自然语言处理技术课程大作业

2019

孟祥宇 201692322

cmcc.coder@gmail.com

张金哲 201692117

own_lay_4_u@163.com

一、 比赛简介及规则

1.比赛背景

此次比赛意在对各种网络评论的危害性进行检测，构建一个适用于广泛会话的模型。该模型应该具有以下性质:

a) 该模型可以准确识别出有危害性的评论词并根据其危害性大小输出相应的权重。相应的方法包括但不限于关键词检测（句子的划分和词的统计），词义的消歧,等等。

b) 该模型应该使与身份相关的无意识偏见最小化。例如，“我是同性恋”这种实际上并没有恶意的语句可能会因为经常在危害言论中出现的“同性恋“一词而被误判为危害言论。今年的 Kaggle 比赛特意强调了这一点。这就要求选手在语言处理的过程中将语句与整个语言环境相结合进行处理。

2.比赛规则及评价方法

本次竞赛将使用一种新开发的度量标准，它结合了几个子度量标准来平衡总体性能和各种意外偏差。

首先，我们将定义每个子度量。

（*注：AUC 为 ROC 曲线下与坐标轴围成的面积，通常用来评测分类器的效果好坏）

Overall AUC（整体 AUC）：完整评估集的 ROC-AUC。

Bias AUC：为了测量非预期偏差，我们再次针对每个恒等式的测试集的三个特定子集计算了 ROC-AUC，每个子集都捕获了非预期偏差的不同方面。

1) Subgroup AUC:在这里，我们将数据集限制为仅涉及特定标识子组的示例。此度量值较低意味着模型在区分提到身份的有危害性和无危害性注释方面做得很差。

2) BPSN(提及身份的无危害性示例和不提及身份的有危害性示例)AUC:在这里，我们将测试集限制为提到身份的无危害性示例和没有提到身份的有危害性示例。在这个度量中值较低意味着模型混淆了提及标识的无危害性示例和不提及标识的有危害性示例，这可能意味着模型预测的危害性评分要高于提及标识的无危害性示例。

3) BNSP (提及身份的有危害性示例和不提及身份的无危害性示例)AUC:在这里，我们将测试集限制为提及身份的有危害性示例和不提及身份的无危害性示例。这里的低值意味着该模型混淆了提及标识的有危害性示例和不提及标识的无危害性示例，这可能意味着该模型预测的危害性评分低于提及标识的有危害性示例的应有水平。

偏置 AUCs 的广义平均值:

为了将单位偏置 AUCs 合并为一个整体测度，我们计算其广义均值，定义如下:

$$M_p(m_s) = \left(\frac{1}{N}\sum_{s=1}^{N} m_s^p\right)^{\frac{1}{p}}$$

其中:
Mp =第 p 次幂平均函数
ms =计算子组 s 的偏置度量 m
N =恒等子群的个数

在本次比赛中,我们使用 p 值为-5 来鼓励竞争对手对模型性能最低的标识子组改进模型
最终的指标:　将总体 auc 与偏执 AUCs 的广义均值相结合,计算最终的模型得分。

3.所用数据集格式

本次比赛的文件描述:
Train.csv - 训练集，包含子组
Test.csv - 测试集，它不包含子组
Sample_submission.csv - 格式正确的示例提交文件

数据中的几个附加的危害性子类型属性:
severe_toxicity
obscene
threat
insult
identity_attack
sexual_explicit

本次比赛的各种标识属性:

male	female
Transgender	Other_gender
Heterosexual	Homosexual_gay_or_lesbian
Bisexual	Other_sexual_orientation
Christian	Jewish
Muslim	Hindu
Buddhist	Atheist
Other_religion	Black
White	Asian
Latino	Other_race_or_ethnicity
Physical_disability	Intellectual_or_learning_disability
Psychiatric_or_mental_illness	Other_disability

二、　摘要

危害性评论是指评价人以故意伤害被评价人为目的，利用本人、他人的名义对其他人或组织进行的恶意评价。通常情况下，危害性评论是发言人的一种情绪宣泄。然而，这种对他人进行的语言攻击，破坏了人与人之间和谐共处,更是对社会文明的一种蹂躏,伤害了别人也伤害了自己。不仅如此，危害性评论已经列入了法律之中。中国《刑法》第二百四十六条：以暴力或者其他方法公然侮辱他人或者捏造事实诽谤他人，情节严重的，处三年以下有期徒刑、拘役、管制或者剥夺政治权利。《民法通则》中提到：利用互联网辱骂他人，也同样应依法承担民事责任。

危害性评论对人与社会造成了严重破坏。然而，利用现有的自然语言处理技术，检测有害的网络评论已经成为可能。论文 Advances in Pre-Training Distributed Word Representations（T. Mikolov 等人）展示了如何通过结合使用一些已知的技巧来训练高质量的单词向量表示，其中有一些技巧（如关键词分割）同样适用于我们危害性评论检测的项目。

