|  |
| --- |
| **机器学习** |
| Toxic Comment Classification Challenge |
| 主 研 人：岳天昕  参 研 人： |
| 审 核 人：  方 向：工业算法案例研究  版 本 号：A |
|  |

**版本更新**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **更新描述** | **作者** |
| 2018/10/31 | A | 初稿 |  |

**选题表格**

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 竞赛名 | 竞赛背景描述（50字以内） | 类型（分类/回归） |
| 2018/10/31 | 网络有害言论识别 | 在此题中，我们将建立一个算法，它能更好地检测网络上威胁、淫秽、侮辱和基于身份的仇恨等不同类型的有害信息。对当前模型的改进有望将在线讨论变得更有成效和充满尊重。 | 分类 |

目录

[1. 背景描述 4](#_Toc25575_WPSOffice_Level1)

[1.1 竞赛赛题描述 4](#_Toc5940_WPSOffice_Level2)

[1.2 评估指标描述 5](#_Toc12357_WPSOffice_Level2)

[2. 数据来源及描述性统计分析 5](#_Toc5940_WPSOffice_Level1)

[2.1 大赛数据来源 5](#_Toc22683_WPSOffice_Level2)

[2.2 数据的描述性统计 5](#_Toc16174_WPSOffice_Level2)

[2.2.1 数据基本情况描述： 5](#_Toc5940_WPSOffice_Level3)

[2.2.2 数据字段介绍： 5](#_Toc12357_WPSOffice_Level3)

[2.2.3 数据描述性统计 6](#_Toc22683_WPSOffice_Level3)

[3. 优秀算法思路 6](#_Toc12357_WPSOffice_Level1)

[3.1 方案一 6](#_Toc32293_WPSOffice_Level2)

[3.1.1 方案一数据预处理及特征工程部分方案 6](#_Toc16174_WPSOffice_Level3)

[3.1.2 方案一模型设计、建立部分方案 6](#_Toc32293_WPSOffice_Level3)

[3.1.3 方案一结果、排名等 6](#_Toc29922_WPSOffice_Level3)

[3.1.4 方案一算法流程图 6](#_Toc4787_WPSOffice_Level3)

[3.2 方案二 7](#_Toc29922_WPSOffice_Level2)

[… 8](#_Toc4787_WPSOffice_Level2)

[4. 算法比较 8](#_Toc22683_WPSOffice_Level1)

[表4-1 算法比较 8](#_Toc7253_WPSOffice_Level2)

[5. 总结与展望 8](#_Toc16174_WPSOffice_Level1)

[5.1 总结 8](#_Toc23198_WPSOffice_Level2)

[5.2 建模思路 8](#_Toc16619_WPSOffice_Level2)

1. 背景描述

在网络上讨论你关心的事情有时会很困难。网络上的言语霸凌和骚扰威胁使许多人不再表达自己的想法，也放弃寻求多元化的意见。平台不能保证网络对话的有效性和尊重性，致使许多社区限制或完全关闭用户评论功能。

对话人工智能团队（conversation AI team）是由Jigsaw和谷歌发起的一项研究计划，目的是开发工具来帮助改善网络交流。其中关注的一个领域是对负面网络行为的研究，比如含毒害性的评论（即粗鲁的、不尊重的或让人没有讨论机会的评论）。到目前为止，他们已经建立了一系列通过透视API提供服务的公共可用模型，包括识别毒害性评论的模型。但是，目前的模型仍然存在错误，并且不允许用户选择他们感兴趣的含某些毒害性的问题类型（例如，某些平台可能接受一些可能别识别为含毒害性的言论，但其他平台不接受这种毒害性内容）。

**1.1 竞赛赛题描述**

在此次竞赛中，你面临的挑战是建立一个multi-headed模型，它能够比当前模型更好地检测不同类型的毒害性言论，如威胁、淫秽、侮辱和基于身份的仇恨。你将使用维基百科谈话页面编辑的评论数据集。对当前模式的改进有望使在线讨论变得更有效率和充满尊重。

**1.2 评估指标描述**

竞赛中评估模型优劣的指标。

此次竞赛采用平均列的ROC AUC。换句话说，竞赛得分为每个预测列的单个AUC的平均值。

1. 数据来源及描述性统计分析

**2.1 大赛数据来源**

数据为大量的维基百科评论

**2.2 数据的描述性统计**

**2.2.1 数据基本情况描述：**

数据超链接：<https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data>

参赛者会得到大量的维基百科评论，这些评论已经被人类评分者贴上了有害言论的标签。有害类型分为：

* toxic
* sever\_toxic
* obscene
* threat
* insult
* identity\_hate

**2.2.2 数据字段介绍：**

**表2-1 训练数据表字段介绍**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| **Toxic** | **有害性** | 离散 | 0% |
| **Sever\_toxic** | **严重有害** | 离散 | 0% |
| **Obscene** | **淫秽的** | 离散 | 0% |
| **Threat** | **威胁** | 离散 | 0% |
| **Insult** | **辱骂** | 离散 | 0% |
| **Identity\_hate** | **基于身份的歧视** | 离散 | 0% |
| **id** | **用户id** | 离散 | 0% |

**表2-2测试数据表字段介绍**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| **Comment\_text** | **评论文本** | 离散 | 0% |
| **Id** | **用户id** | 离散 | 0% |

