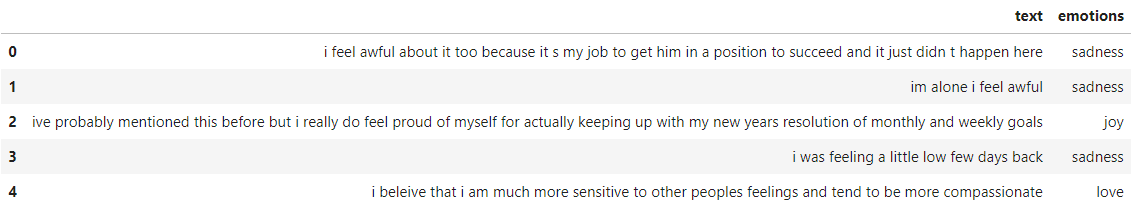
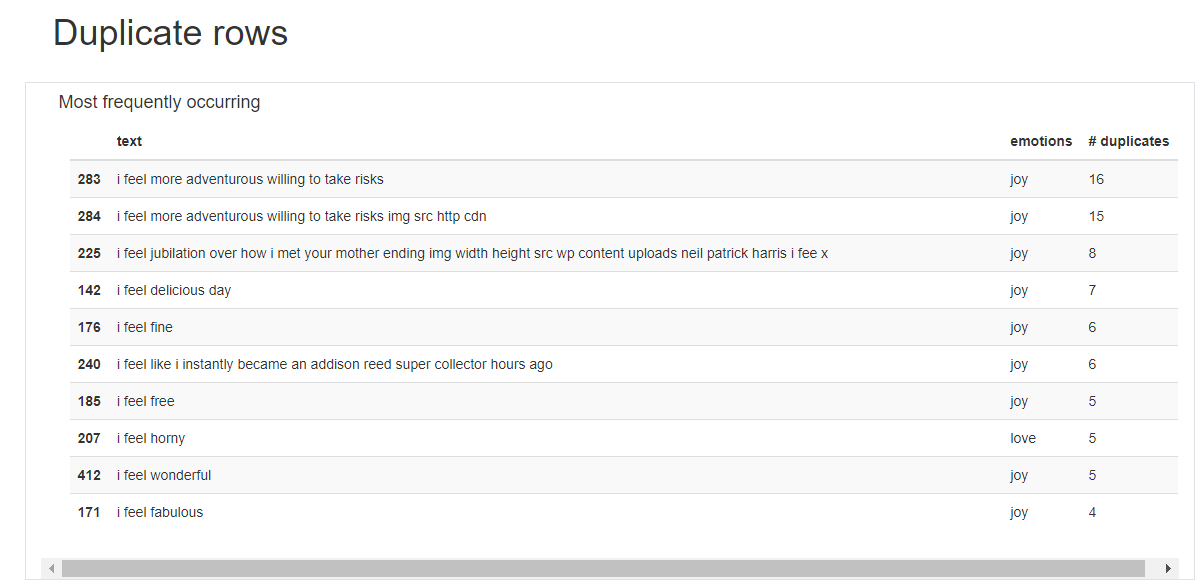
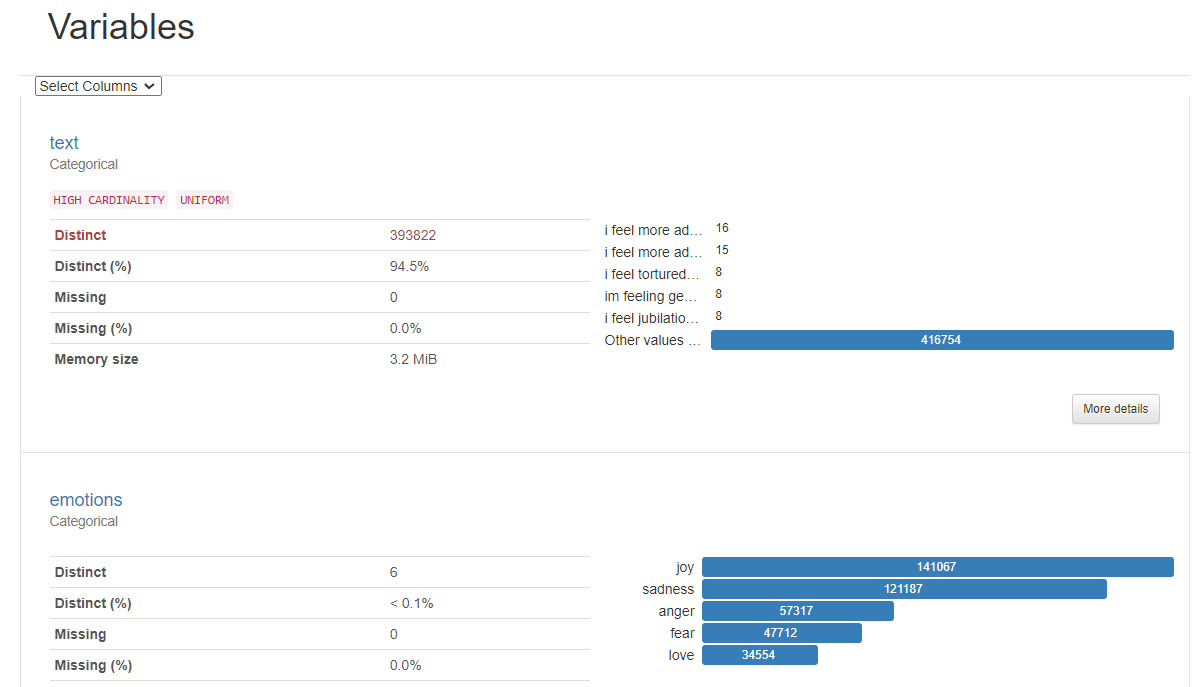
1. **数据准备与探索性特征工程**
   1. **预览与分析原始数据**