本文的研究目的是对各种网络评论的危害性进行检测，构建一个适用于广泛会话的模型，并输出这句话的危害性指数。在处理数据的过程中，我们用到了很多技术，其中有两个难点，一是基于语料库构建词的共现矩阵。共现矩阵通过统计一个事先指定大小的窗口内的 word 共现次数，以 word 周边的共现词的次数做为当前 word 的 vector。这种做法在一定程度上缓解了 one-hot 向量相似度为 0 的问题。二是分词器 Tokenizer 的技术，在之后的方法中会有所介绍。

本项目通过在 kaggle 官网上测试运行并且提交，最终的准确率达到了 88.0%，与普通人对评论是否为危害性评论的判别能力基本一致。故可以认为本项目训练出来的模型在一定意义上是一个成功的模型。

三、 Abstract

Harmfulness comment refers to the malicious comment made by the evaluator on others or organizations in the name of himself or others for the purpose of deliberately hurting the evaluated. Usually, a harmful comment is an emotional catharsis from the speaker. However, this kind of verbal attack on others destroys the harmonious coexistence between people, and it is also a ravaging of social civilization, hurting others as well as ourselves. Not only that, but hazardous comments are already in the law. Article 246 of the criminal law of the People's Republic of China: whoever, by violence or other means, publicly insults another person or fabricates facts to defame another person, if the circumstances are serious, shall be sentenced to fixed-term imprisonment of not more than three years, criminal detention, public surveillance or deprivation of political rights. As mentioned in the general principles of the civil law, those who abuse others through the Internet should also bear civil liabilities according to law.

Harmful comments cause serious damage to people and society. However, it is possible to detect harmful online comments using existing natural language processing techniques. Paper Advances in the Pre - Training Distributed Word Representations (t. Mikolov etc.) by using a combination shows how to use some skills to train high quality known Word vector, said some of the skills (such as keyword segmentation) also apply to our harmfulness review testing project.

The purpose of this study is to detect the harmfulness of various network comments, build a model applicable to a wide range of conversations, and output the harmfulness index of this sentence. In the process of data processing, we use many techniques, among which there are two difficulties. One is the construction of co-occurrence matrix of words based on corpus. Co-occurrence matrix counts the number of word co-occurrence in a window of a pre-specified size, and takes the number of word co-occurrence around word as the vector of the current word. This approach alleviates the problem that the similarity of one-hot vector is 0 to some extent. The second is the technique of Tokenizer, which will be introduced in later methods.

The project was tested and submitted on kaggle's official website, and the final accuracy reached 88.0%, which was basically consistent with the ability of ordinary people to judge whether a comment is harmful or not. Therefore, it can be considered that the model trained in this project is a successful model in a certain sense.

四、 引言

这是卷积神经网络的一个基本内核。在这个内核中，我们训练了一个关于 k-fold 的 CNN 模型，并在 kaggle 测试集上获得了不错的效果。

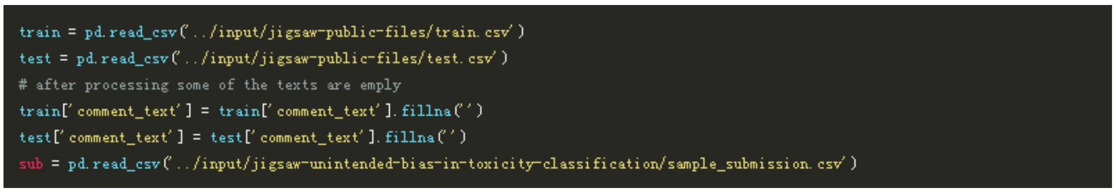
接下来我们将介绍整个方法。我们的方法分为六个步骤：数据读取，数据清洗，k-fold，引入预训练的词向量，模型和训练，预测数据方法。我们使用 python3.6 的环境和 keras 深度学习框架，核心方法是 CNN（即卷积神经网络），处理过程中也用到了和 k-fold，分类器 Tokenizer 以及基于语料库构建词的共现矩阵等方法。

我们的 kaggle 比赛结果为 88.0%，比赛结果图片会在之后进行详细展示。

五、 方法

a) 数据读取与清洗

使用 pd.read_csv 读取 csv 文件为 dataframe 格式数据集



```
train = pd.read_csv('../input/jigsaw-public-files/train.csv')
test = pd.read_csv('../input/jigsaw-public-files/test.csv')
# after processing some of the texts are emply
train['comment_text'] = train['comment_text'].fillna('')
test['comment_text'] = test['comment_text'].fillna('')
sub = pd.read_csv('../input/jigsaw-unintended-bias-in-toxicity-classification/sample_submission.csv')
```

由于输入的文本不仅带有字母,还带有符号,而文本中的符号大多带来的信息量很小。将符号分为两类:

- 标点符号类：如 ‘，’ ‘.’ ，将标点符号类的字符删除

例如：将 I love you, Merry. This is what I want to say.