* + 1. **数据描述性统计**

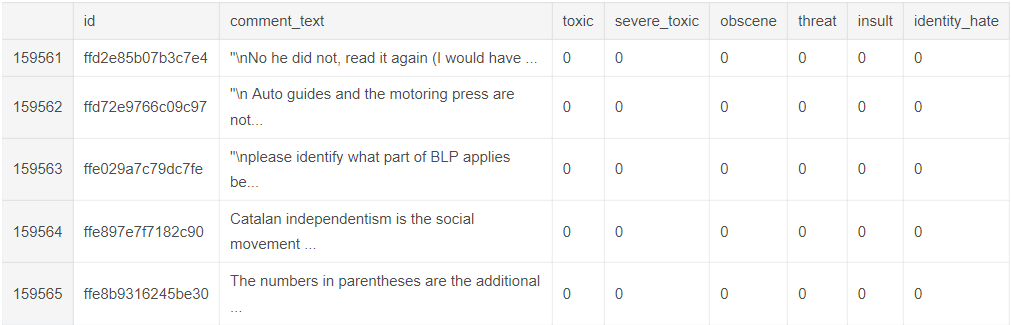
**数据总览：**

这里的数据集来自wiki语料库数据集，该数据集被人工评估者评定有害性，语料库包含了从2004年到2015年关于用户页面和文章的讨论中得出的6300万条评论。

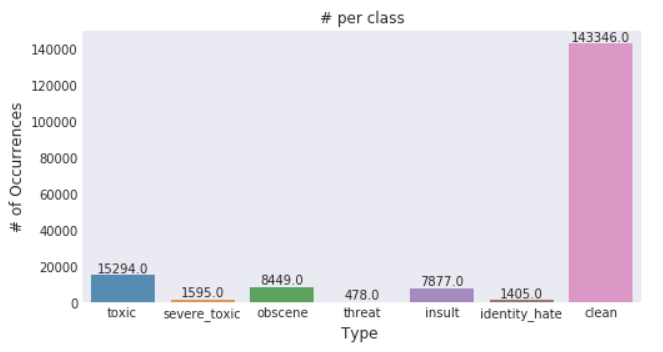
不同的平台/场所对其毒性筛选过程可能有不同的标准。因此，注释标记在以下五个类别中：toxic,sever\_toxic,obscene,threat,insult,identity\_hate

标记是通过众包完成的，这意味着数据集是由不同的人进行评级的，标记也可能不是100%准确。

**训练集预览：**

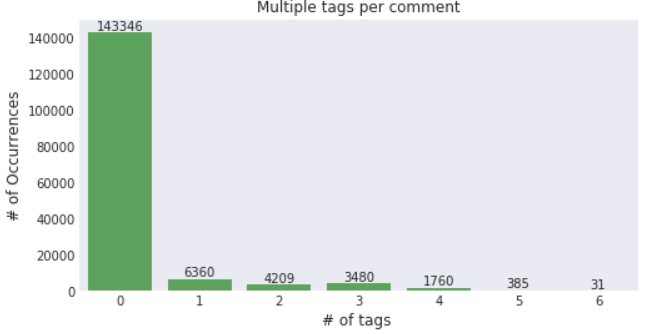


**评论标签：**



可以看出干净的评论占绝大多数

**查看含多个标签的评论**：



看出只有约10%的评论含某种有害性标签。

**查看哪些标签之间含关联性：**

一个很好的指标就是相关图。



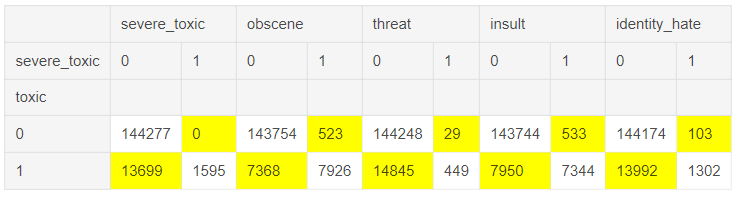
上面的图显示了一种共现模式，但是pandas用皮尔逊相关的默认corr函数在这里不适用，因为涉及的变量是分类（二进制）变量。

因此，为了在两个分类变量之间找到一个模式，我们可以使用其他工具，比如:

混淆矩阵/交叉表

克莱默v统计量

cramer的v stat是卡方检验的一个扩展，在卡方检验中，关联度/强度也被测量



上表表示与其他类的有害评论的交叉表/信息。

一些有趣的观察：sever\_toxic 的评论总是toxic。

除了少数例外，其他类似乎是toxic的子集。

**查看一些有害评论后发现：**

有些评论是极其简单的复制粘贴

注释仍然可能包含IP地址（例如：62.158.73.165）、用户名（例如：arkjedi10）和一些神秘数字（我猜测是文章ID），这一点极可能造成过拟合。

**Wordclouds常用词：**

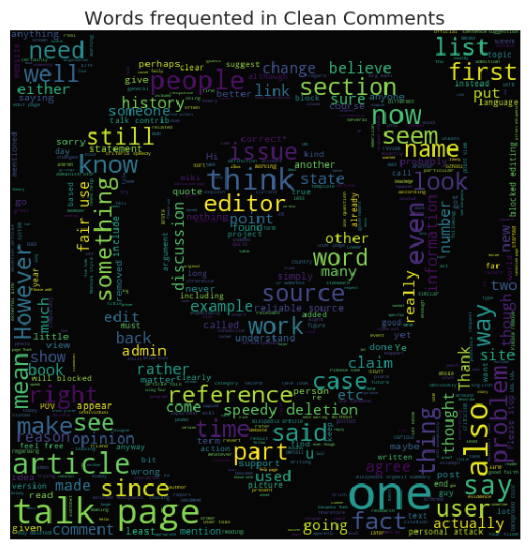
现在，让我们看看与这些类相关联的单词。

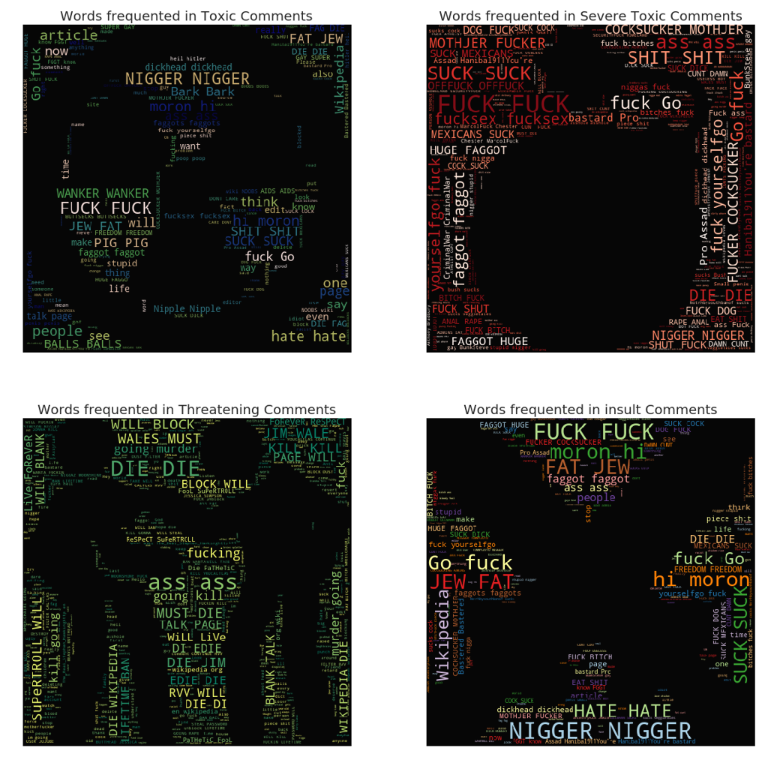
图表描述：这里的视觉效果是单词云使更频繁的单词显得更大。这里给出了一种用图片创建单词云的酷方法。它涉及以下步骤：

\*搜索图像及其基64编码

\*在单元格中粘贴编码并使用codecs包将其转换为图像

\*使用新图像创建Word Cloud作为遮罩





**特征工程：**

直接特征：直接由于文字/内容而产生的特征。我们将探索以下内容

词频特征、词的矢量距离映射（例如：word2vec）、 情绪得分

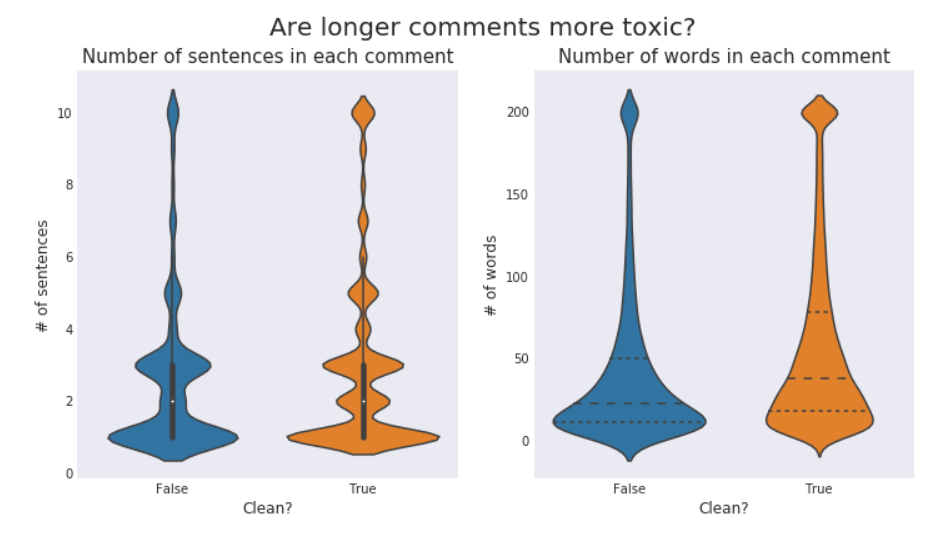
间接特征：

句数、字数、唯一字数、字数、标点计数、大写单词/字母计数、停止字数、每个词的平均长度

**泄露特征：**

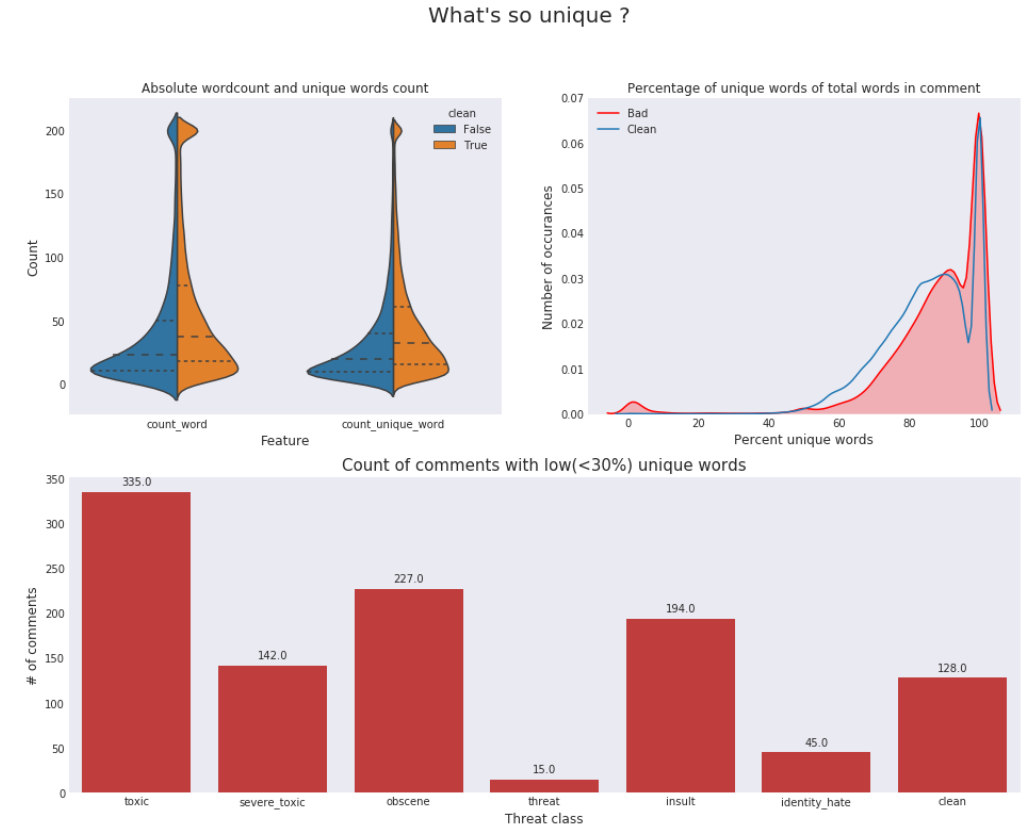
从示例中，我们知道注释包含标识符信息（例如：IP、用户名等）。我们可以从中创建特性，但是，这肯定会导致适合这个特定的维基百科用例的过拟合。

