**Jigsaw有毒评论分析**

16组：王理琼 张振羽 钟梓健 苏晓甜

1. **数据预处理与可视化**
   1. **数据初步分析与可视化**

如今网络环境快速发展，人们可以利用众多在线讨论平台发表自己的观点与见解，然而部分个体喜欢在网络上发表一些过激言论，只需要一条有毒评论就可以使在线讨论变味。为了确定有害内容，拥有一个安全协作的网络环境，我们对谷歌的评论内容进行毒性分析。首先给出毒性的定义，任何粗鲁、无礼或其他可能导致某人离开讨论的行为都视为毒性，进一步，它主要包含四个方面的内容：淫秽、威胁、侮辱以及人身攻击。在后续分析中，我们将无毒评论定义为0，将有毒评论定义为1。

对数据集进行简单可视化，观察训练集和测试集的样本分布情况，如图1、2所示。可以看到，评论在不同语言的分布较为均衡，然而标签数据严重失衡，无毒评论数量远大于有毒评论数量。我们认为这可能会对模型训练有一定的影响。

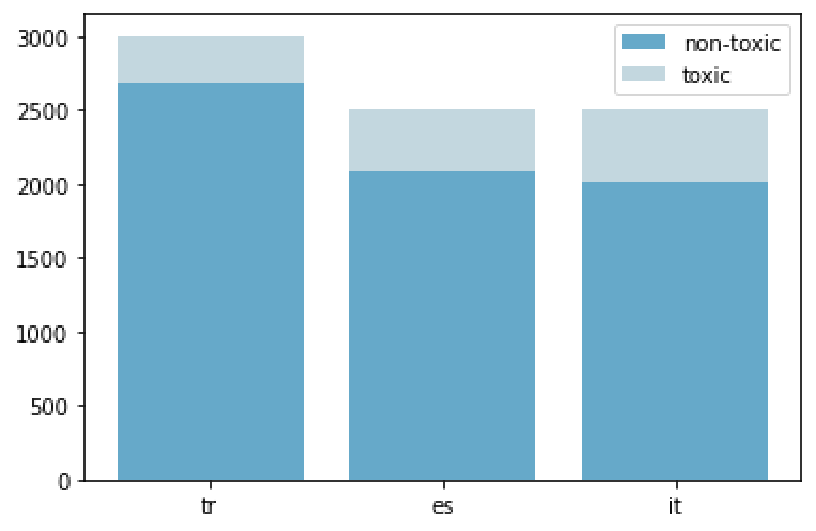
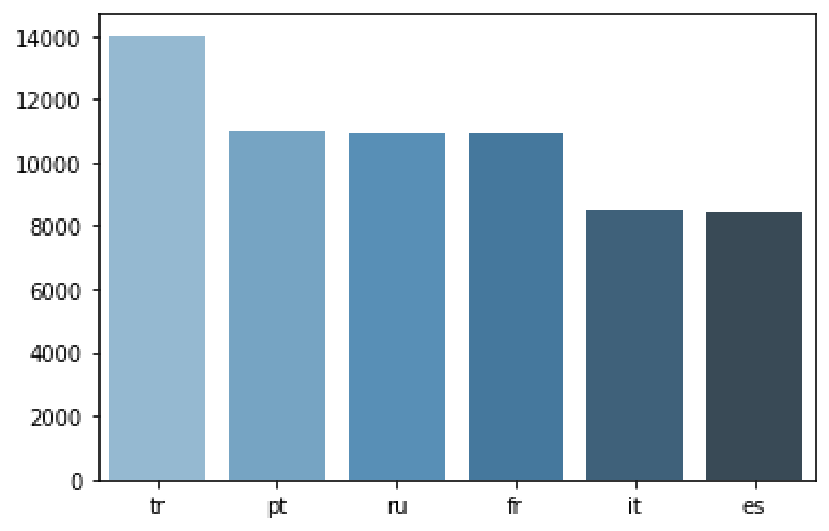
 

图1 训练集标签分布 图2 测试集标签分布

接着进一步分析无毒评论和有毒评论可能存在的一些差异。我们分别就评论文本构建单词数、大写字母数、特殊字符数、唯一单词数、字符数、停用词数六个变量，绘制箱线图，结果如图3所示。然而结果发现无毒评论与有毒评论在各个方面均无显著差异，说明两种评论的文本情况较为相似。