在DataFrame中预览前五行数据：

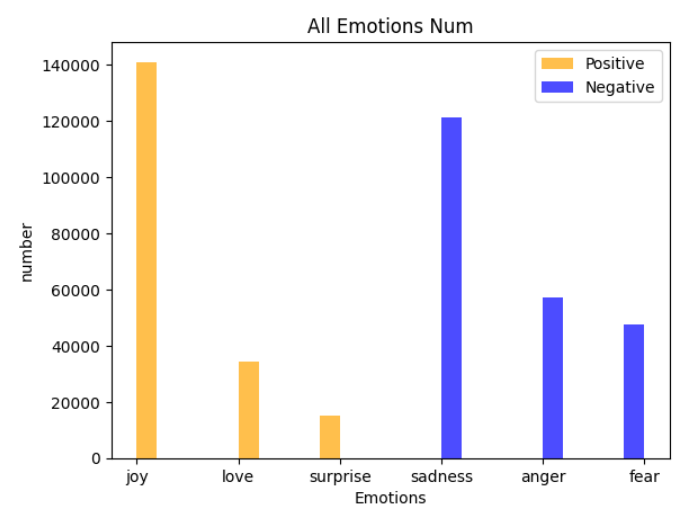


可以看到原始数据由三部分组成：ID、text和emotions标签。

对原始数据进行分析检视：



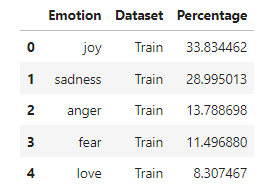
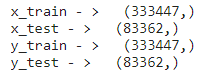
统计各类型数据分布：