注意：首先创建间接和泄漏特性。这有两个原因：计数特征（直接特征）只有在从干净评论创建时才有用，此外，间接特性有助于补偿清除数据集时信息的丢失。



长句子或多个单词似乎不是有害的重要指标。

图表描述：小提琴作图是传统方块作图的替代品。内部标记显示百分位数，“小提琴”的宽度显示该级别/实例的评论量。



字数与特殊字数：

在干净和有害的评论中，字数和唯一字数的平均值都有明显的变化。

图表描述：第一张图表是一张分开的小提琴图表。它是传统盒形图/小提琴图的一种变体，允许我们根据分类变量将小提琴从中间分开。

唯一字数百分比：

在0-10%标记附近有一个凸起，表示大量有害评论，其中包含的词语变化非常小。 图表描述：第二个图表是两个内核密度估计图的叠加，其中包含注释中所有单词中唯一单词的百分比，用于清除和有毒注释。

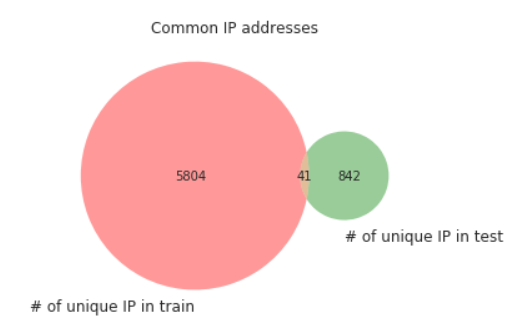
**泄漏特征**

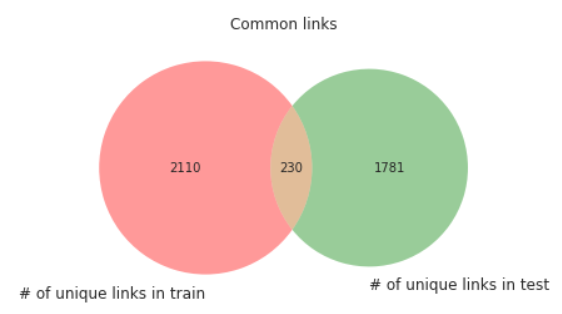
注意：尽管在这个特定的场景中包含这些特性可能有助于我们更好地执行操作，但是在最终模型/通用模型中添加这些特性并不是明智之举。

这里我们创建自己的自定义计数向量器来创建与regex条件匹配的计数变量。

**泄漏特性稳定性：**

检查泄漏特征的再次出现，以检查它们在预测测试集时的效用。





**语料库清理：**

在创建计数功能之前使用干净的数据集是很重要的。

**直接特征：**

1. 基于计数的特征（对于unigrams）：

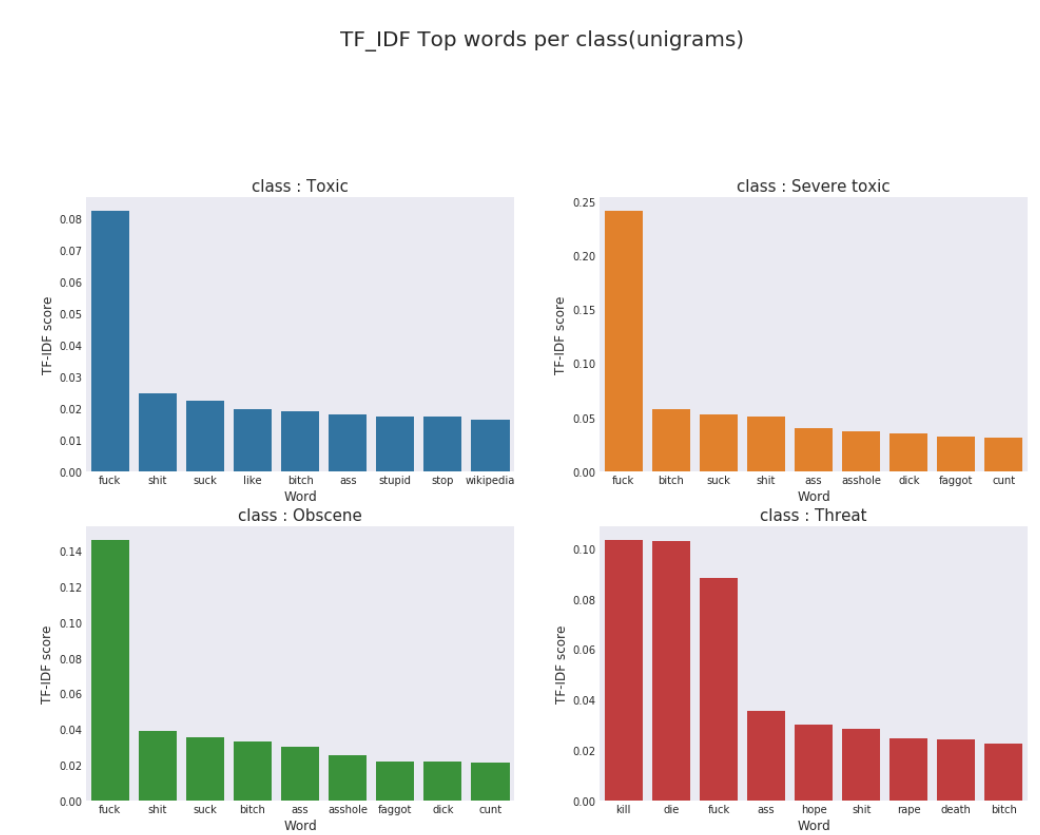
让我们基于单词的频率分布创建一些功能。首先让我们考虑一次取一个单词（即unigrams）。python的sklearn提供了3种创建计数特性的方法，这三种方法首先创建单词的词汇表（字典），然后为字典中存在的句子中的单词创建稀疏的单词计数矩阵。它们的简要描述：