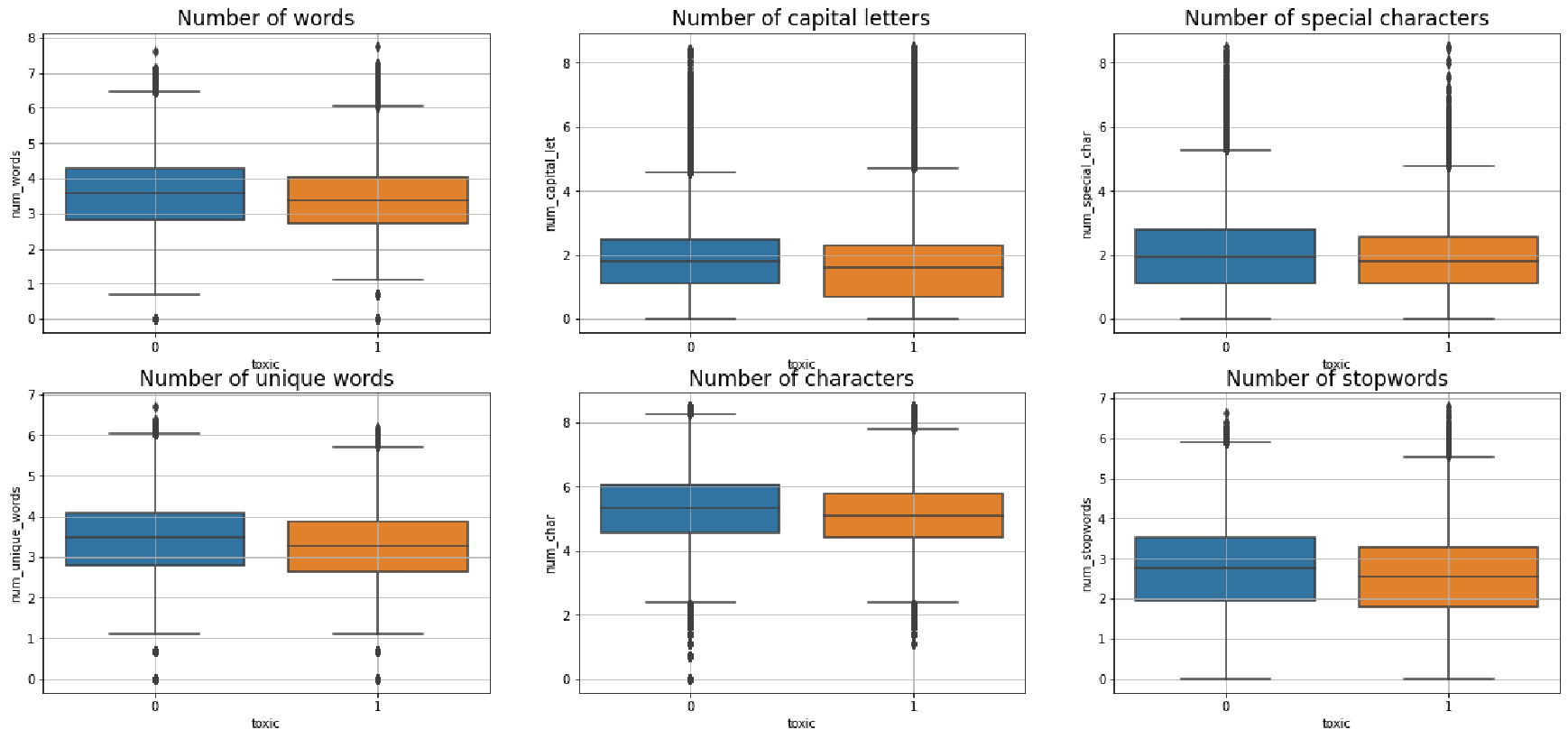


图3 评论标签分布箱线图

为进一步探索文本内容中可能存在的差异，我们分别对有毒评论和无毒评论绘制词云图，结果如图4、5所示。观察词云图的分布发现，有毒评论与无毒评论的高频词汇显著不同。在有毒评论中我们可以明显看到一些贬义词汇，如fuck、stupid等，而无毒评论的高频词汇则多为中性词。

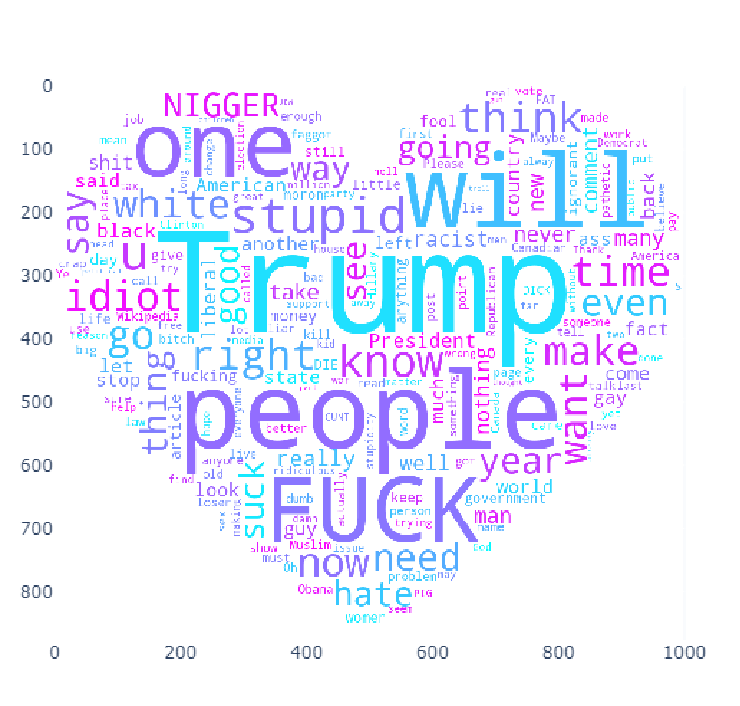
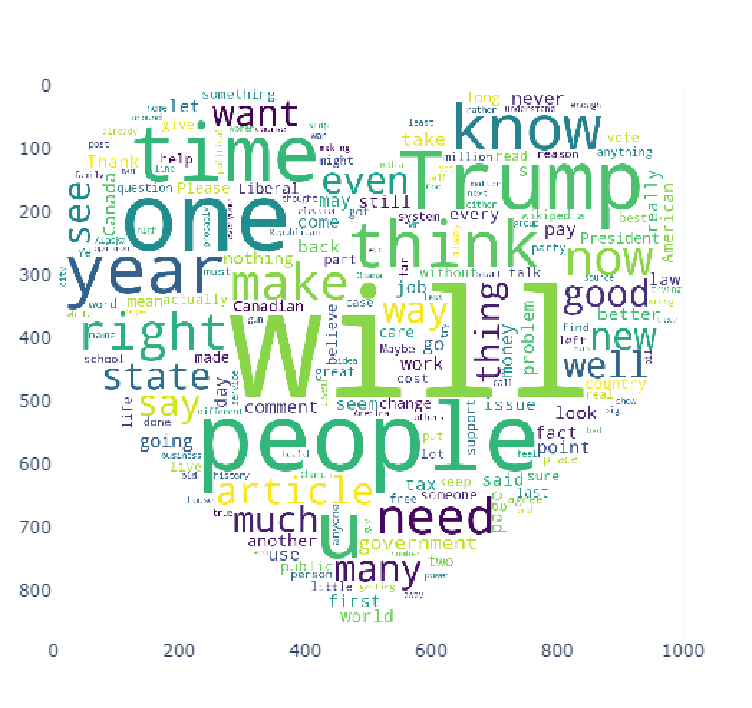
 

图4 有毒评论数据词云图 图5 无毒评论数据词云图

* 1. **数据预处理**

为实现数据特征提取与建模，我们对数据进行简单清洗与处理。主要包括数据大小写转换、单词缩写处理、噪声删除，如删除评论文本中的URLs、HTML、Non-ASCI、符号、表情符号及其他图形字符等特殊符号、标点符号删除。通过数据清洗得到纯文本数据信息。接着我们对评论文本进行分词、删除停止词、提取词干、标注词性等处理，得到格式化数据信息。

1. **模型初步效果**
   1. **Word2Vec+knn/fasttext**

我们对于评论是否有毒进行了初步的建模预测。首先我们使用了初步的机器学习模型，进行较为基础的嵌入和拟合工作。

**2.1.1 Word2vec**

Word2vec是经典的将非结构化的数据转化成结构化的数据的方法。它从大量文本语料中以无监督学习的方式学习语义知识，被广泛应用到自然语言处理中。

我们首先对英文语料进行了一定的预处理，去除句子的标点符号，去除停用词，将字符全部转化为小写，同时消除时态及词根影响，仅保留词根，最后进行分词。这样处理之后的数据就已经将语义以外的可能影响模型训练的因素降到最低了。

之后我们采用skip-gram的方法训练模型，即通过中间词预测上下文可能出现的词。模型的示意图如下：

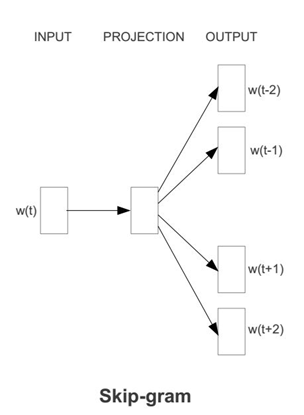


图6 Skip-gram模型示意图

由于word2vec是一个无监督学习模型，所以在进行嵌入后还需要其他的模型来对嵌入得到的向量进行建模预测。由于数据集庞大，所以我们选用了训练速度较快的knn模型来进行训练预测。我们对于训练集，使用word2vec训练所得的词库，使用算术平均，算出句子的向量表之后再使用knn算法。经过调试，发现k = 6时模型结果最好，此时在训练集上，模型的预测准确率(accuracy)达到0.86，精确率（precesion）达到0.95，召回率却只有0.45。我们认为这是因为训练集存在较为严重的样本不均衡导致的。由于在实际的应用场景中，分类模型需要挑出“有毒”的评论，所以召回率较低的模型使用价值非常有限。