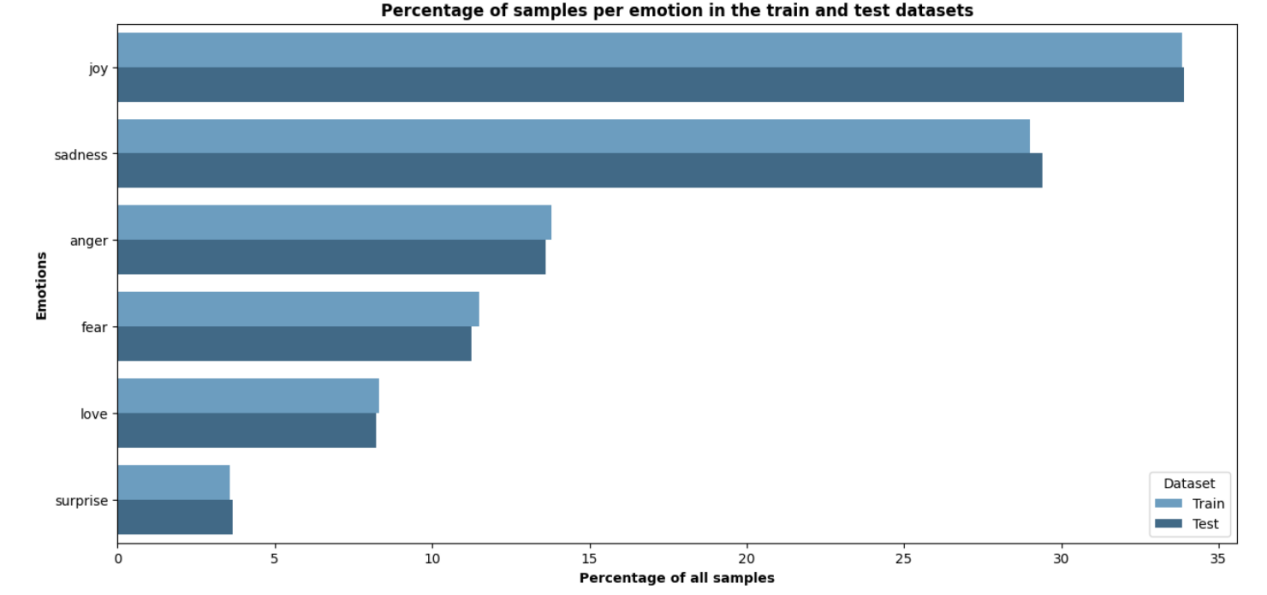
可以看出，类型数据分布较不均衡，考虑后续数据增强以平衡数据。

* 1. **划分训练集和验证集：**

依照各个类型的分布，均衡划分出训练集和验证集：



统计分布可视化：



1. **机器学习分类模型**

**2.1 数据预处理**

去除文本中的标点、emoji、特殊字符等，同时所有大写全部转为小写。清洗后数据展示：



**2.2 数据增强**

SMOTE算法进行类型样本过采样

1. from sklearn.pipeline import make\_pipeline
2. from imblearn.over\_sampling import  SMOTE
3. *#过采样*
4. smt = SMOTE(random\_state=777, k\_neigShbors=1)
5. X\_SMOTE, y\_SMOTE= smt.fit\_resample(vec\_fit,train.text)
6. from collections import Counter
7. *#样本已被扩充*
8. print(Counter(y\_SMOTE))

过采样后的样本分布更为均衡。

**2.2 词向量化与特征提取**

对于传统的机器学习任务，模型训练的输入必须为给定维度的特征向量。如何有效地提取文本特征信息成为了问题的关键所在。常见的四种特征提取方法有：

（1）Bag of Words

（2）LDA

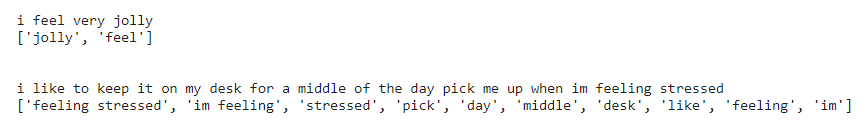
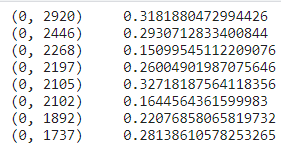
（3）Average Word Vectors

（4）TF-idf

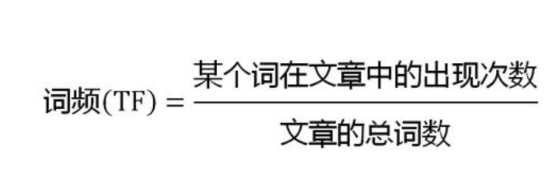
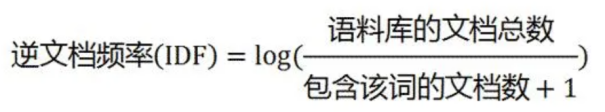
其中，法（3）是将文本中每个分词利用预训练模型先进行向量化表示，再将文本中各个分词的向量相加取平均，这种方法较为复杂，且取平均对每个token的特征损失较大，在这里先不予采用；法（1）（2）（4）的想法则十分简单朴素，通过统计词汇在文本中的出现词频等数学统计量，来统计并提取中文本中最为关键的那些分词，作为该文本的特征向量。我们在这里主要使用TF-idf方法。

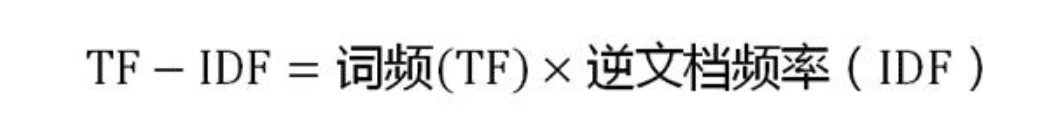
**2.2.1 TF-IDF**

提取句子中的unique words（最能体现文本有效信息的分词/词组），效果展示如下：



其中unique words用0-50000的编码表示，同时每个编码有不同的值，代表该词的TF-IDF特征，计算公式如下：



词频代表该单词在文本中的出现频率，TF较高的单词可能暗示了文章主题，也可能是常用词（如this that is等）；IDF是逆文本频率，单词在语料库中出现频率越少，IDF越高，比如专业术语Alzheimer（阿尔茨海默）就可能被赋予较高的IDF，这种词往往暗示了文章主题，但也有可能仅仅是生僻词而对分类任务价值不大。