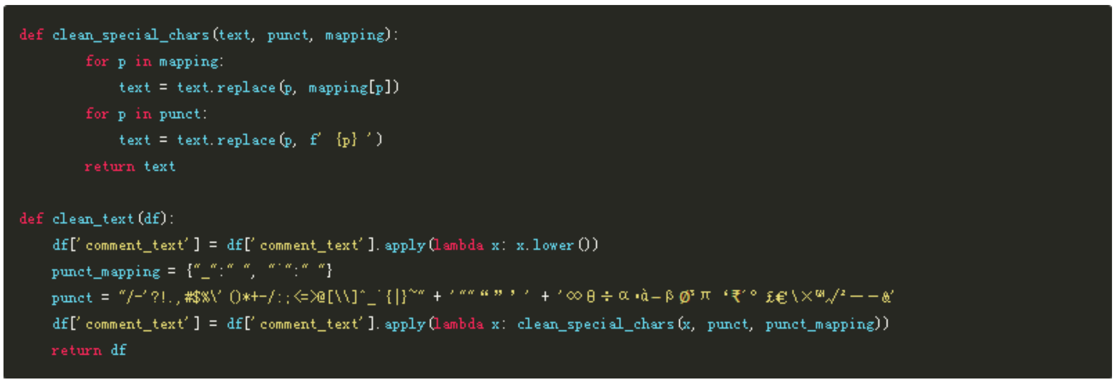
改为 I love you Merry This is what I want to say

- 分隔符类：如 ‘_’ ‘’’ ,这类符号的左面和右面通常是整个有意义的单词，我们将这一类的字符看作空格，即：处理时将它们替换为空格。这样替换之后相当于把两个有意义的词分隔开了。

例如：将 comment_text

改为 comment text

此外,鉴于大写字母和小写字母在通常意义下表达的意思相同，处理时将所有的字母都改为小写。



```
def clean_special_chars(text, punct, mapping):
        for p in mapping:
            text = text.replace(p, mapping[p])
        for p in punct:
            text = text.replace(p, f' {p} ')
        return text

def clean_text(df):
    df['comment_text'] = df['comment_text'].apply(lambda x: x.lower())
    punct_mapping = {"_":" ", "`":" "}
    punct = "/-'?!.,#$%\'()*+-/:;<=>@[\\]^_`{|}~" + '""“”’' + '∞θ÷α•à−β∅³π‘₹´°£€\×™√²—–&'
    df['comment_text'] = df['comment_text'].apply(lambda x: clean_special_chars(x, punct, punct_mapping))
    return df
```

对数据进行清洗操作

```
train = clean_text(train)
test = clean_text(test)
train.head()
```

b) K-fold

首先,根据训练集中的危害性权值辨别话语是否具有危害性,并覆盖 train 中原始的权值数据。
根据 k-fold 方法,在这里我们设置 k 的大小为 5 折,即:将数据集 A 随机分为 5 个包,每次将其中一个包作为测试集,剩下 k-1 个包作为训练集进行训练 。

c) 引入预训练的词向量

在这里，我们用 FastText 引入了两个预处理的词向量

```
embedding_path1 = "../input/fasttext-crawl-300d-2m/crawl-300d-2M.vec"
embedding_path2 = "../input/glove840b300dtxt/glove.840B.300d.txt"
```

基于语料库构建词的共现矩阵，然后基于共现矩阵和 glove 的模型对词汇进行向量化表示.,这里参考了这个 kernel: https://www.kaggle.com/tanreinama/simple-lstm-using-identity-parameters-solution

```
def get_coefs(word,*arr):
    return word, np.asarray(arr, dtype='float32')

def build_matrix(embedding_path, tokenizer):
    embedding_index = dict(get_coefs(*o.strip().split(" ")) for o in open(embedding_path))

    word_index = tk.word_index
    nb_words = min(max_features, len(word_index))
    embedding_matrix = np.zeros((nb_words + 1, embed_size))
    for word, i in word_index.items():
        if i >= max_features:
            continue
        embedding_vector = embedding_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector
    return embedding_matrix

# combining embeddings from this kernel: https://www.kaggle.com/tanreinama/simple-lstm-using-identity-parameters-solution
embedding_matrix = np.concatenate([build_matrix(embedding_path1, tk), build_matrix(embedding_path2, tk)], axis=-1)
```

d) 模型

输入的最大长度设置为 250,首先进行 128 个卷积和,大小为 2 的一次卷积,接下来的池化操作设置 seq_size 为 5,即将原先数据的 size 变成原来的五分之一。接下来再进行一次卷积操作和池化操作，然后把得到的数据转换为只有一层的数组（Flatten()函数的操作），网络构建完成。