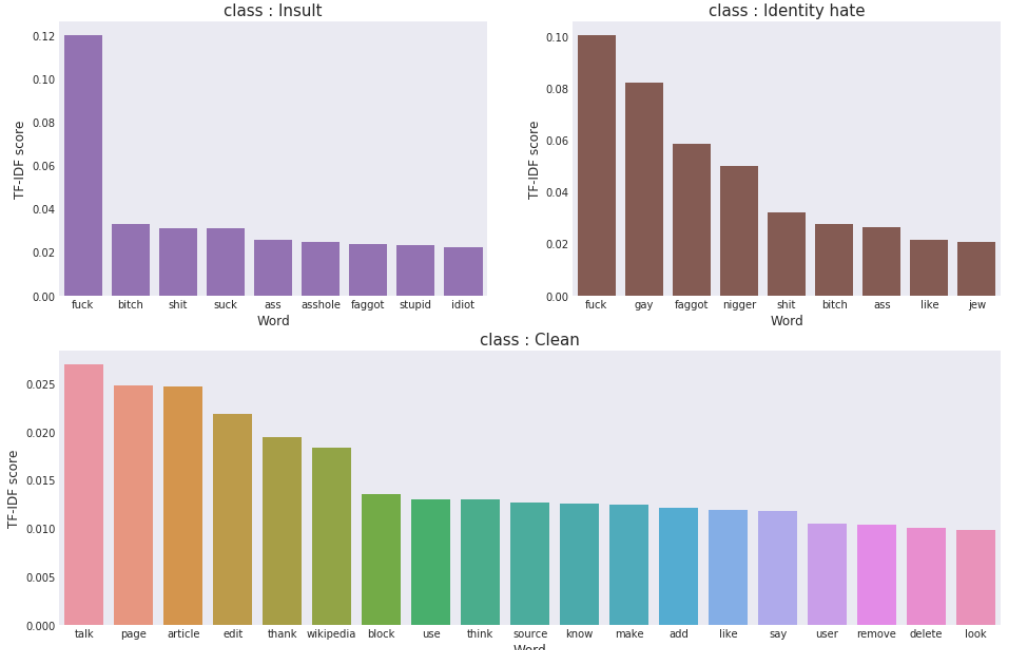
计数记录器-使用文本文集中每个单词的频率计数创建矩阵

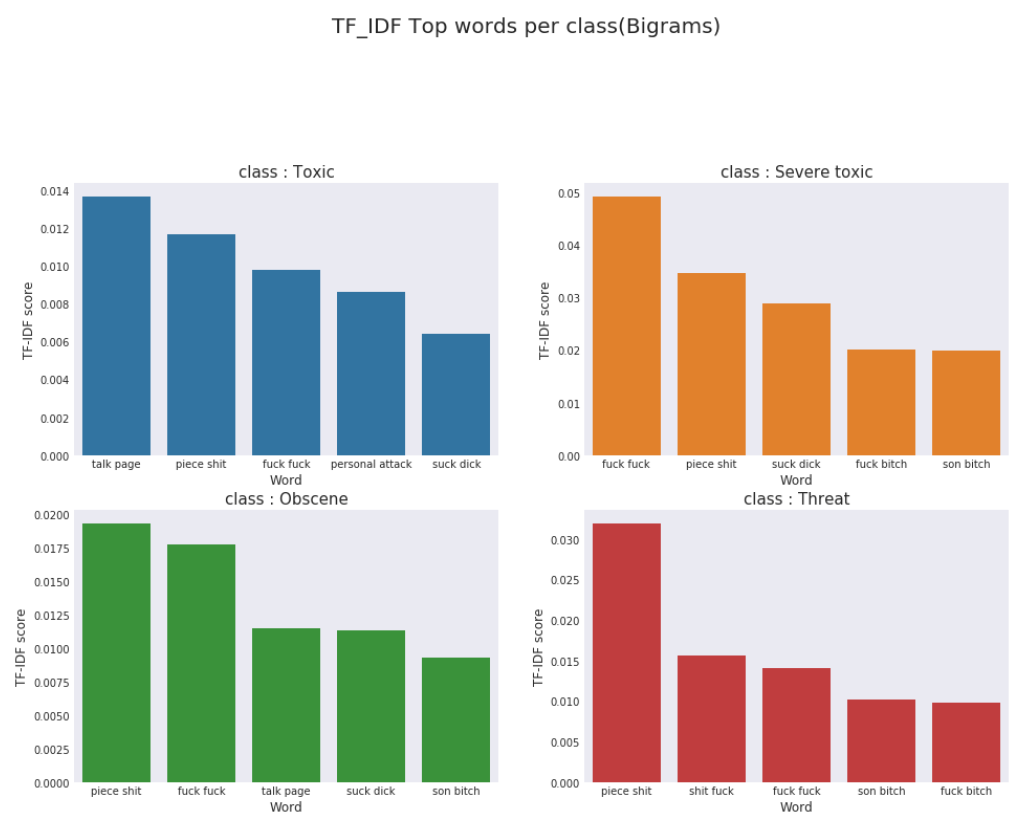
TF-IDF矢量器-tf—词条频率—文本语料库中单词（词条）的计数（与count vect相同）。 IDF—文档反向频率—惩罚过于频繁的单词。我们可以把这看作是正规化

