conv1D(128, 1, kernel_initializer='normal', activation = 'relu')(x)

6. Dropout依次为0.4， 0.45， 0.5，用以降低过拟合。

   drop_0= Dropout(0.4)(conv_0)

7. 池化层用于对信号进行最大值池化，其窗口大小为4.

   max_pool0= MaxPooling1D(pool_size = 4)(drop_0)

8. GRU输出维度为100.

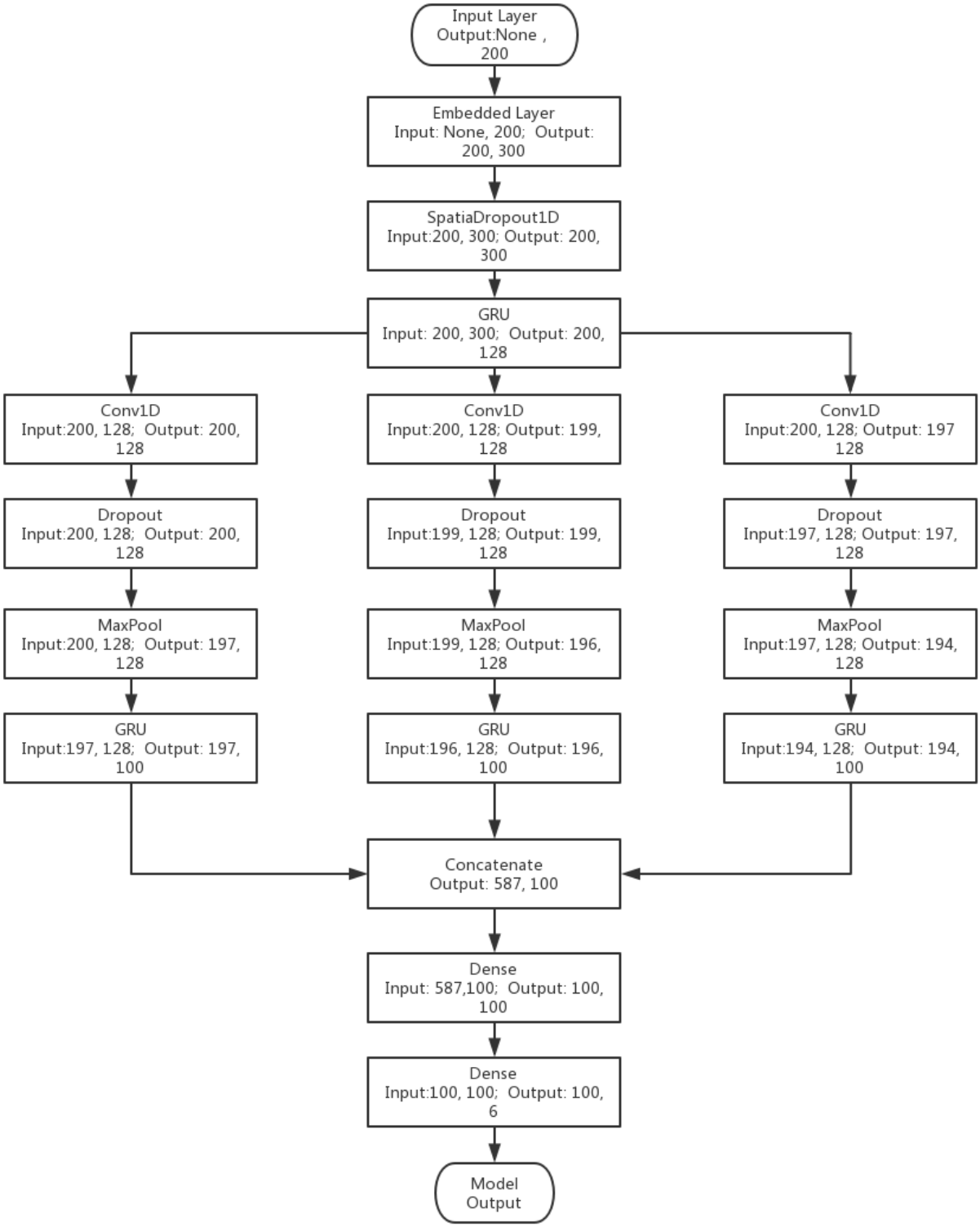
   gru_0= GRU(100, dropout=0.2, recurrent_dropout=0.2)(max_pool0)