**2.1.2 Fasttext**

接着我们使用了fasttext模型进行了模型的训练和拟合。

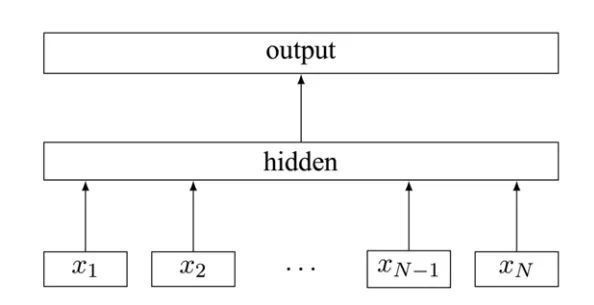


图7 fasttext原理示意图

Fasttext的模型结构和word2vec中的cbow如出一辙，但同时fasttext中也增加了n-gram特征，这使得fastttext可以关注到词序信息。所谓n-gram，即是将词语放到字符粒度去分析，这样可以关注到字符的组合所表达的含义。这就让模型对于低频词的学习效果提高了不少。同时fasttext还具有训练速度快，精度高，不需要预训练词向量等等优点。我们使用fasttext,将n-gram设置为2。最终在训练集上，fasttext的表现全面超过了word2vec+knn的组合。下表是fasttext和word2vec+knn的训练结果对比：

表1 fasttext和word2vec+knn模型效果对比

|  |  |  |
| --- | --- | --- |
|  | word2vec+knn | fasttext |
| Accuracy | 0.86 | **0.9** |
| Precision | 0.95 | **0.95** |
| Recall | 0.45 | **0.63** |

由上表可以看出，fasttext的表现要明显优于word2vec+knn，但是召回率仍然不理想，只有0.63。可见数据集的样本不均衡问题较为严重。对于这部分的处理我们将放到第三部分详述，这里只介绍机器学习部分我们作出的改进。

**2.1.3使用集成学习的方法改进**

这一部分我们主要是使用了集成学习的方法对于机器学习模型进行改进。我们主要使用imblearn.ensemble中的方法进行集成学习的训练。主要使用下采样方法和bagging方法对样本进行处理，再使用集成学习的方法进行模型训练。具体的方法不再详述，以下是模型训练的结果：

表2 ensemble方法模型效果对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | EasyEnsemble | RUSboost | BalancedBagging | BalancedRandomForest |
| Accuracy | 0.86 | 0.87 | **0.88** | 0.88 |
| Precision | 0.67 | 0.7 | **0.77** | 0.71 |
| Recall | 0.86 | 0.85 | 0.75 | **0.87** |

由上表可以看出，集成学习给模型带来的增益巨大，召回率最高可以达到0.87.虽然总体的accuracy和precision比之前的模型略有下降，但是对于此数据集的应用场景来说是更加有效的模型。

* 1. **神经网络探索**

在初期的神经网络探索中，我们分别使用了Bi-LSTM，Text-CNN，SimpleRNN和Bi-LSTM+Attention四个模型对分类效果进行简单探索，以fpr<0.05为阈值选取标准选取阈值，结果如表3所示：

表3 四个分类模型的效果对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bi-LSTM | Text-CNN | SimpleRNN | Bi-LSTM+Attention |
| Accuracy | 0.91 | **0.92** | 0.91 | 0.91 |
| Precision | 0.84 | **0.85** | 0.84 | 0.84 |
| Recall | 0.80 | **0.82** | 0.79 | 0.80 |
| AUC-value | 0.95 | **0.97** | 0.94 | 0.96 |
| Threshold | 0.060 | 0.028 | 0.019 | 0.167 |

可以看到虽然在测试集上神经网络的效果要好于传统的机器学习算法，但是由于样本的严重不均衡问题，导致选取的阈值非常小，且准确率也不能很好地评价我们的模型。由于任务是识别有毒评论，那么任务中的Precision和Recall就应该更受重视。最好的Text-CNN模型上的指标还远远不足以满足我们的需求，模型还有待改进。

* 1. **BERT base pretrained**

BERT预训练模型由Google发布我们选择了其中的BERT base pretrained模型，其大小为：12-layer, 768-hidden, 12-heads, 110M-parameters。我们对模型的结构作了可视化，图8、图9与图10分别为BERT的输入层、中间层与输出层的结构。