TF-IDF综合了词频和逆文本频率的特征，TF-IDF的值越大的单词，用来作为分类特征的价值就越高。

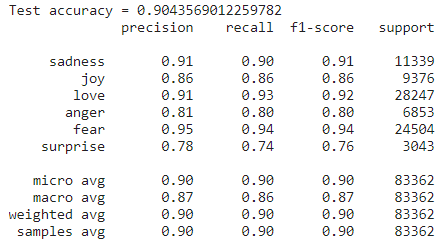
**2.3 SVM支持向量机模型**

**2.3.1模型搭建**

使用LinearSVC分类器完成二分类任务，代码如下：

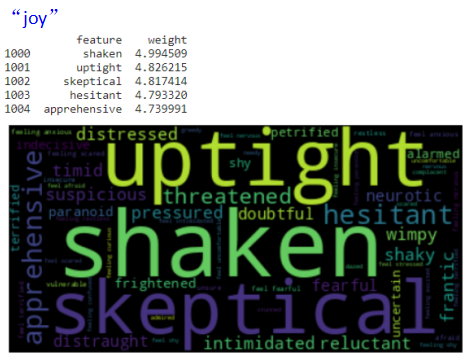
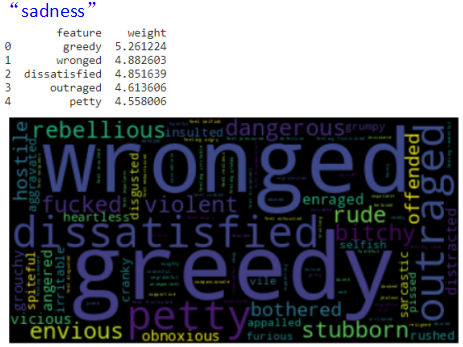
1. from sklearn import svm
2. from collections import Counter
3. model\_svm=svm.LinearSVC()
4. model\_svm.fit(xtrain\_d2v, y\_train)
5. *# Making predictions*
6. model\_svm\_preds = model\_svm.predict(xtest\_d2v)
7. model\_svm\_preds = pd.get\_dummies(model\_svm\_preds)
8. model\_svm\_preds.head(5)
9. emotions = ['sadness', 'joy', 'love', 'anger', 'fear', 'surprise']
10. *# Check test accuracy*
11. test\_accuracy = accuracy\_score(one\_hot\_test, model\_svm\_preds)
12. print('Test accuracy = {}'.format(test\_accuracy))
13. *## Show results*
14. print(classification\_report(one\_hot\_test, model\_svm\_preds, target\_names=emotions))

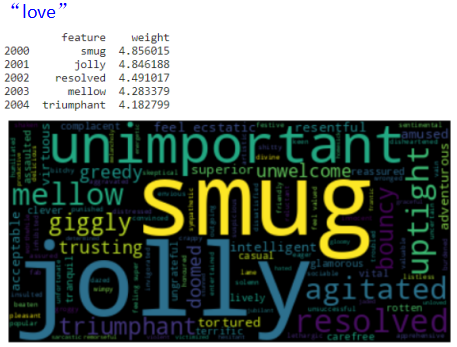
**2.3.2 分类结果**



**2.3.3 结果可视化**

分析SVM中各特征矩阵的权重，对每个标签构建下列词云，其中单词越醒目代表权重越高：

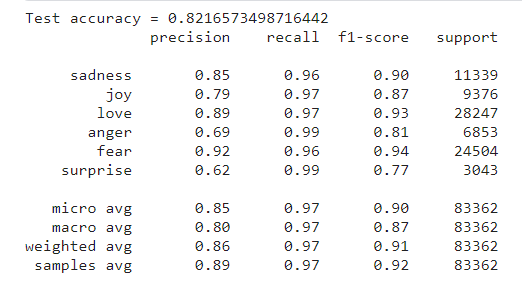




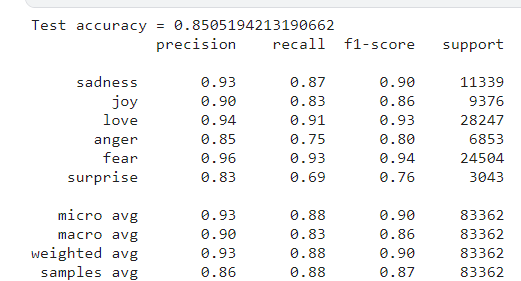
可以看出虽然各种情感标签的关键词有重复的部分，但同时也有所区别，这些关键词的出现会对我们判断一条文本为何种情感起到重要的作用。