9. 最后将三路并联模型融合，并采用Dense层，压缩成对应6个分类标签。

   conv_sum = concatenate([gru_0, gru_1, gru_2], axis = 1) # 模型融合

   dense1 = Dense(100, activation='relu')(conv_sum) # 压缩成对应6个标签 preds = Dense(6, activation = "sigmoid")(dense1)

本项目模型流程如下：



参数调整：

- Conv 1D filter filter size参数表征CNN输出的维度，本项目选择了32，64,128三个值分别进行计算，得出的结果分别为：0.9773， 0.9819，0.9832。随着filter size的值逐渐增大，计算得分逐渐升高，因此选择128作为最后维度。
- epoch 根据提供的benchmark model等参考模型，通常epoch取值为2~3。但在计算发现，随着epoch的增大，计算得分提升较大。但通常超过5后，计算得分会降低。且随着epoch的增大，运算时间也会倍数上升。因此最终选择epoch = 5.

- Dropout Dropout值主要是防止过拟合，但值较小时，不能有效的降低过拟合程度，当值较大时，容易影响样本本身的总量，导致欠拟合。因此在并联的三个模型中，我依次选择0.4， 0.45， 0.5作为dropout取值。
- batch_size 通过计算发现，随着batch_size的增大，并不能有效的优化fit效果，且发现batch_size 对模型优化的作用并不明显。因此选择适中值。

项目中对其它参数也有进行调整，这里主要是阐述影响较大的部分，影响相对较小的参数没有一一记录。

IV. 结果

（大概 2-3 页）

模型的评价与验证

- 线下得分

根据kaggle的评价要求，需要计算ROC-AUC值。本项目将训练集按0.85:0.15的比例，划分为训练集和验证集。根据算法模型，计算出验证集的ROC-AUC得分如下：

训练epoch设置为5，经过试验，epoch设置并非越高越好，基本超过5训练分数开始下降。

本项目的验证集最高得分为0.987左右。

```
Train on 135635 samples, validate on 23936 samples
Epoch 1/5
 - 399s - loss: 0.0463 - acc: 0.9821 - val_loss: 0.0430 - val_acc: 0.9836

 ROC-AUC - epoch: 1 - score: 0.984933
Epoch 2/5
 - 398s - loss: 0.0424 - acc: 0.9834 - val_loss: 0.0428 - val_acc: 0.9833

 ROC-AUC - epoch: 2 - score: 0.986788
Epoch 3/5
 - 397s - loss: 0.0402 - acc: 0.9840 - val_loss: 0.0415 - val_acc: 0.9838

 ROC-AUC - epoch: 3 - score: 0.987277
Epoch 4/5
 - 397s - loss: 0.0383 - acc: 0.9847 - val_loss: 0.0401 - val_acc: 0.9838

 ROC-AUC - epoch: 4 - score: 0.987849
Epoch 5/5
 - 396s - loss: 0.0363 - acc: 0.9853 - val_loss: 0.0415 - val_acc: 0.9835

 ROC-AUC - epoch: 5 - score: 0.987520
```

- 线上得分

根据算法模型，可以计算出test数据集的预测值，其提交至kaggle上的分数如下：

由于test数据集与验证集数据量大小等有差别，提交至kaggle上的得分较线下测试低，但也高于benchmark模型得分(0.97)。由此说明模型合理。

合理性分析

根据实际得分可以看出，最终模型得分超过基准模型得分，相对更加出色。

另外在本模型中，加入了对数据的清洗步骤，能在一定程度上降低过拟合，因此模型的实用性和试用下更强。

而在实际观察submission文件，输出的概率情况也能很好反映模型的正确性。

test数据某文本语句：

id	sentence
0017d4d47894af05	:Fuck off, you anti-semitic xxx.