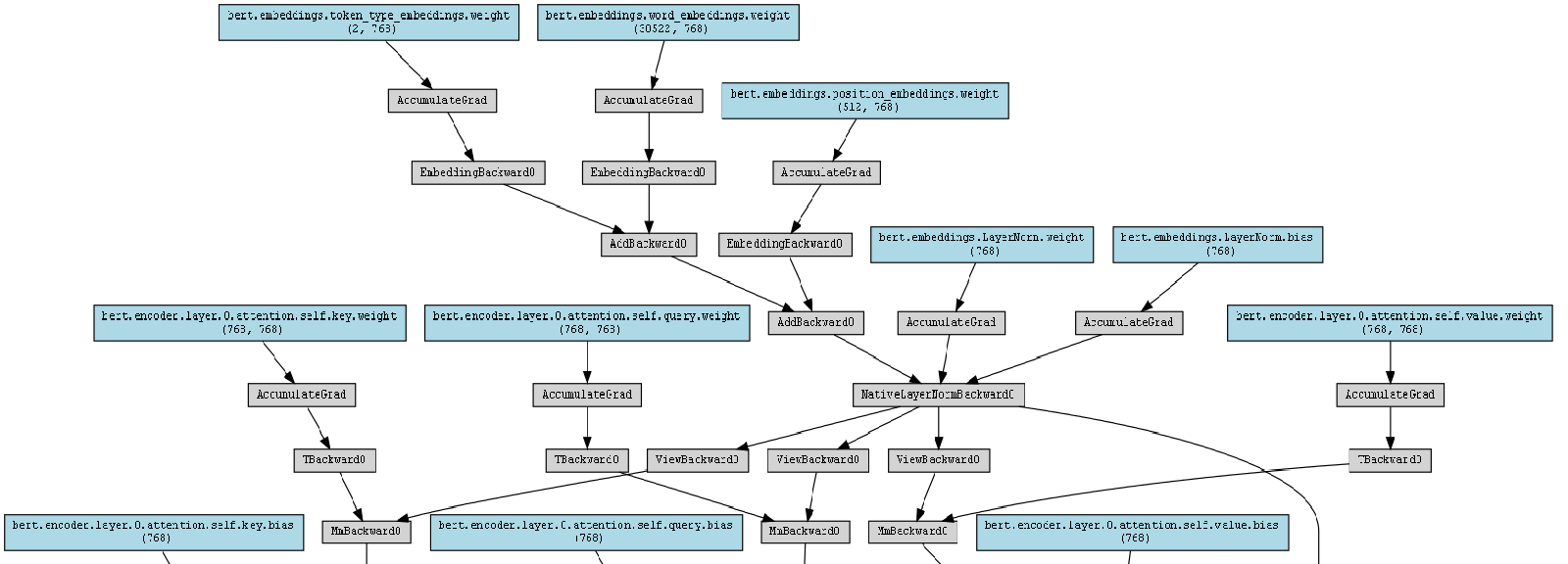


图8 BERT模型输入层结构

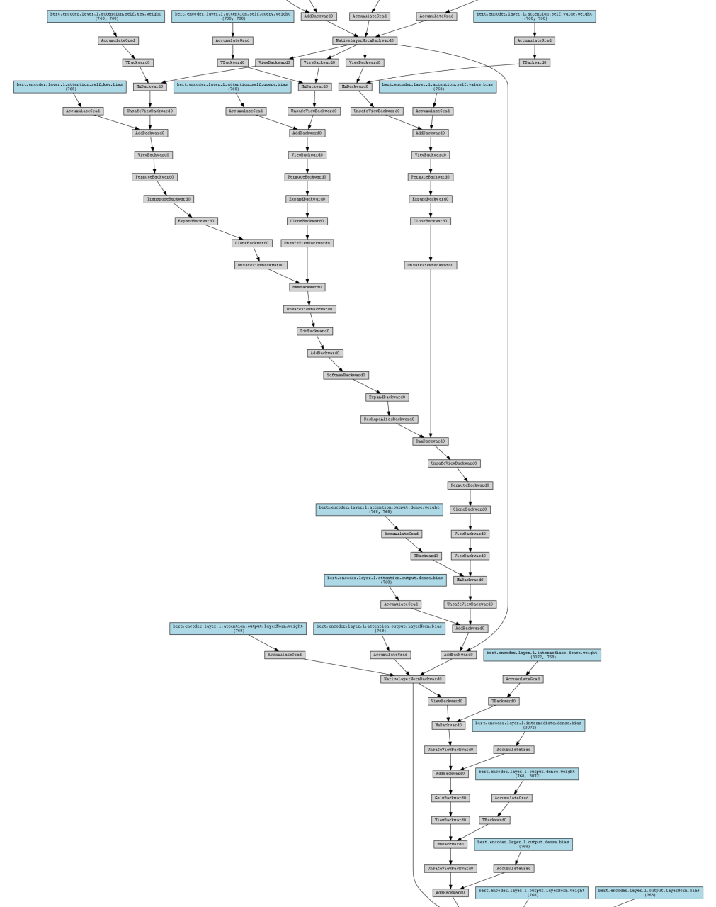


图9 BERT模型中间层结构

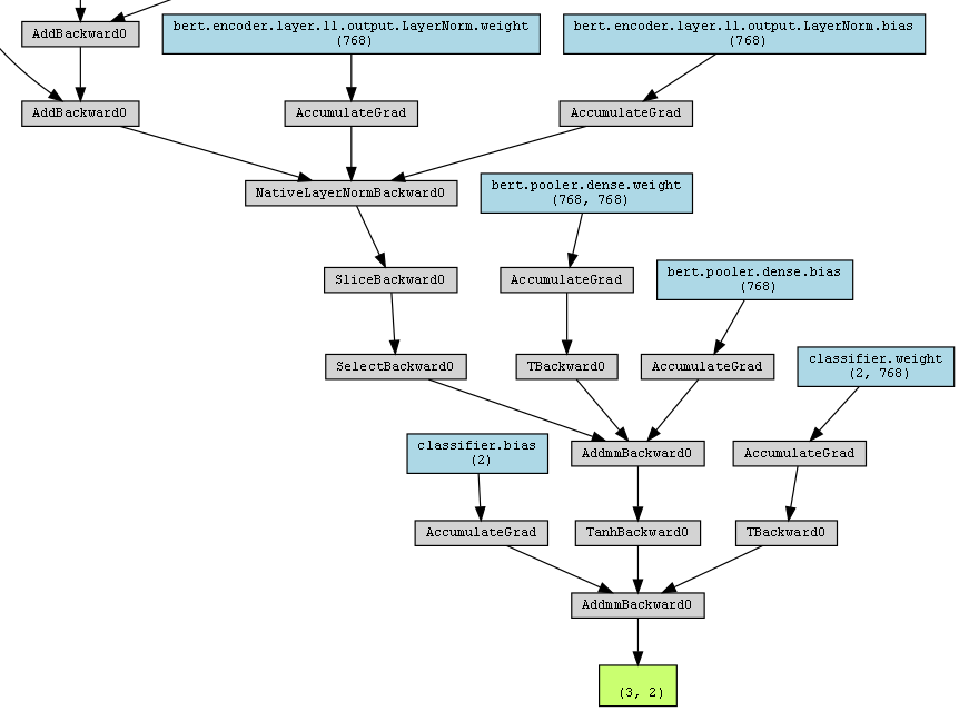


图10 BERT模型输出层结构

最后将BERT应用于有毒评论的分类任务，在测试集上准确率为0.929，召回率为0.715，效果良好。

1. **文本失衡处理**
   1. **文本增强**

为了解决文本失衡问题，首先我们对训练集的有毒评论进行文本增强，通过同义词替换、随机删除、随机交换、随机插入四种方法生成新样本。

例句：Hell, no. I ain't serving the war, I was helping people, This is fucked up.

同义词替换：Hell, no. i ain't serving the war, i was helping people, This is know up.

随机删除：ain't serving the was people, This is fucked.

随机交换：Hell, no. is ain't serving the war, I was helping people, This I fucked up.

随机插入：Hell, cost no. I ain't serving the war, I was helping people, This is fucked up.

文本增强完成后使用BERT进行评测，结果测试准确率与召回率均下降了1%左右，效果不理想。

* 1. **SeqGAN**

鉴于有毒评论数量太少，我们考虑采用生成模型来生成更多有毒评论，为此我们采用了应用于文本生成的SeqGAN模型，分别用GRU作为生成器（如图11），CNN作为判别器（如图12）。将问题定义为文本续写任务，将所有有毒评论连接成为一个有毒评论文本，让SeqGAN在此基础上续写文本。

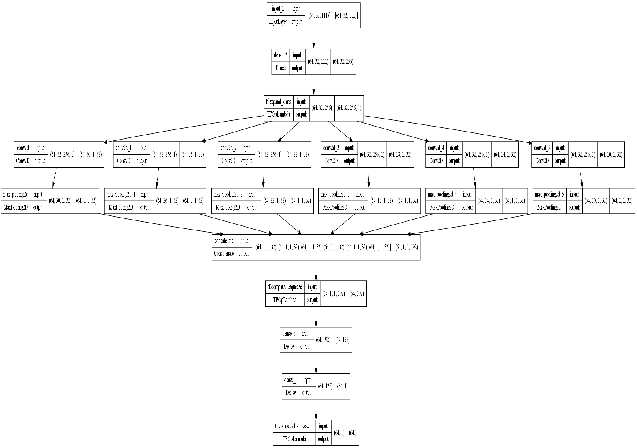
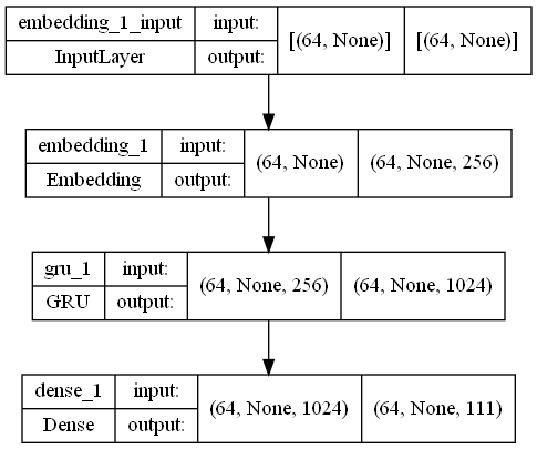