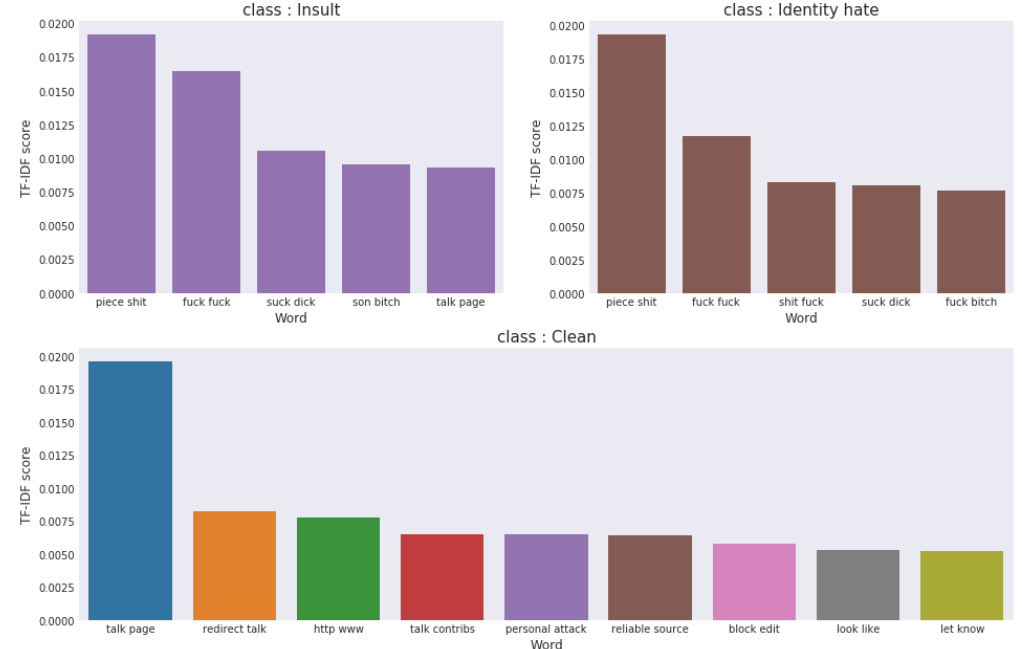
哈希向量器-创建哈希映射（基于哈希技术的单词到数字映射），而不是词汇字典 。 这使得它对于较大的文本coprus更具可扩展性和更快的速度。可以跨多个线程并行

在这里使用tf-idf。注意：使用连接的数据帧“merge”，它包含来自训练和测试数据集的文本，以确保我们创建的词汇表不会遗漏在测试集独有的单词上。









3 优秀算法思路

这个部分麻烦寻找竞赛结果排名至少top10的思路分享（天池大赛会在技术圈置顶，kaggle的话需要自己在交流区搜索一下）优秀算法思路可以包含多个，可以选择2-3个算法方案进行描述。

**3.1 方案一**

**3.1.1** 方案一数据预处理及特征工程部分方案

**3.1.2** 方案一模型设计、建立部分方案

**方案总结：**

1.多样的预训练嵌入（基线公共LB为0.9877）

2.将翻译当做训练/测试时间增强（train/test-time augmentation，TTA）（3.将LB从0.9877提升至0.9880）

Rough-bore伪标签（pseudo-labelling，PL）（将LB从0.9880提升至0.9885）

4.强大的CV和stacking框架（将LB从0.9885提升至0.9890）

我们没有以前的NLP特定领域知识，所以我们的总体策略是以系统的方式测试标准的ML/DL技术。我们很高兴看到TTA和伪标签等通用技术在这里有效地发挥作用。 更不用说我们的硬件设置促进了对解决方案空间的彻底搜索：我们之间有6个GPU。我们对代码进行了优化，使我们能够处理8倍的OOF数据，这些数据是针对100多万个样本（TTA+PL）进行训练的，每个模型在大约2小时内收敛。

**方案细节：**

1. 多样的预训练嵌套层  
   考虑到模型的90%以上的复杂性都存在于嵌入层中，我们决定将重点放在嵌入层上，而不是后嵌入层上。对于后者，我们把两个BiGru层嵌入两个最后的密集层。对于前者，我们在网络上搜索可用的预先训练过的单词嵌入，并主要解决了针对普通Crawl、维基百科和Twitter的预先训练过的最高维度的FastText和Glove嵌入。
2. 将翻译当做训练/测试时间增强TTA

我们利用PavelOstyakov的机器翻译思想，利用法语、德语和西班牙语翻译回英语，对训练和测试数据集进行了扩充。考虑到信息泄露的可能性，我们确保翻译与原始评论保持在同一侧。对于预测，我们简单地平均了4条评论（en、de、fr、es）的预测概率

这对我们的模型的性能产生了巨大的影响。

例如：

香草双gru型：0.9862lb

“（w/训练时间增加）：0.9867 lb

“（w/测试时间增加）：0.9865 lb

“（两个训练/测试时间均增加）：0.9874 lb

换句话说，一个单一的TTAED模型击败了大部分团队（大概是）在公共阶梯上提交的集成文件。我们很好奇这项技术是否主要通过“修正”非英语评论起作用——我们测试了将原始评论直接翻译成英语（这有助于翻译非英语评论的副作用），结果导致了比我们的完整增强形式更低的性能。