**2.4 其他常见分类机器学习模型比较**

**2.4.1 Ridge Classification**



**2.4.2 Logistic Classification**



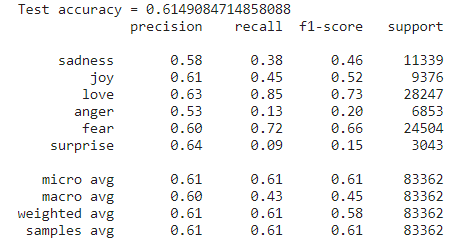
**2.5 改进创新**

TF-IDF特征提取是对文本单个分词的概率统计，并未考虑到分词在文本中的位置及其上下文关系。据此，我们尝试将一段文本或一个句子整体向量化表示（考虑上下文关系，同义词关系等），以期取得更好的效果。

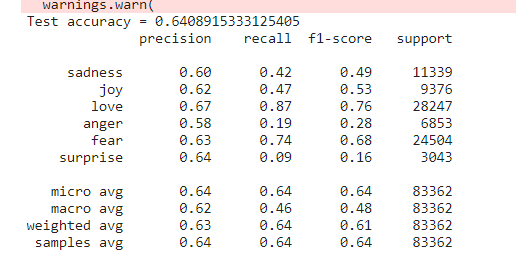
**2.5.1 手写Sentence Embedding**

在Word2Vec的向量化基础上，考虑到该方法没有引入token与token直接的位置信息，我们将语句中的每个token vector加权取平均之后，再在每个维度上添加正余弦位置信息。具体的位置信息编码方式我们参考了transformer的position embedding方法，利用周期性函数，可以较完美的体现出上下文关系，又不会过度影响本身的语义内涵。

这里展示不添加position embedding的结果：



添加position embedding后结果：



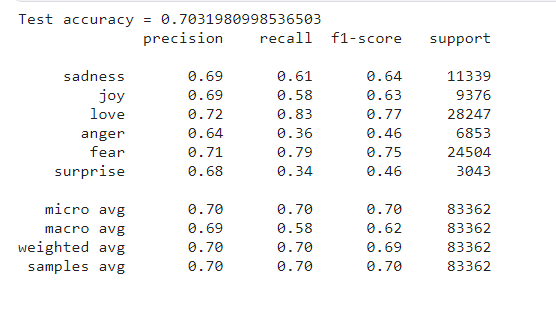
通过对比可以看到，包含位置编码信息的语句向量分类准确率更高，效果更好；说明手写sentence embedding改良方案初步成功。

**2.5.2 SentenceBert**

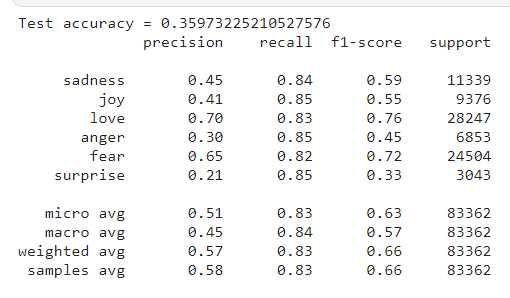
Bert模型已经在NLP各大任务中都展现出了强者的姿态。在语义相似度计算（semantic textual similarity）任务上也不例外，但是，由于bert模型规定，在计算语义相似度时，需要将两个句子同时进入模型，进行信息交互，这造成大量的计算开销。

而改进后的SentenceBert模型，借鉴了孪生网络思想，将不同的句子输入到两个bert模型中（但这两个bert模型是参数共享的，也可以理解为是同一个bert模型），获取到每个句子的句子表征向量。从而更好地适应更窄的下游任务，使语义空间更为紧密。