图11 SeqGAN的生成器 图12 SeqGAN的判别器

应用于此数据集后，由于不同评论并非一人发表，风格差异较大，因此拼接后的文本并没有明显的风格，生成效果并不好，因此不采用此方案。

* 1. **过采样**

过采样方面我们采用了随机过采样，SMOTE，ADASYN和SVMSMOTE四种过采样策略尝试解决样本不均衡问题。

针对不平衡数据, 最简单的一种方法就是生成少数类的样本, 这其中最基本的一种方法就是: 从少数类的样本中进行随机采样来增加新的样本；SMOTE方法就是对于少数类样本a,随机选择一个最近邻的样本b,然后从a与b的连线上随机选取一个点c作为新的少数类样本;ADASYN方法关注的是在那些基于K最近邻分类器被错误分类的原始样本附近生成新的少数类样本；SVMSMOTE方法就是在SMOTE的基础上，使用支持向量机分类器产生支持向量然后再生成新的少数类样本。

我们随机抽取20%的样本子集进行过采样，使用Text-CNN模型进行测试，在已划分好的测试集上进行评估，结果如表4所示：

表4 过采样结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 随机过采样 | SMOTE | ADASYN | SVMSMOTE |
| Accuracy | **0.89** | 0.86 | 0.86 | 0.86 |
| Precision | **0.83** | 0.79 | 0.79 | 0.80 |
| Recall | **0.71** | 0.57 | 0.57 | 0.60 |
| AUC-value | **0.93** | 0.88 | 0.88 | 0.89 |
| F1-Score | **0.76** | 0.66 | 0.67 | 0.68 |

过采样方法需要重复训练近邻分类器和SVM分类器，训练时间长，对模型没有提升效果，并不能解决我们关注的问题。效果不好的原因可能在于：过采样方法是在样本子集上进行训练的，但即使是子集上的耗时也超出了我们承受的范围；过采样方法是无监督学习的方法，依赖于近邻分类方法和SVM等机器学习方法的有效性，而有毒句子的风格可能差异过大，使得过采样方法难以取得很好的效果。

* 1. **欠采样**

欠采样方面我们采取了ClusterCentraids,RandomUnderSampler,自助法RandomSampler,EditedNearestNeighbours和重复EditedNearestNeighbours方法进行欠采样。

ClusterCentraids方法以K-Means算法的中心点来代替一个类别，从而实现欠采样；RandomUnderSampler和自助法RandomSampler即随机从数据集中选取数据，或以自助法生成新的平衡数据集；EditedNearestNeighbours应用最近邻算法来编辑(edit)数据集, 找出那些与邻居不太友好的样本然后移除。对于每一个要进行下采样的样本,那些不满足一些准则的样本将会被移除;他们的绝大多数或者全部的近邻样本都属于同一个类,这些样本会被保留在数据集中；重复EditedNearestNeighbours就是在EditedNearestNeighbours基础算法上进行多次重复。

在训练全集上进行欠采样，在已划分好的测试集上进行评估，结果如表5所示：

表5 欠采样结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ClusterCentraids | RandomUnderSampler | 自助法RandomSampler | EditedNearestNeighbours | 重复EditedNearestNeighbours |
| Accuracy | 0.78 | **0.91** | **0.91** | 0.89 | 0.89 |
| Precision | 0.64 | 0.84 | 0.84 | **0.95** | **0.95** |
| Recall | 0.28 | 0.78 | 0.78 | **0.91** | 0.90 |
| AUC-value | 0.76 | **0.95** | **0.95** | 0.93 | 0.93 |
| F1-Score | 0.39 | 0.81 | 0.80 | **0.93** | **0.93** |

可以看到EditedNearestNeighbours和重复EditedNearestNeighbours方法在模型提升的效果是最好的。虽然牺牲了一些准确度，但是在Precision和Recall这两项我们关注的结果上达到了非常好的成绩。

* 1. **过采样和欠采样结合**

在之前的SMOTE方法中,当由边界的样本与其他样本进行过采样差值时,很容易生成一些噪音数据。因此,在过采样之后需要对样本进行清洗。通过过采样和欠采样结合，可能达到更好的效果，我们采用了SMOTEENN和SMOTETomek这两种方法进行过采样和欠采样的结合。

使用子集训练，在已划分好的测试集上进行评估，结果如表6所示：

表6 过采样和欠采样的结合

|  |  |  |
| --- | --- | --- |
|  | SMOTEEN | SMOTETomek |
| Accuracy | 0.73 | **0.84** |
| Precision | 0.35 | **0.95** |
| Recall | 0.08 | **0.85** |
| AUC-value | 0.63 | **0.83** |
| F1-Score | 0.13 | **0.90** |

过采样和欠采样的结合并没有如预期那样取得更好的结果，相反效果变差了，在之后的研究中我们不考虑这两种方法。

1. **主题提取**
   1. **数据说明**

前面我们给出有毒评论的定义，并将其分为obscene、threat、insult、identity hate四种类型，进一步我们对有毒评论进行主题提取，尝试给出其含义。

首先我们对有毒评论数据进行简单分析，绘制标签分布条形图，如图13所示。观察图形发现样本标签数量差异较大，有毒评论中，Obscene和Insult类型的评论数较多，而threat和identity hate类型评论数较少，接着我们分析了每条评论的标签，对每条评论所含标签数量绘制条形图，结果如图14所示。通过图形可以看到大部分评论都同时含有三条及以上标签，说明四种类型的毒性边界并不明显。

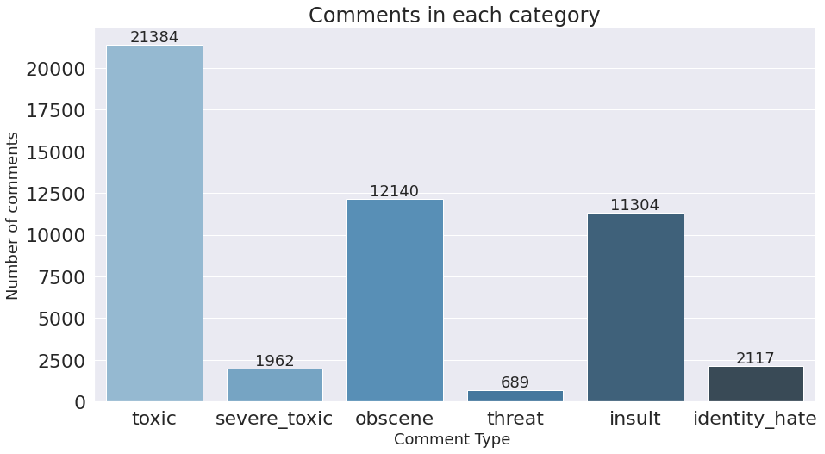
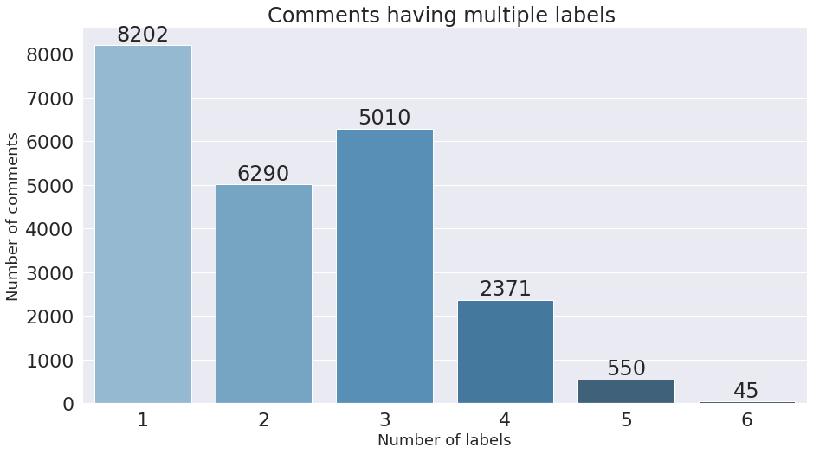
 