Concatenate 将 Attention 对象对应的数组和刚才生成的 x 互相拼合,其实质等同于

二维 list 的加法运算

Dropout()函数是指在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃。

这里需要解释的是,对于随机梯度下降来说，由于是随机丢弃，故而每一个 mini-batch 都在训练不同的网络。每个神经网络单元每次训练丢弃的概率都相同,在训练 n 次之后的利用次数的期望值是相同的,这样做的好处是既增加了随机性,又不会造成数据的利用缺失，从而成为 CNN 中防止过拟合提高效果的一个常用方法。

最后，利用 sigmoid()函数,将输入的 imp 数据转换成二进制数据输出。

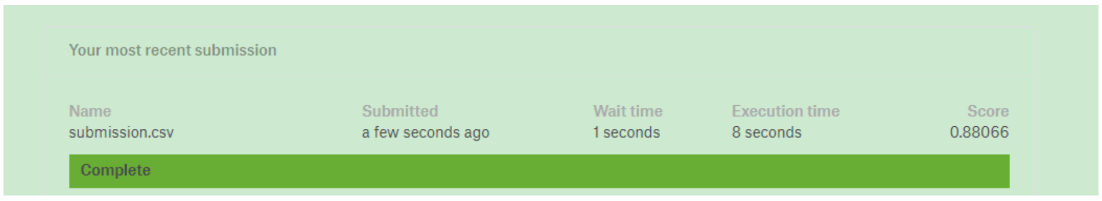
e) 训练,预测数据方法

在训练方法中,我们主要运用到了分词器 Tokenizer,其工作就是分解文本流成词(也就是 tokens)。在我们的数据集文本中,每一个 token 都是这些字符的一个子序列。分析器必须知道它所配置的字段,但是 tokenizer 不需要,分词器从我们所给的字符流读取数据,生成一个 Token 对象的序列。

一个标准的分词器提供基于语法的分词器，那是一个适合大部分欧洲语言文档的很好的分词器。分词器实现 Unicode 文本分割算法，该分割算法在 Unicode Standard Annex #29 中指定。分词的方法分为连词分词器，关键字分词器，正则分词器。在这个程序里，我们使用的是空格分词器（以空格来分词）。

在我们字符串处理（见“五、方法——a）”）输入的过程中，我们将空格设定成为字符串的分隔符，那么在 Tokenizer 读入输入流的时候，所有的空格将会被丢弃。一个 token 包含多种元数据除了它的原始文本值,如字段中词(token)出现的位置。

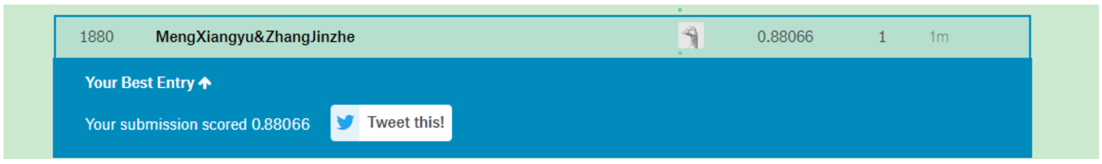
六、 结果



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission.csv	a few seconds ago	1 seconds	8 seconds	0.88066

Complete



1880	MengXiangyu&ZhangJinzhe	0.88066	1	1m

Your Best Entry

Your submission scored 0.88066

七、 结果分析

我们在自己的电脑上运行，结果为 0.94，但是在 kaggle 服务器上运行的准确率为 0.88。提

交代码的总人数和是 2000 多人，虽然我们通过自己的努力完成了指定的任务，但是在比赛中获得的成绩并不乐观，我们分析了可能存在的原因：

- 训练次数过少。由于时间原因，我们只训练了三轮的数据。训练次数过少会导致模型没有拟合到位，训练次数过多会导致模型过拟合。如果想得到合适的结果，则需要知道恰当的训练次数，但是通过试验方法（通常通过指数查找+二分查找的方法）选择恰当的训练次数需要时间成本。以后在时间充裕的前提下，我们会考虑求得恰当的训练次数。
- 训练数据过少，给予模型的不够。这是深度学习领域的一个普遍的痛点，属于客观原因。应该尽量想办法克服（如使用增强数据等方法）

八、 心得

孟祥宇

通过几个周的学习,我对自然语言处理有了初步的认识.对信息抽取/对话系统,情感分析技术有了一定的了解..也学会的一种写文档的方法:”解决这个问题的方法有三类：第一基于规则方法，第二机器学习，第三两种的结合.”而最后的结课作业,加深了我对 NLP 的理解,使我更深入地体验 keras/sklearn 的相关知识,在 kaggle 网站看了很多大佬们的文章,受益匪浅,我看到人外有人,天外有天,每天网站都会用涌现出新想法/新 kenel,他最主要是开阔了我的眼界,让我学会了一种新的学习知识的方法!