1. Rough-bore伪标签（PL）

我们尝试了好几种伪标签变体：规范化每批更新、改变损失函数等。表现最好的变体是利用表现最好的整体对测试样本进行简单标记，将它们添加到训练集中，并进行收敛训练。

1. 稳定的CV和stacking框架

对于堆叠，我们使用了算术平均和堆叠的加权平均，这比任何一种方法的效果都好（〜.0001）。我们主要使用LightGBM，比XGBoost更快，并能通过[贝叶斯](http://www.aibbt.com/a/tag/%e8%b4%9d%e5%8f%b6%e6%96%af/" \t "_blank" \o "查看与 贝叶斯 相关的文章)优化达到更好地CV分数。

我们在贝叶斯优化运行的250次实验中选择了最佳参数；参数中的关键点是深度较小且强L1正则化的小型树。我们用不同的seed对6次运行D[AR](http://www.aibbt.com/a/tag/ar/)T和GBDT的过程进行打包（bagged），已解决堆叠过程中的潜在差异。

对于CV，我们记录了准确性、log loss和AUC。如果将模型添加到我们的栈中，除了能提高public board的分数，还可以改善CV-log loss和CV-AUC。我们丢弃了堆栈+混合中的很多模型，因为担心过度拟合而无法达到上述三个标准的任何一个。

1. 其他收获

在思考解决方案的过程中，我们尝试了不同的方法，也都成功了。我们认为有帮助的地方有以下几点：

1）由于模型大多是因为预训练嵌入而变得复杂，所以架构带来微小的变化对分数的影响很小。更多的密集层、高斯与空间dropout、密集层次上多余的dropout层、注意力、时间分布的密集层以及其他因素也不会对模型的整体分数产生较大影响。

2）预处理也不是特别有效，虽然会在某些模型的嵌入中留下标点符号，这有助于堆叠。

3）许多评论只在最后一句话有攻击性，所以除了起始的200—300个字符模型之外，还要增加训练最后徐25—50个字符的模型。

4）一些方法在处理单词“排序”的问题上有困难。由两个单词组成的短语交换位置后意思可能完全不一样，这意味着CNN方法很难使用，因为它们将最大池化作为支持。我们最好的CNN比最好的[RNN](http://www.aibbt.com/a/tag/rnn/)得分低了0.0015。

5）总的来说，其他结构很难达到与RNN相同水平的表现力。唯一达到RNN标准的架构是Attention Is All You Need，然而它的训练时间更长。

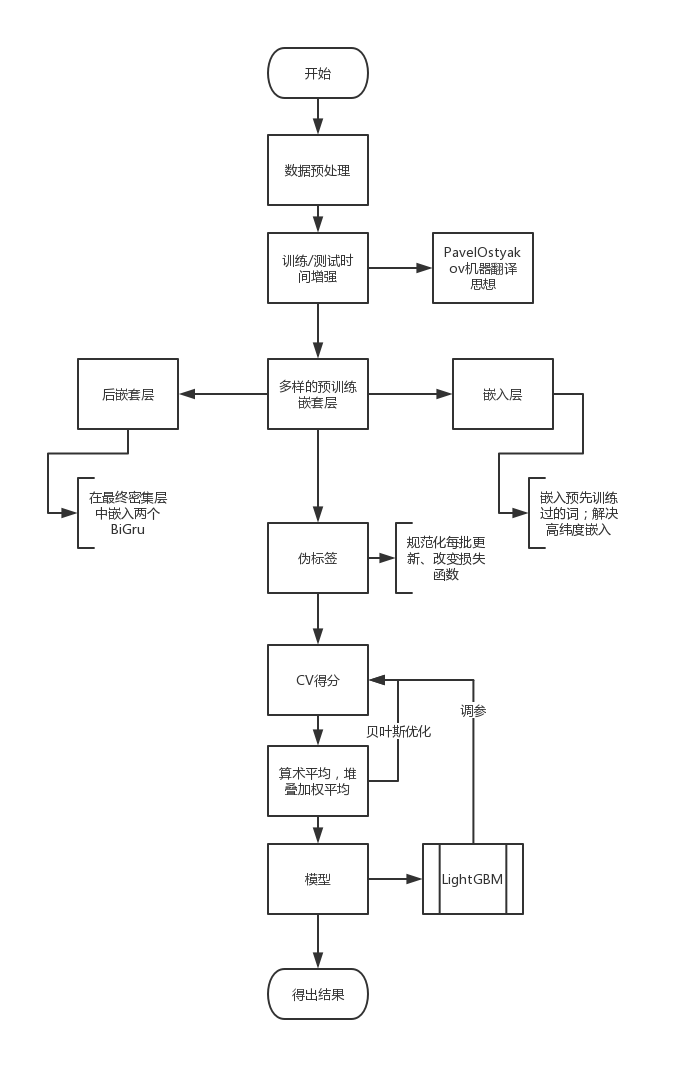
6）另外，TensorFlow在很多方面都优于[Keras](http://www.aibbt.com/a/tag/keras/" \t "_blank" \o "查看与 Keras 相关的文章)——可以试着在[Keras](http://www.aibbt.com/a/tag/keras/)中安装Attention Is All You Need或者可扩展的模型以进行实际演示。也就是说，Keras也很棒，尤其是对快速迭代。

**3.1.3** 方案一结果、排名等

TOP1

Score = 0.98856

**3.1.4** 方案一算法流程图

****

**图3-1**

**3.2 方案二**

**3.2.1**方案一数据预处理及特征工程部分方案

作者未解释

**3.2.1**方案一模型设计、建立部分方案

*Bojan的子模型：*  
我已经完成了我通常的“kitchen sink”方法 - 尝试了几种不同的特征工程方法，算法，嵌入等等。其中大多数都与kernel中发布的东西非常相似 - 简单的单词预处理，单词和字符标记化，使用LogisticRegression，GRU，LSTM等。只有堆叠，n-gram和总特征长度的不同选择，元特征交互术语和几个堆叠层（我想我设法进入第4层），我能够在公共LB上接近0.9810。我将其与几个GRU和LSTM神经网络混合使用，并设法在公共LB / 0.9865上得到0.9870的私人评分。我的模型包括：  
1.1-6 Log和1-4字不同的tf-idf矢量化与Logistic回归  
2.使用glove和快速文本（fasttext），100-300字长和100-300矢量大小嵌入  
3.64-200长度GRU，LSTM和GRU + CNN网络架构  
4.我的一个模型包括两个额外的语言翻译，但是，唉，我没有追求太多。