图13 测试集毒性标签分布 图14 单条评论标签数量分布

接着，我们分别对四种毒性评论绘制词云图，结果如图15所示。从图中我们可以看到threat和identity hate的关键词与其他两类差异较为明显，而Obscene和insult的关键词重叠较多。



图15 毒性评论词云图

* 1. **主题提取**

首先，我们对有毒评论进行LDA主题提取，通过最小化损失确定主题数为4，通过LDA主题建模，得到各个主题的特征词汇，结果如图16所示。观察主题提取结果，发现主题1和3的距离较近，也符合前面分析结果。

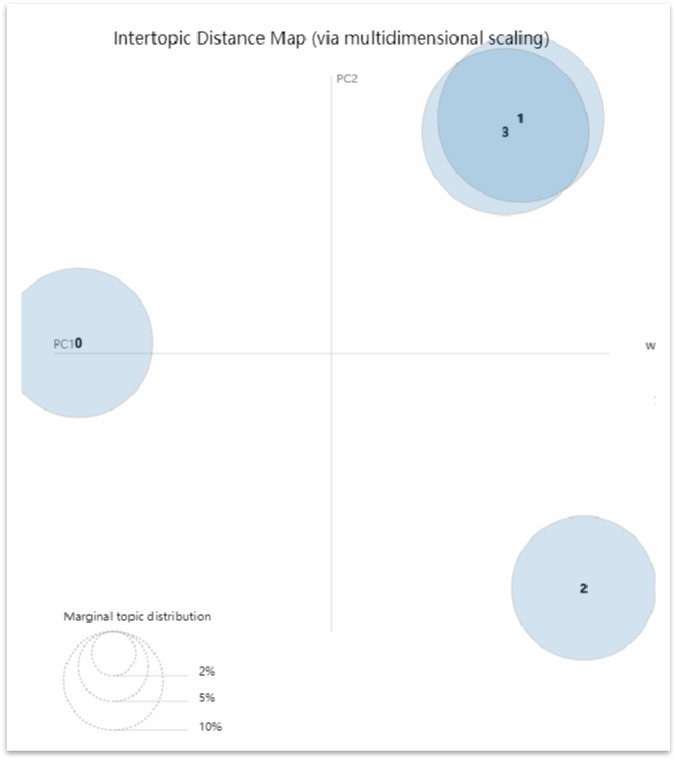


图16 LDA主题提取可视化

接着我们利用BERTopic进行主题提取，BerTopic使用BERT嵌入和基于类的TF-IDF来创建密集集群，生成主题，在短文本上的主题提取效果通常优于LDA。它首先使用BERT提取文档嵌入，接着利用UMAP降低嵌入的维数，使用HDBSCAN技术聚类减少嵌入并创建语义相似文档的聚类。最后利用基于类的TF-IDF进行主题提取和精简，提高最大边缘关联词的一致性。主题距离可视化结果如图17所示，可以看到它较为明显的将四种毒性进行区分，表7展示了各个主题的含义。

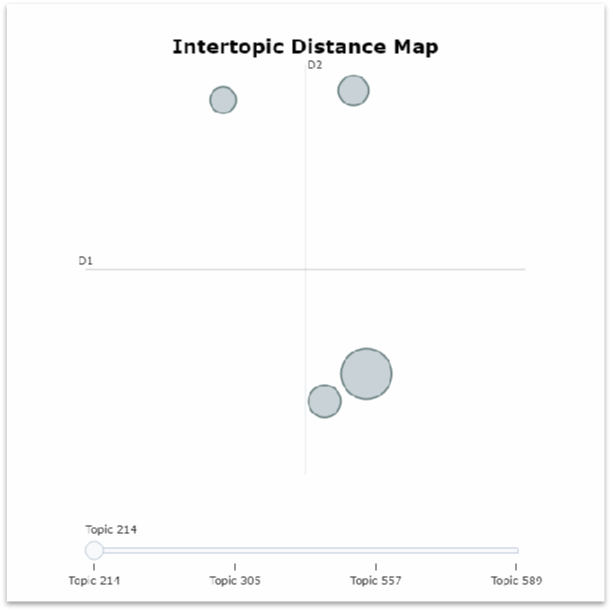


图17 BERTopic主题提取可视化

表7 BERTopic主题提取

|  |  |  |
| --- | --- | --- |
| Topic | Count | Name |
| -1 | 118716 | trump\_will\_fuck\_stupid |
| 214 | 7382 | canada\_canadian\_liberal\_trudeau |
| 557 | 2990 | bullshit\_dick\_vandal\_traitor |
| 589 | 2582 | vote\_stupid\_idiot\_voter |
| 305 | 1940 | catholic\_god\_christian\_religion |

最后我们结合多种主题提取的方法，给出各个主题的主题词和相关含义，结果如表8所示。

表8 有毒评论主题词及含义

|  |  |  |
| --- | --- | --- |
| 毒性 | 主题词 | 含义 |
| Obscene | Fuck、Fucking、Bitch | Connected with sex in a way that most people find offensive |
| Threat | Will、Die、Kill | A statement in which you tell sb that you will punish or harm them, especially if they do not do what you want |
| Insult | Fuck、Stupid、Idiot | To say or do sth that offends sb |
| Identity Hate | Nigger、Jew、Gay | Intentionally making an offensive remark about sb.’s  character or appearance in an argument |

1. **细粒度分类**

在数据集中除了有毒和无毒的分类以外，还有更细分的对有毒语言的分类。我们考虑使用这部分信息来提升模型效果。在细粒度分类上，以有毒类型进行辅助分类，进而替身模型效果。

有毒语言共有五种，分别是severe\_toxic,obscene,threat,insult,identity\_hate，加上无毒(non\_toxic)，一共六种类型。我们共提出了三种模型，即六分类，二分类，六分类+二分类。

* 1. **简单神经网络细粒度分类**

我们先使用初期效果最好的Bi-LSTM模型进行进一步分类尝试，效果如表9所示：

表9 Bi-LSTM模型细粒度分类结果

|  |  |  |  |
| --- | --- | --- | --- |
|  | 二分类 | 六分类 | 六分类+二分类 |
| Accuracy | 0.91 | 0.92 | **0.92** |
| Precision | 0.84 |  | **0.93** |
| Recall | 0.80 |  | **0.96** |
| AUC-value | 0.95 |  |  |
| F1-Score | 0.91 |  | **0.94** |