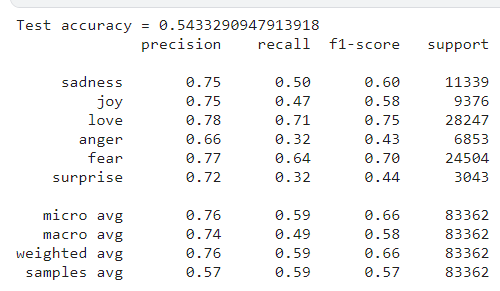
Svm：



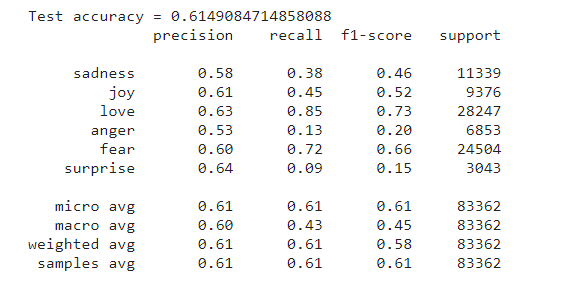
Ridge：



Logistic：



**2.5.3 Doc2Vec**



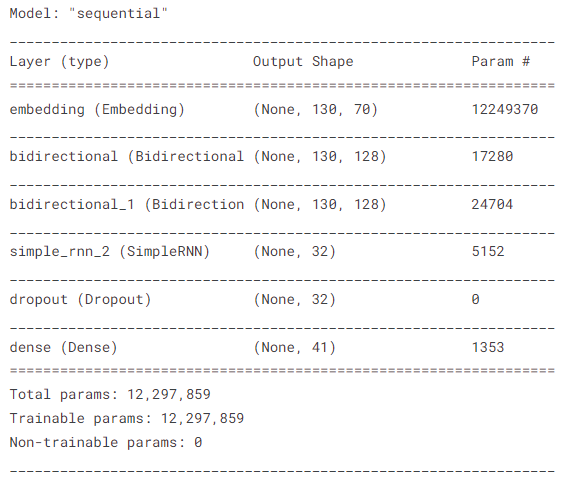
1. **深度学习分类模型**

有别于传统的机器学习模型，深度学习无需人为提取样本的特征信息；而是“端到端”自主学习样本特征，从而更好的学习这些向量的特征。常见的深度学习模型例如RNN、LSTM、Bert等，都在NLP任务上取得了良好的成绩。