张金哲

通过此次自然语言处理的作业，我体验到了处理自然语言这个过程的魅力，体会了处理成功自然语言，让机器读懂自然语言的成就感。但是我觉得最主要的是在这个过程中学习到的有关于自然语言处理的知识。主流的自然语言处理技术是以统计机器学习为基础的，虽然说我们对统计学习比较了解，但是还是有几个特别难的地方。比如词义的消岐和句法的模糊性，老师曾经在课上讲过“猴子想吃香蕉，因为它饿了。”和“猴子想吃香蕉，因为它熟透了”的例子，以及课上各种稀奇古怪的叠字中文句子。这些例子在课上可能当作笑话来听，但是其实在课后处理实际的句子时候很多分类错误都是由于句法的模糊性所导致的。这让我们很头疼，处理的时间很长，但实际上我们的确在实践中增长了代码能力，增加了自己的耐心。

此外，我还针对此次比赛了解了如何解决语境的问题。由于现在的自然语言处理技术强调了“大规模”，强调了“真实文本”，现在的大规模真实语料库和大规模、信息丰富的词典的编制工作都得到了加强。在这次的比赛中，我们每次处理的单位不能仅仅根据单个词语和句子，还需要根据语境进行优化的处理。

九、 参考文献

T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin. Advances in Pre-Training Distributed Word Representations

Junhua Ding, XinChuan Li, Venkat N. Gudivada, "Augmentation and evaluation of training data for deep learning", Big Data (Big Data) 2017 IEEE International Conference on, pp. 2603-2611, 2017.

Abdullah, Mohammad S. Hasan, "An application of pre-trained CNN for image classification", Computer and Information Technology (ICCIT) 2017 20th International Conference of, pp. 1-6, 2017.

Y. M. Kassim, V B. S. Prasath, O. V. Glinskii, V. V. Glinsky, V. H. Huxley, K. Palaniappan, "Microvasculature segmentation of arterioles using deep CNN", Image Processing (ICIP) 2017 IEEE International Conference on, pp. 580-584, 2017.

Chang Liu, Yu Cao, Marlon Alcantara, Benyuan Liu, Maria Brunette, Jesus Peinado, Walter Curioso, "TX-CNN: Detecting tuberculosis in chest X-ray images using convolutional neural network", Image Processing (ICIP) 2017 IEEE International Conference on, pp. 2314-2318, 2017.

Gianni S., et al. Unifying features in protein-folding mechanisms, Proc. Natl Acad. Sci. USA , 2003, vol. 100 (pg. 13286-13291)

Compiani M., et al. Dynamics of the minimally frustrated helices determine the hierarchical folding of small helical proteins, Phys. Rev. E Stat. Nonlin. Soft Matter Phys. , 2004, vol. 69 (pg. 051905-051909)

Zhou H.,   Zhou Y.. Folding rate prediction using total contact distance, Biophys. J. , 2002, vol. 82 (pg. 458-463)

B. McCann, J. Bradbury, C. Xiong, and R. Socher, “Learned in Translation: Contextualized Word

Vectors,” ArXiv e-prints, July 2017.

S. Arora, Y. Liang, and T. Ma, “A simple but tough-to-beat baseline for sentence embeddings,” 2017.

M.-C. De Marneffe, C. Archambeau, P. Dupont, and M. Verleysen, “Local vector-based models for sense discrimination,” 03 2018.

G. Simon, “tensorflow-glove,” Mar 2017.

T. Kubota, “Technique can see objects hidden around corners,” Mar 2018

Aytar, Yusuf, Vondrick, Carl, and Torralba, Antonio.
Soundnet: Learning sound representations from unlabeled video. In Advances in Neural Information Processing Systems, pp. 892–900, 2016.

Choi, Keunwoo, Fazekas, George, and Sandler, Mark. Automatic tagging using deep convolutional neural networks. In The 17th International Society of Music Information

Retrieval Conference, New York, USA. International Society of Music Information Retrieval, 2016.

McFee, Brian, McVicar, Matt, Nieto, Oriol, Balke, Stefan, Thome, Carl, Liang, Dawen, Battenberg, Eric, Moore,
Josh, Bittner, Rachel, Yamamoto, Ryuichi, Ellis, Dan, Stoter, Fabian-Robert, Repetto, Douglas, Waloschek, Simon, Carr, CJ, Kranzler, Seth, Choi, Keunwoo, Viktorin, Petr, Santos, Joao Felipe, Holovaty, Adrian, Pimenta, Waldir, and Lee, Hojin. librosa 0.5.0, February 2017. URL https://doi.org/10.5281/zenodo.293021.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints, abs/1605.02688, May 2016. URL http://arxiv.org/abs/1605.02688.