在六分类+二分类的结果上，取得了比单纯的二分类模型更好的效果，在我们关注的Precision和Recall上也取得了很大的提升，说明我们的策略是有效的。

但是六分类模型仍然存在着严重的样本不均衡问题，表10是六分类模型的测试集上的混淆举证：

表10 六分类模型混淆矩阵

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Non\_toxic | Severe\_toxic | Obscene | Threat | Insult | Indentity\_hate |
| Non\_toxic | 7249 | 236 | 66 | 9 | 28 | 5 |
| Severe\_toxic | 406 | 406 | 327 | 13 | 61 | 9 |
| Obscene | 81 | 263 | 659 | 4 | 41 | 0 |
| Threat | 5 | 11 | 5 | 4 | 2 | 1 |
| Insult | 58 | 98 | 58 | 1 | 40 | 5 |
| Indentity\_hate | 5 | 5 | 3 | 2 | 3 | 4 |

从表10中我们可以看出，因为前三类样本的数量远多于后三类样本的数量，导致模型会极大地偏向前三类样本，极端情况下会将所有测试集样本归类为前三类。

为了解决这个问题，我们尝试过六分类的过采样和欠采样和加权重的方法。但是六分类的样本不均衡问题比二分类的还要严重，过采样会导致严重的过拟合，且过采样的算法为无监督学习算法，样本间风格差异过大，难以控制样本生成的质量；加权重的方法过于主观，具有一定的偶然性。所以两种效果都不佳。

在这种情况下，我们考虑充分使用原本数据集的信息，更换更强大的模型，以提升对数据集信息的提取使用能力。

* 1. **Transformer**

对此我们选用更能提取文本信息的Transformer模型，使用Transformer的encoder，使用三层encoder进行文本信息提取和分类，同样以fpr<0.05选取阈值，结果如表11所示：

表11 Transformer模型效果

|  |  |  |  |
| --- | --- | --- | --- |
|  | 二分类（未选择Threshold） | 二分类（Threshold=0.075） | 六分类+二分类 |
| Accuracy | 0.898 | **0.916** | 0.909 |
| Precision | 0.886 | **0.950** | 0.909 |
| Recall | **0.992** | 0.939 | 0.976 |
| AUC-value | 0.95 | 0.95 |  |
| F1-Score | 0.936 | **0.945** | 0.941 |

Transformer模型对比Bi-LSTM模型有明显的进步，可以看到我们关注的Recall和Precision上明显高于Bi-LSTM，在牺牲部分准确率和情况下Recall能达到0.992.说明Transformer模型能更加充分地利用文本中的信息。

1. **语言迁移**
   1. **数据说明**

训练集使用jigsaw-toxic-comment-train.csv文件，该数据集中仅有英文文本，标签的分布如图18所示。

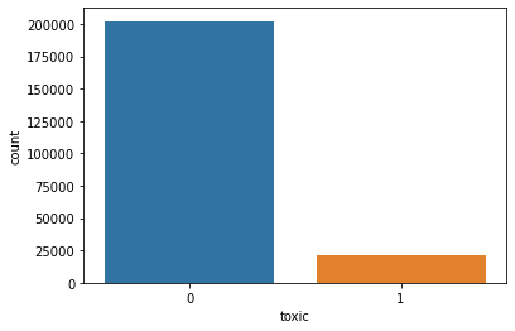


图18 训练集标签分布

测试集使用validation.csv文件，该数据集中包含西班牙语、意大利语、土耳其语三种语言，语言的样本数量分布如图19所示，标签的分布如图20所示。

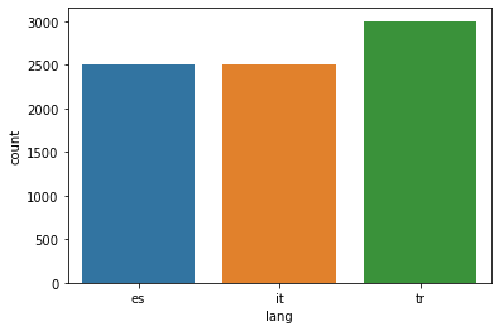
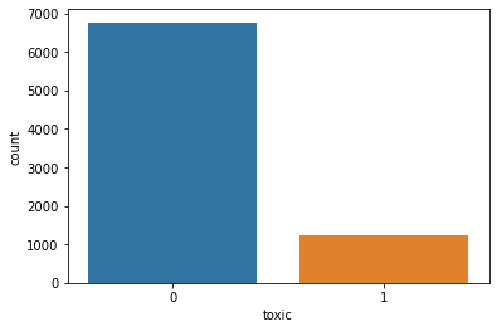
 

图19 测试集语言分布 图20 测试集标签分布

* 1. **ERNIE-M**

Ernie-M是2021年初百度发布多语言预训练模型。Ernie-M的多语言预训练模型能够识别并理解多达96种语言。模型创新性地提出了回译机制，可以从单语语料中学习语言间的语义对齐关系。该模型的词表大小多达25万，涵盖了96种语言中常用的词汇，总字符量达到1.5亿。由于本例我们的训练集语言为英语，而测试集为意大利语，西班牙语和土耳其语。所以这是一个zero-shot的训练集，也就是说模型需要通过过去对于英语语言的理解来理解之前没有训练过的语言资源。Ernie模型恰好可以实现这一点，通过在高资源语言上训练，应用到低资源语言的理解上，如下图所示：

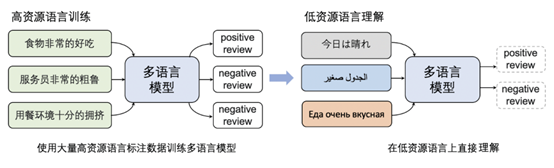


图21 跨语言学习示意图

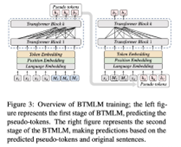
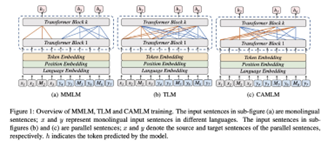


图22 Ernie-M两阶段预训练对比示意图

跨语言学习中最重要的部分就是（多组）双语语料的对齐和训练。Ernie-M主要通过两个阶段的预训练来完成这一点。

第一阶段，Ernie-M提出了基于attention机制的CAMLM任务，相比于之前的MMLM和TLM，CAMLM的训练只依赖语另一种语言的token,而不依赖于本语言的token,这样可以起到加强对齐效果的作用。

而在并行语料不足的情况下，Ernie-M使用BTMLM来直接使用单语语料学习。它通过在一种语言之后加入遮挡，然后用训练好的CAMLM来填充这些遮挡，因此就可以生成伪平行语料。我们使用的是ERNIE-M的base版本，共12层，768维，12头。

但是在训练中，使用ERNIE-M模型并没有取得预期中的目标。因为该预训练模型参数众多，导致显存占用高，batchsize设置过大则显存不足，设置过小则训练耗时长。同时因为样本的不均衡问题仍然存在，又采用了过采样的方法，导致训练集进一步增大，平均训练一个epoch耗时40多分钟，且模型收敛缓慢，训练超过8小时模型都无法收敛。最终在测试集上的Accuracy为0.67,f1score为0.26,都不是非常理想 。

* 1. **mBERT**

**6.3.1 模型结果**

mBERT于2018年由Google发布，出自大名鼎鼎的《Bert: Pre-training of deep bidirectional transformers for language understanding》，是用于实现多语言任务的预训练模型，它的想法非常简单，就是将BERT模型用多种语言进行训练，它的配置为：