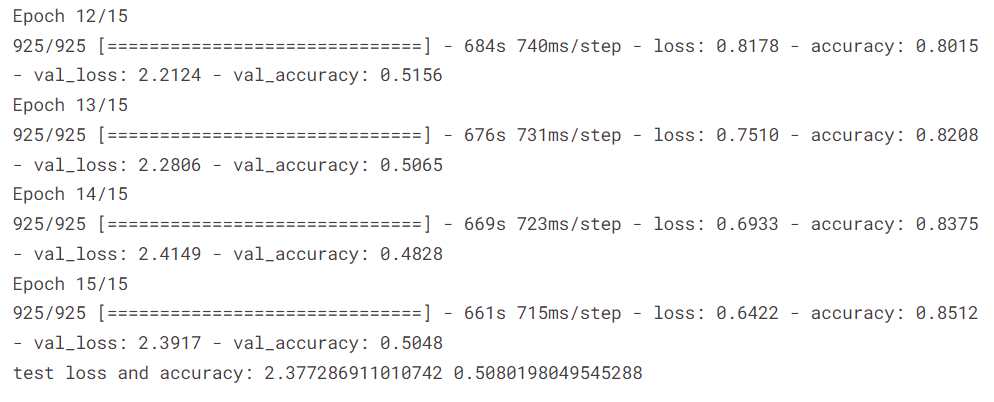
**3.1 SimpleRNN**

先从最简单的循环神经网络RNN开始。

模型参数架构：



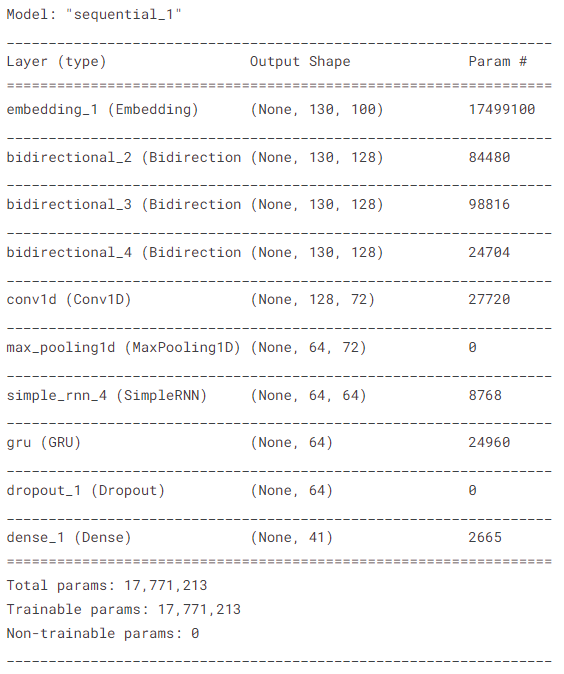
预测结果：



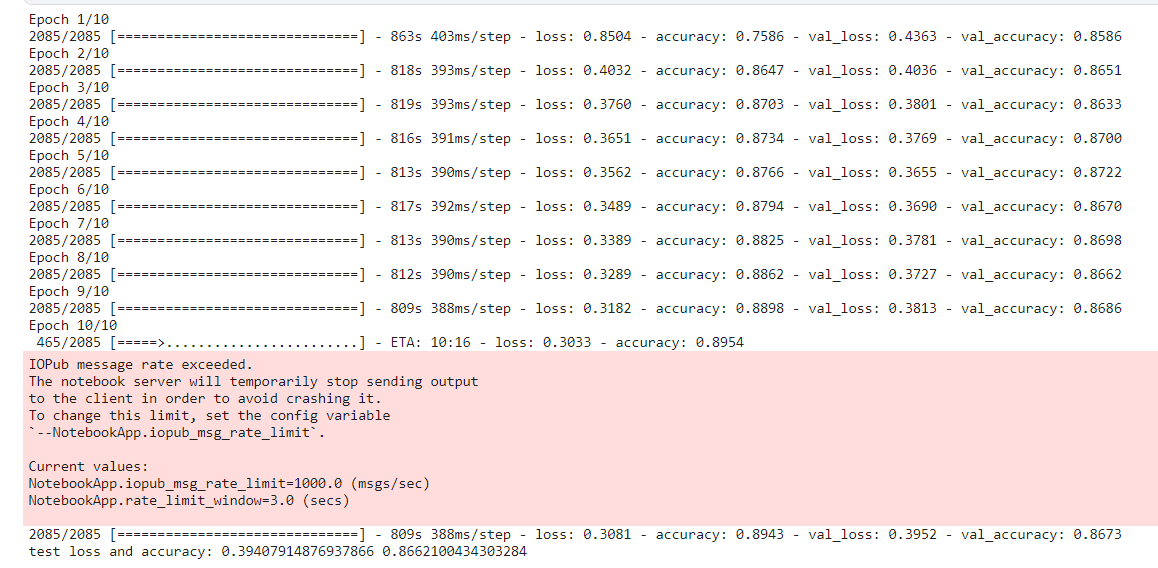
**3.2 BiLSTM+RNN**

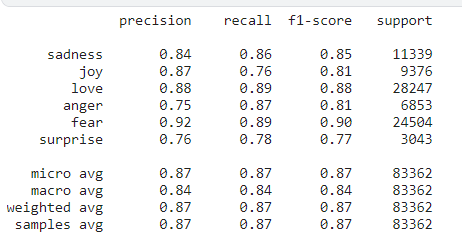
在此基础上，我们将模型架构更加复杂化。为了能更好的学习到文本中的上下文关系，我们使用了两层BiLSTM，双向学习上下文；其次，RNN的参数量也有所增加；最终，引入Dropout层一方面减少时间成本，另一方面防止严重的过拟合。