Van Merrienboer, Bart, Bahdanau, Dzmitry, Dumoulin, ¨Vincent, Serdyuk, Dmitriy, Warde-Farley, David, Chorowski, Jan, and Bengio, Yoshua. Blocks and fuel: Frameworks for deep learning. arXiv preprint arXiv:1506.00619, 2015.

附录：代码

```
import datetime
import time
import os
from scipy import stats
from scipy.sparse import hstack, csr_matrix
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score

from nltk.tokenize import TweetTokenizer
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns #https://www.cnblogs.com/kylinlin/p/5236601.html,Seaborn 其实是在 matplotlib 的基础上进行了更高级的 API 封装，从而使得作图更加容易
from nltk.corpus import stopwords
from nltk.util import ngrams
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import metrics
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, LSTM, Embedding, Conv1D, Dropout, Activation, GRU, CuDNNGRU, Input, CuDNNLSTM, BatchNormalization
from keras.layers import Bidirectional, GlobalMaxPool1D, MaxPooling1D, Add, Flatten
from keras.layers import GlobalAveragePooling1D, GlobalMaxPooling1D, concatenate, SpatialDropout1D
from keras.models import Model, load_model
from keras import initializers, regularizers, constraints, optimizers, layers, callbacks
from keras import backend as K
from keras.engine import InputSpec, Layer
from keras.optimizers import Adam
import lightgbm as lgb #LightGBM 是个快速的，分布式的，高性能的基于决策树算法的梯度提升框架。可用于排序，分类，回归以及很多其他的机器学习任务中。
from wordcloud import WordCloud #wordcloud 库，可以说是 python 非常优秀的词云展示第三方库
from collections import Counter #http://www.pythoner.com/205.html ，Counter 类的目的是用来跟踪值出现的次数
from keras.callbacks import ModelCheckpoint, TensorBoard, Callbac
k, EarlyStopping
```

数据读取

```
data_train = pd.read_csv('../input/jigsaw-unintended-bias-in-toxic
ity-classification/train.csv')
data_test = pd.read_csv('../input/jigsaw-unintended-bias-in-toxici
ty-classification/test.csv')
a = data_train['comment_text'].fillna('')
b = data_test['comment_text'].fillna('')
data_train['comment_text'] = a
data_test['comment_text'] = b
sub = pd.read_csv('../input/jigsaw-unintended-bias-in-toxicity-cla
ssification/sample_submission.csv')
```

数据清洗

```
def cleanchars(text, punct, mapping):
        for p in mapping:
            text = text.replace(p, mapping[p])
        for p in punct:
            text = text.replace(p, f' {p} ')
        return text

def cleantext(df):
    df['comment_text'] = df['comment_text'].apply(lambda x: x.lower
())
    punct_mapping = {"_":" ", "`":" "}
    punct = "/-'?!.,#$%\'()*+-/:;<=>@[\\]^_`{|}~" + '""“”’' + '∞θ÷α
•à−βØ³π'₹´°£€\×™√²—–&'
    df['comment_text'] = df['comment_text'].apply(lambda x: cleanch
ars(x, punct, punct_mapping))
    return df

data_train = cleantext(data_train)
data_test = cleantext(data_test)
```

```
full_text = list(data_train['comment_text'].values) + list(data_test['comment_text'].values)
max_features = 300000
tk = Tokenizer(lower = True, filters='', num_words=max_features)
tk.fit_on_texts(full_text)
```

```
embedding_path1 =  '../input/fasttext-crawl-300d-2m/crawl-300d-2M.vec'
embedding_path2 = '../input/glove840b300dtxt/glove.840B.300d.txt'
embed_size = 300
```

```
def get_coefs(word,*arr):
    return word, np.asarray(arr, dtype='float32')#array 和 asarray 都可将结构数据转换为 ndarray 类型。但是，array 仍会 copy 出一个副本，占用新的内存，但 asarray 不会。

def build_matrix(embedding_path, tokenizer):#构建矩阵
    embedding_index = dict(get_coefs(*o.strip().split(" ")) for o in open(embedding_path,encoding='gb18030', errors='ignore'))

    word_index = tk.word_index
    nb_words = min(max_features, len(word_index))
    embedding_matrix = np.zeros((nb_words + 1, embed_size))
    for word, i in word_index.items():
        if i >= max_features:
            continue
        embedding_vector = embedding_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector
    return embedding_matrix
#这里是把两个结构拼起来，axis=-1 和=1 是一样的,都是按照纵向拼一块儿了
embedding_matrix = np.concatenate([build_matrix(embedding_path1, tk), build_matrix(embedding_path2, tk)], axis=-1)
embedding_matrix
```

k-fold

```
y = np.where(data_train['target'] >= 0.5, True, False) * 1 #1. np.where(condition, x, y)满足条件(condition)，输出 x，不满足输出 y。
```

```
identity_columns = ['male', 'female', 'homosexual_gay_or_lesbian', 'christian', 'jewish', 'muslim', 'black', 'white', 'psychiatric_or_mental_illness']
```

```
for col in identity_columns + ['target']:
    data_train[col] = np.where(data_train[col] >= 0.5, True, False)

n_fold = 5
folds = StratifiedKFold(n_splits=n_fold, shuffle=True, random_stat
e=11)
```