test预测输出概率：

id	toxic	severe_toxic	obscene	threat	insult	identity_hate
0017d4d47894af05	0.9997698665	0.5473092794	0.9968622923	0.0027076399	0.9488716722	0.6624474525

V. 项目结论

（大概 1-2 页）

结果可视化

选择submission前10个输出结果展示如下表，最终提交结果由概率形式表示：

toxic	severe_toxic	obscene	threat	insult	identity_hate
0.997295	0.329954	0.956259	0.097659	0.873318	0.646170
0.004758	0.000019	0.000165	0.000068	0.000244	0.000055
0.001637	0.000025	0.000056	0.000035	0.000078	0.000032
0.000263	0.000004	0.000004	0.000009	0.000005	0.000002
0.011265	0.000053	0.000364	0.000140	0.000811	0.000130
0.000978	0.000005	0.000015	0.000027	0.000028	0.000020
0.004427	0.000007	0.000080	0.000031	0.000182	0.000034
0.583627	0.002034	0.017909	0.003164	0.174073	0.007318
0.341595	0.000226	0.003430	0.000644	0.082057	0.001456
0.000991	0.000006	0.000011	0.000018	0.000025	0.000013

对项目的思考

自然语言文本分类是自然语言中比较普遍的应用，如文件邮件自动分类等，目前主要有传统机器学习和深度学习模型方法等。常见的处理流程包括：训练样本预处理、特征选择、计算特征权重得到文本向量和模型训练与预测。

- 本项目的主要流程为：

读取数据，主要应用numpy，pandas等工具。

数据预处理，主要是利用正则表达式去除一些不必要的字符及停止词等。

数据分词数据，利用keras等工具，将自然语言转化为词向量。

模型构建：机器学习或深度学习算法等应用。

模型评估：利用Evaluation Matrics等方法评估模型得分。

- 比较有意思的地方：

模型构建是比较有意思的地方，在自然语言处理这块，可以采用多种算法来实现自己的目的。

- 比较困难的地方：

模型构建是有趣却有事困难的地方，要想深入了解模型的具体框架已经比较困难，而在此基础上建造自己的轮子，挑战十分巨大。

特征工程，是指从数据中提取并筛选特征用于后续机器学习模型的过程，特征工程的好坏直接影响到最终的输出结果。但特征工程是需要长期积累的，需要在项目中不断提升。

- 项目总结与评价：

利用机器学习的方法能很好地判断网络语言是否为恶毒评论，能有效地净化环境，但提高运算速度也十分必要。

需要作出的改进

- 本项目中主要采用CNN+GRU方法。在自然语言处理算法中，还有很多优秀的算法，如LSTM，fastText，以及传统的机器学习算法SVM等。各个算法应用的优劣势，还需要以后不断探索。
- 在本项目中数据清洗这块，只是简单了进行了处理。相关文献中采用正则表达式等进行处理，可以达到很好的效果。而某些模型并不对数据清洗，也能有不错的效果。之后还需要深入研究。

参考

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

[2]. https://blog.csdn.net/Heloiselt/article/details/80870794

[3]. XinRong, word2vec Parameter Learning Explained, 2016.

[4]. http://wiki.jikexueyuan.com/project/deep-learning/word-vector.html

[5]. https://jizhi.im/blog/post/understanding-convolutional-neural-networks-for-nlp

[6]. https://www.kaggle.com/tunguz/logistic-regression-with-words-and-char-n-grams

[7]. https://www.kaggle.com/yekenot/textcnn-2d-convolution

[8]. https://en.wikipedia.org/wiki/Receiver_operating_characteristic

[9]. https://keras-cn.readthedocs.io/en/latest/layers

[10]. https://zhuanlan.zhihu.com/p/28087321

[11]. https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/