模型参数架构：



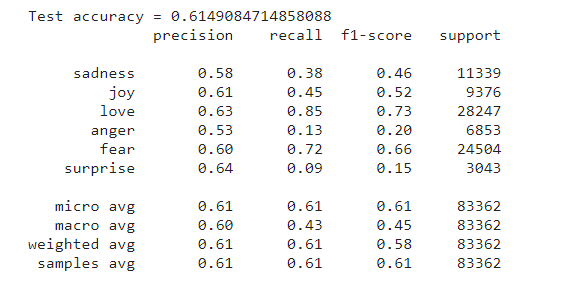
预测结果：



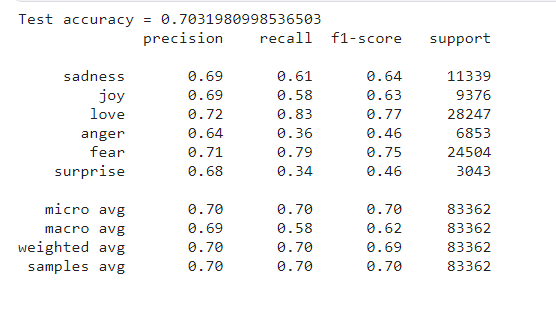


\*\*\*\*实验结果\*\*\*\*

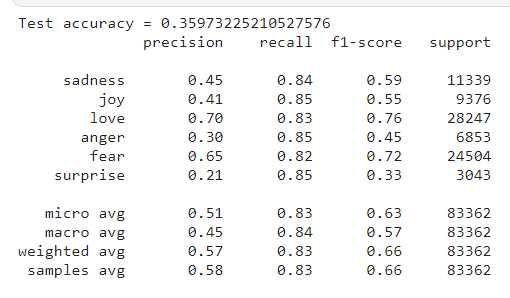
Doc2vec -》 svm



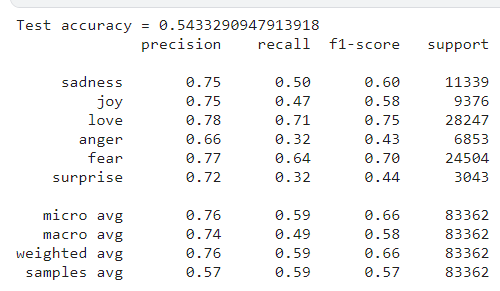
Sentencebert -》svm



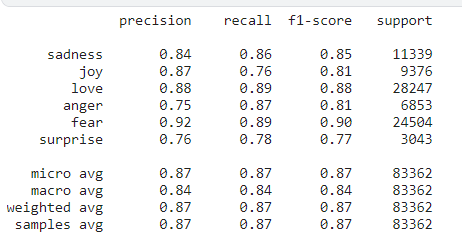
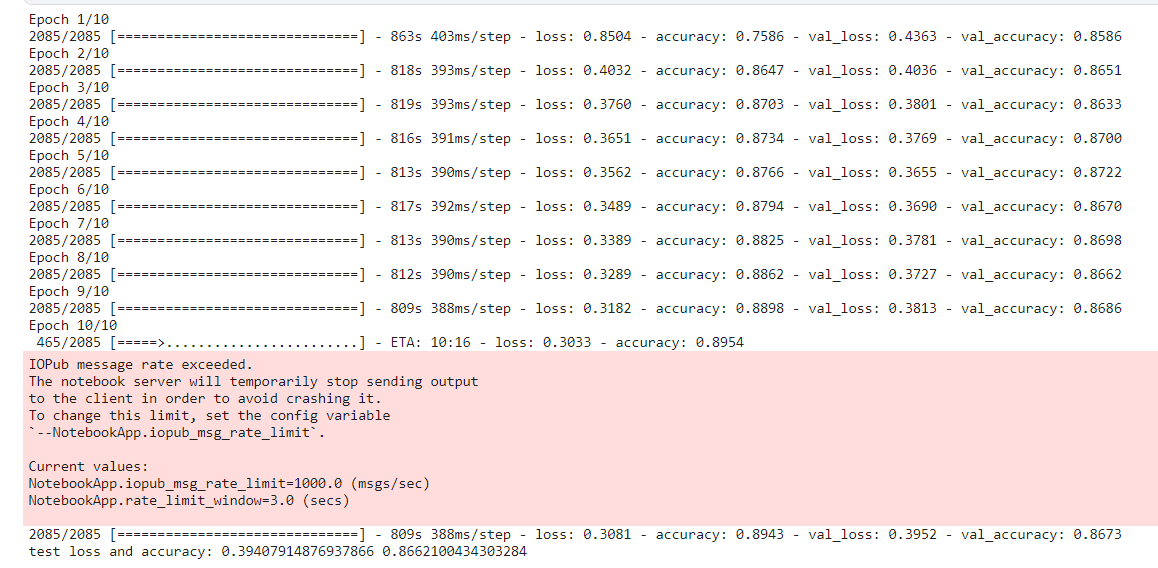
Sentencebert -》ridgemulticlassification



Sentencebert -》logisticmulticlassification

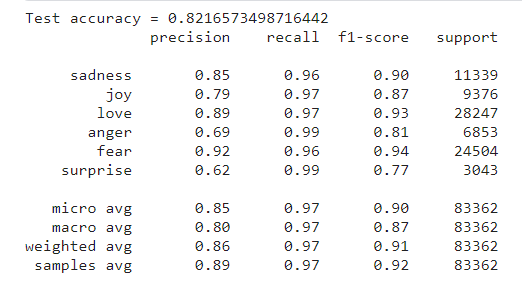


Embedding -》 rnn+bilstm

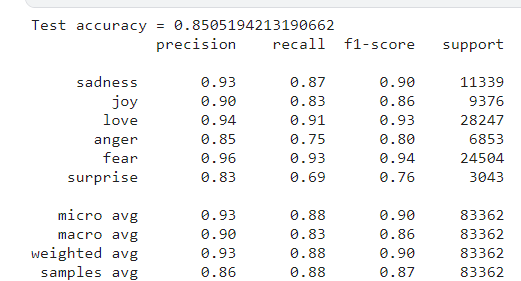


Embedding -》 lstm

Tfidf -》ridgemulticlassification



Tfidf -》logisticmulticlassification



Tfidf -》svm

Glove -》 bi-lstm

、//////////////////////////