数据的评价指标

```
SUBGROUP_AUC = 'subgroup_auc'
BPSN_AUC = 'bpsn_auc'  # stands for background positive, subgroup n
egative
BNSP_AUC = 'bnsp_auc'  # stands for background negative, subgroup p
ositive

def compute_auc(y_true, y_pred):
    try:
        return metrics.roc_auc_score(y_true, y_pred)#Metric 属于一种
计量体系，衡量标准
    except ValueError:
        return np.nan

def compute_subgroup_auc(df, subgroup, label, oof_name):
    subgroup_examples = df[df[subgroup]]
    return compute_auc(subgroup_examples[label], subgroup_examples
[oof_name])

def compute_bpsn_auc(df, subgroup, label, oof_name):
    """Computes the AUC of the within-subgroup negative examples an
d the background positive examples."""
    subgroup_negative_examples = df[df[subgroup] & ~df[label]]
    non_subgroup_positive_examples = df[~df[subgroup] & df[label]]
    examples = subgroup_negative_examples.append(non_subgroup_posi
tive_examples)
    return compute_auc(examples[label], examples[oof_name])

def compute_bnsp_auc(df, subgroup, label, oof_name):
    subgroup_positive_examples = df[df[subgroup] & df[label]]
    non_subgroup_negative_examples = df[~df[subgroup] & ~df[label]]
    examples = subgroup_positive_examples.append(non_subgroup_nega
tive_examples)
    return compute_auc(examples[label], examples[oof_name])
```

```python
def compute_bias_metrics_for_model(dataset,
                                   subgroups,
                                   model,
                                   label_col,
                                   include_asegs=False):
    """Computes per-subgroup metrics for all subgroups and one model."""
    records = []
    for subgroup in subgroups:
        record = {
            'subgroup': subgroup,
            'subgroup_size': len(dataset[dataset[subgroup]])
        }
        record[SUBGROUP_AUC] = compute_subgroup_auc(dataset, subgroup, label_col, model)
        record[BPSN_AUC] = compute_bpsn_auc(dataset, subgroup, label_col, model)
        record[BNSP_AUC] = compute_bnsp_auc(dataset, subgroup, label_col, model)
        records.append(record)
    return pd.DataFrame(records).sort_values('subgroup_auc', ascending=True)

def calculate_overall_auc(df, oof_name):
    true_labels = df['target']
    predicted_labels = df[oof_name]
    return metrics.roc_auc_score(true_labels, predicted_labels)

def power_mean(series, p):
    total = sum(np.power(series, p))
    return np.power(total / len(series), 1 / p)

def get_final_metric(bias_df, overall_auc, POWER=-5, OVERALL_MODEL_WEIGHT=0.25):
    bias_score = np.average([
        power_mean(bias_df[SUBGROUP_AUC], POWER),
        power_mean(bias_df[BPSN_AUC], POWER),
        power_mean(bias_df[BNSP_AUC], POWER)
    ])
    return (OVERALL_MODEL_WEIGHT * overall_auc) + ((1 - OVERALL_MODEL_WEIGHT) * bias_score)
```