104种语言

规格：BERT base

tokenize方式：WordPiece

词典：110K

语料：Wekipedia

我们在此任务中应用了mBERT，其在测试集准确率为0.77，f1-score为0.40，效果并不是很理想，mBERT可能更适合多语言任务，而非包含zero-shot的跨语言任务。

**6.3.2 模型可视化**

我们对于BERT模型的注意力机制比较感兴趣，因此对其进行了可视化工作。首先是对预测结果的可视化，我们选择了一条英文有毒评论与一条西班牙语有毒评论，分别使用mBERT进行预测，可视化结果如图23与图24所示

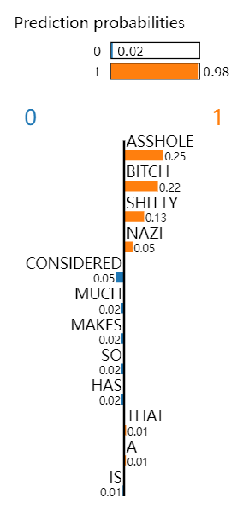
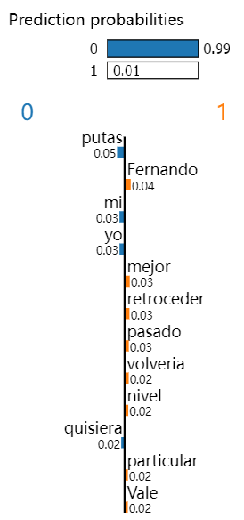
 

图23 英文评论预测结果 图24 西班牙语评论预测结果

可以看到，mBERT对于训练集上的英文评论分类效果很好，能够注意到对于是否有毒比较关键的词语，而对于西班牙评论分类较差，对于“putas”等关键词语的判断比较差。

除此之外，以相同的英文评论为输入，我们对整个模型的不同层以及不同的注意力头进行了可视化，了解其注意力机制的结果，结果如图25与图26所示（此两图仅为截图，具体可参见附件中的model\_view.html与head\_view.html），可以看到mBERT模型不仅限于记忆单词，确实可以学习到语法知识。

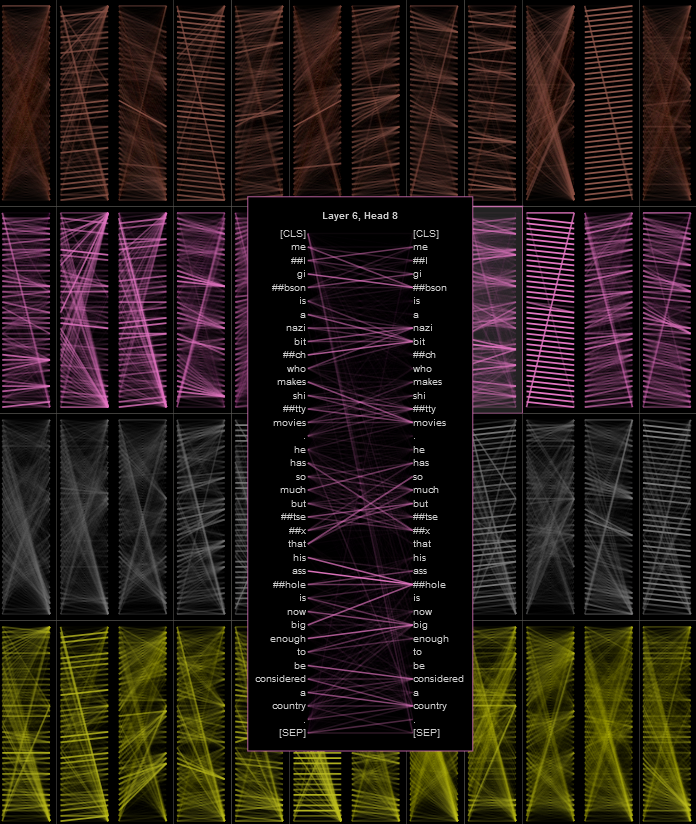
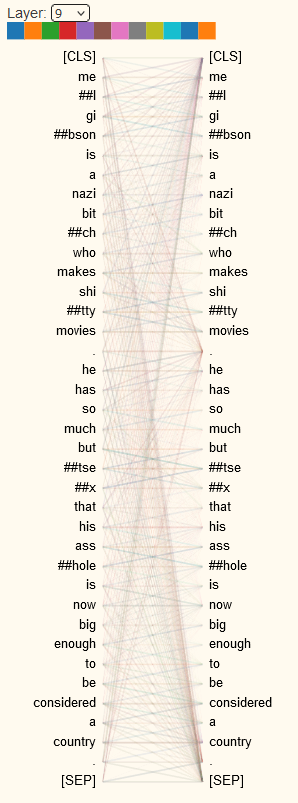
 

图25 整个模型可视化 图26 分层可视化

* 1. **XLM-RoBERTa**

XLM于2019年由Facebook发布，出自文章《Cross-lingual language model pretraining》，相较于mBERT，它专注于跨语言任务，能够比较好的解决zero-shot问题，其分词方式为BPE，词典大小为200K。

XLM预训练时使用三种目标函数，分别是在单语种语料上使用非监督的CLM(Causal Language Modeling)和MLM(Masked Language Modeling)；在平行语料上使用TLM(Translation Language Modeling)目标函数，训练方式为将平行句子拼接后随机mask，希望让模型能借助另一种语言的信息来还原出被遮蔽的词，如图27所示。

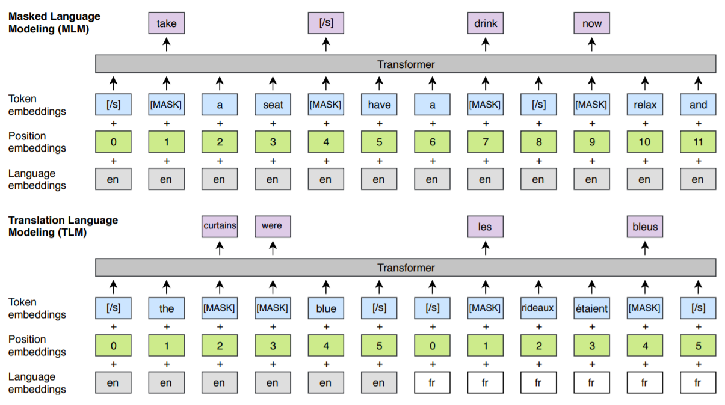


图27 MLM与TLM目标函数

同年，Facebook发布了功能更为强大的XLM-RoBERTa，出自文章《Unsupervised Cross-lingual Representation Learning at Scale》。相较于XLM，XLM-RoBERTa更换了分词方式，并且继承了RoBERTa的血统，在更大的cc100语料库上进行训练，词典大小为250K，模型去掉了Language Embedding只使用单种语料，训练目标只有MLM。

最终XLM-RoBERTa在我们的测试集上准确率达到了0.86，f1-score也达到了0.56，取得了比较好的效果。

1. 小组分工

王理琼：模型初探BERT部分（2.3）、文本失衡处理SeqGAN部分（3.2）、语言迁移mBERT与XLM-RoBERTa部分（6.3&6.4）

张振羽：模型初探fasttext与Word2Vec部分（2.1）、语言迁移ERNIE-M部分（6.2）

钟梓健：模型初探神经网络部分（2.2）、文本失衡处理过采样与欠采样部分（3.3&3.4）、细粒度分类（5）

苏晓甜：数据预处理与可视化（1）、主题提取（4）