*Andre’s的子模型：*Andre基本上使用了三种不同的模型：一个双向-GRU层神经网络，一个双向双层-LSTM层神经网络，以及一个XGBoost模型，该模型经过大约100个“分析”特征的训练 - 使用文本中单词和字符的统计信息。 Andre还使用了他自己的一套预处理/语言规范化规则。 每个模型在许多种子上取平均值，这三个模型的最终混合在公共LB上得分为0.9870。  
当Andre和我联手时，我们希望从我们的混合模型中获得一点点提升。 令我们惊讶的是，混合我们的模型让我们在公共排行榜上达到0.9876。

*Ryan和Alex 的子模型：*  
Ryan和Alex进行了大量的特征工程，文本规范化和嵌入，这导致Alex在他的帖子中谈到了令人敬畏的单一0.9872模型。Ryan还谈到了他们在帖子中对各种模型做了什么。他们的模型是我们最终模型的主要支柱。通过堆叠他们的几个模型，他们设法在公共LB上达到0.9878。 他们的主要三个模型是：  
1.双向两级模型，LSTM作为第一层，GRU作为第二层，连接facebook快速文本和glove twitter矢量  
2.具有与单词模型相同的体系结构的字符模型（尽管不同的超参数）  
3.注意力模型。 该模型使用DeepMoji项目的加权注意层。它计算输入字序列的加权注意力，并在几次卷积后计算输入序列。

他们使用的其他模型：  
1.Tfidf模型  
2.在维基百科评论语料库上基于他们自己训练的快速文本向量的模型。  
3.Vowpal Wabbit模型

合并后，我们开发了几种不同的堆叠和混合模型，使用20到35个子模型进行堆叠，最多20个不同的L1和L2模型进行混合。 我们使用10倍CV方案进行堆叠和混合。不幸的是，我们的堆叠或混合解决方案都没有能够突破难以捉摸的0.9880障碍。

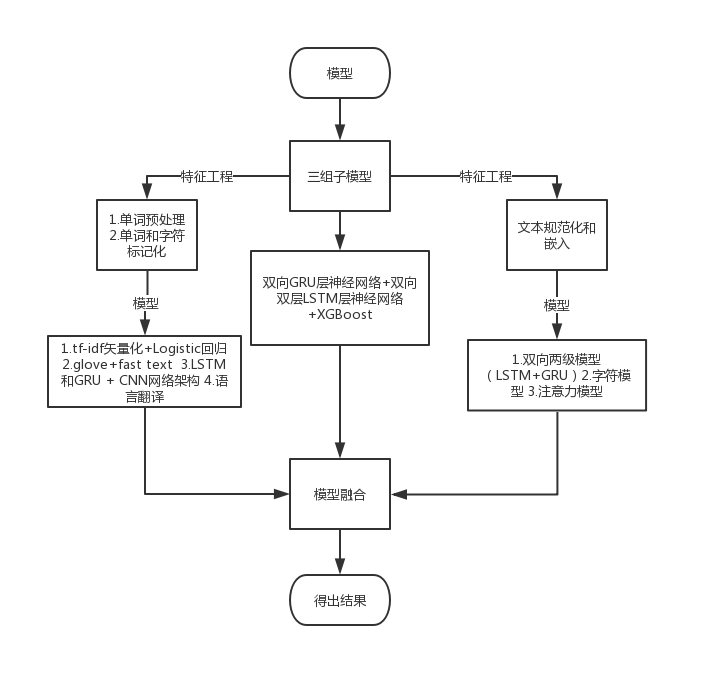
*最终模型*  
最终的模型融合了我们的各种解决方案。我们结合了以下三个组成部分：  
1.融合了我们最好的得分堆叠和局部混合解决方案，所有这些解决方案在公共排行榜上得分约为0.9879，在本地中得分为0.99270-0.99308。该组件占我们最终解决方案的60％。  
2.Bojan最好的个人混合。约20％的最终混合物。  
3.Andre最好的个人融合。约20％的最终混合物。  
然后我们将1240个非英语注释设置为0.这稍微改善了我们的混合。最终的混合在公共排行榜上得分为0.9881，在私人排行榜上得分为0.9880。

**3.2.3** 方案一结果、排名等

Score = 0.98805

**TOP3**

**3.2.4** 方案一算法流程图



4. 算法比较

**表4-1 算法比较**

|  | **评估指标** | **特征工程** | **基础算法** | **基本库** |
| --- | --- | --- | --- | --- |
| **算法1** | **CV** | **TTA** | **LightGBM** | **Sklearn，tensorflow** |
| **算法2** | **CV** |  | **GRU，LSTM，XGBoost** | **Sklearn，keras** |

5. 总结与展望

**5.1 总结**

本题是有关于自然语言处理的题目。我认为此类题型在数据处理方面很重要。对文本单词选取好的嵌套方式可能会对结果产生较大的影响。单词的嵌套我还不太了解需要多加学习。

两个高分方案都注重嵌套层的多样性，且模型性能尽可能地多样。在具体调参的时候也要根据CV等评估系统做出合理的假设和试验。

**5.2 建模思路**

根据待处理文本的特征结合实际需求选择合适的嵌套方式，选取尽可能多的模型进行训练，选择出训练效果好的模型进行权重融合，多次尝试，根据评估结果调整参数、比重，最终得出结果。