模型

```python
# adding attention from this kernel: https://www.kaggle.com/christofhenkel/keras-baseline-lstm-attention-5-fold
class Attention(Layer):
    def __init__(self, step_dim,
                 W_regularizer=None, b_regularizer=None,
                 W_constraint=None, b_constraint=None,
                 bias=True, **kwargs):
        self.supports_masking = True
        self.init = initializers.get('glorot_uniform')

        self.W_regularizer = regularizers.get(W_regularizer)
        self.b_regularizer = regularizers.get(b_regularizer)

        self.W_constraint = constraints.get(W_constraint)
        self.b_constraint = constraints.get(b_constraint)

        self.bias = bias
        self.step_dim = step_dim
        self.features_dim = 0
        super(Attention, self).__init__(**kwargs)

    def build(self, input_shape):
        assert len(input_shape) == 3

        self.W = self.add_weight((input_shape[-1],),
                                 initializer=self.init,
                                 name='{}_W'.format(self.name),
                                 regularizer=self.W_regularizer,
                                 constraint=self.W_constraint)
        self.features_dim = input_shape[-1]

        if self.bias:
            self.b = self.add_weight((input_shape[1],),
                                     initializer='zero',
                                     name='{}_b'.format(self.name),
                                     regularizer=self.b_regularizer,
                                     constraint=self.b_constraint)
        else:
            self.b = None
        self.built = True

    def compute_mask(self, input, input_mask=None):
        return None

    def call(self, x, mask=None):
        features_dim = self.features_dim
        step_dim = self.step_dim

        eij = K.reshape(K.dot(K.reshape(x, (-1, features_dim)),
                        K.reshape(self.W, (features_dim, 1))), (-1, st
ep_dim))

        if self.bias:
            eij += self.b

        eij = K.tanh(eij)

        a = K.exp(eij)

        if mask is not None:
            a *= K.cast(mask, K.floatx())

        a /= K.cast(K.sum(a, axis=1, keepdims=True) + K.epsilon(),
K.floatx())

        a = K.expand_dims(a)
        weighted_input = x * a
        return K.sum(weighted_input, axis=1)

    def compute_output_shape(self, input_shape):
        return input_shape[0],  self.features_dim


def build_model(X_train, y_train, X_valid, y_valid, max_len, max_f
eatures, embed_size, embedding_matrix, lr=0.0, lr_d=0.0, spatial_d
r=0.0,
                dense_units=128, conv_size=128, dr=0.2, patience=3, f
old_id=1):
    file_path = f"best_model_fold_{fold_id}.hdf5"
    check_point = ModelCheckpoint(file_path, monitor="val_loss", ve
rbose=1,save_best_only=True, mode="min")
    early_stop = EarlyStopping(monitor="val_loss", mode="min", pati
ence=patience)
```

```
    inp = Input(shape = (max_len,))
    x = Embedding(max_features + 1, embed_size * 2, weights=[embedding_matrix], trainable=False)(inp)
    x1 = SpatialDropout1D(spatial_dr)(x)
    att = Attention(max_len)(x1)
    # from benchmark kernel
    x = Conv1D(conv_size, 2, activation='relu', padding='same')(x1)
    x = MaxPooling1D(5, padding='same')(x)
    x = Conv1D(conv_size, 3, activation='relu', padding='same')(x)
    x = MaxPooling1D(5, padding='same')(x)
    x = Flatten()(x)
    x = concatenate([x, att])

    x = Dropout(dr)(Dense(dense_units, activation='relu') (x))
    x = Dense(1, activation="sigmoid")(x)

    model = Model(inputs=inp, outputs=x)
    model.compile(loss="binary_crossentropy", optimizer=Adam(lr=lr, decay=lr_d), metrics=["accuracy"])
    model.fit(X_train, y_train, batch_size=128, epochs=3, validation_data=(X_valid, y_valid),
                    verbose=2, callbacks=[early_stop, check_point])
    return model
```

```
def train_model(X, X_test, y, tokenizer, max_len):

    oof = np.zeros((len(X), 1))
    prediction = np.zeros((len(X_test), 1))
    scores = []
    data_test_tokenized = tokenizer.texts_to_sequences(data_test['comment_text'])
    X_test = pad_sequences(test_tokenized, maxlen = max_len)
    for fold_n, (train_index, valid_index) in enumerate(folds.split(X, y)):
        print('Fold', fold_n, 'started at', time.ctime())
        X_train, X_valid = X.iloc[train_index], X.iloc[valid_index]
        y_train, y_valid = y.iloc[train_index], y.iloc[valid_index]
        valid_df = X_valid.copy()

        train_tokenized = tokenizer.texts_to_sequences(X_train['comment_text'])
        valid_tokenized = tokenizer.texts_to_sequences(X_valid['comment_text'])

        X_train = pad_sequences(train_tokenized, maxlen = max_len)
        X_valid = pad_sequences(valid_tokenized, maxlen = max_len)

        model = build_model(X_train, y_train, X_valid, y_valid, max_len, max_features, embed_size, embedding_matrix,
                            lr = 1e-3, lr_d = 0, spatial_dr = 0.1, dense_units=128, conv_size=128, dr=0.1, patience=3, fold_id=fold_n)

        pred_valid = model.predict(X_valid)
        oof[valid_index] = pred_valid
        valid_df[oof_name] = pred_valid

        bias_metrics_df = compute_bias_metrics_for_model(valid_df, identity_columns, oof_name, 'target')
        scores.append(get_final_metric(bias_metrics_df, calculate_overall_auc(valid_df, oof_name)))

        prediction += model.predict(X_test, batch_size = 1024, verbose = 1)

    prediction /= n_fold

    # print('CV mean score: {0:.4f}, std: {1:.4f}.'.format(np.mean(scores), np.std(scores)))
    return oof, prediction, scores
```

预测训练数据

```
oof_name = 'predicted_target'
max_len = 250
oof, prediction, scores = train_model(X= data_train, X_test= data_test, y= data_train['target'], tokenizer=tk, max_len=max_len)
print('CV mean score: {0:.4f}, std: {1:.4f}.'.format(np.mean(scores), np.std(scores)))
```

```
sub['prediction'] = prediction
sub.to_csv('submission.csv', index=False)
```