Toxic Comment Classification Challenge

团队:芜湖起飞队

队长：李斌

团队成员：张艳蕾 王子怡 张赛

2021 / 04 / 14

目录

1. 问题描述

识别和分类有毒的在线评论(Toxic Comment Classification Challenge)

2. 数据说明

数据集来自维基语料库数据集,该数据集是由人类评级机构对毒性进行评级的。语料库包含了 6300 万条评论，这些评论来自 2004-2015 年的用户页面和文章。这些评论被分为以下六类:

toxic(恶意),severetoxic(穷凶极恶),obscene(猥琐),threat(恐吓),insult(侮辱),identityhate(种族歧视)。

要求基于数据集分类六种不同的情感。

3. 解决思路解决方案

1.导入必要的库

```
import os

import csv

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

2.训练集数据读取

```
data_path = "/Users/blackhole6/Downloads/train(1).csv"

data_raw = pd.read_csv(data_path)

print("Number of rows in data =",data_raw.shape[0])

print("Number of columns in data =",data_raw.shape[1])

print("\n")

print("**Sample data:**")

data_raw.head()
```

运行如下：

```
Number of rows in data = 159571
Number of columns in data = 8
```

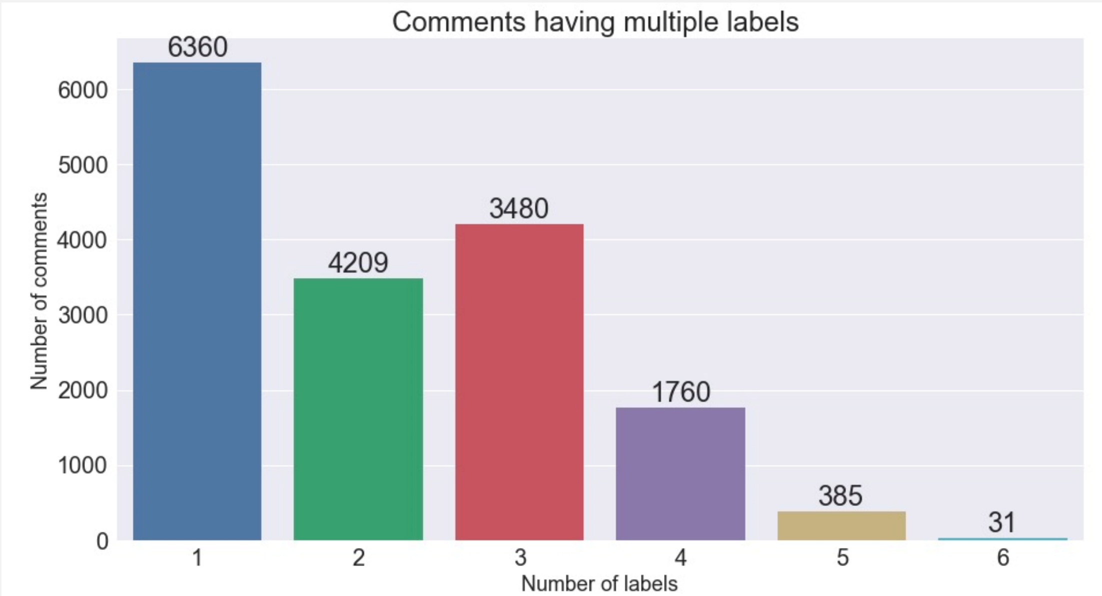
Sample data:

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

3. 统计每个标签的数量

```
categories = list(data_raw.columns.values)

sns.set(font_scale = 2)

plt.figure(figsize=(15,8))

ax= sns.barplot(categories, data_raw.iloc[:,2:].sum().values)

plt.title("Comments in each category", fontsize=24)
```

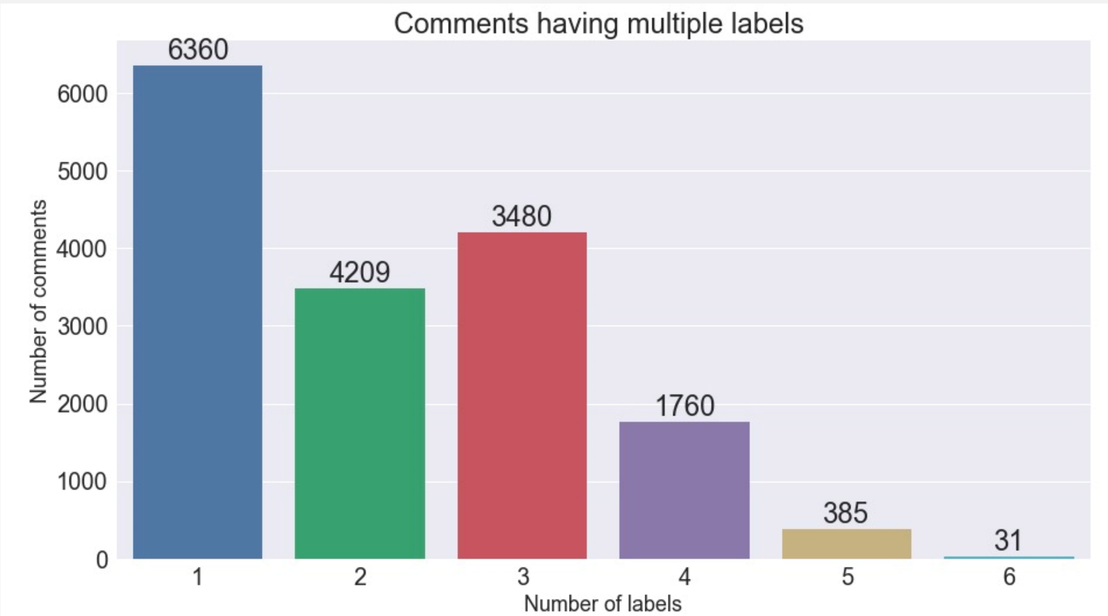
```
plt.ylabel(&apos;Number of comments&apos;, fontsize=18)

plt.xlabel(&apos;Comment Type &apos;, fontsize=18)

#adding the text labels

rects = ax.patches

labels = data_raw.iloc[:,2:].sum().values

for rect, label in zip(rects, labels):

height = rect.get_height()

ax.text(rect.get_x() + rect.get_width()/2, height + 5, label,

ha=&apos;center&apos;, va=&apos;bottom&apos;, fontsize=18)

plt.show()
```



4. 计算多重标签的评论数

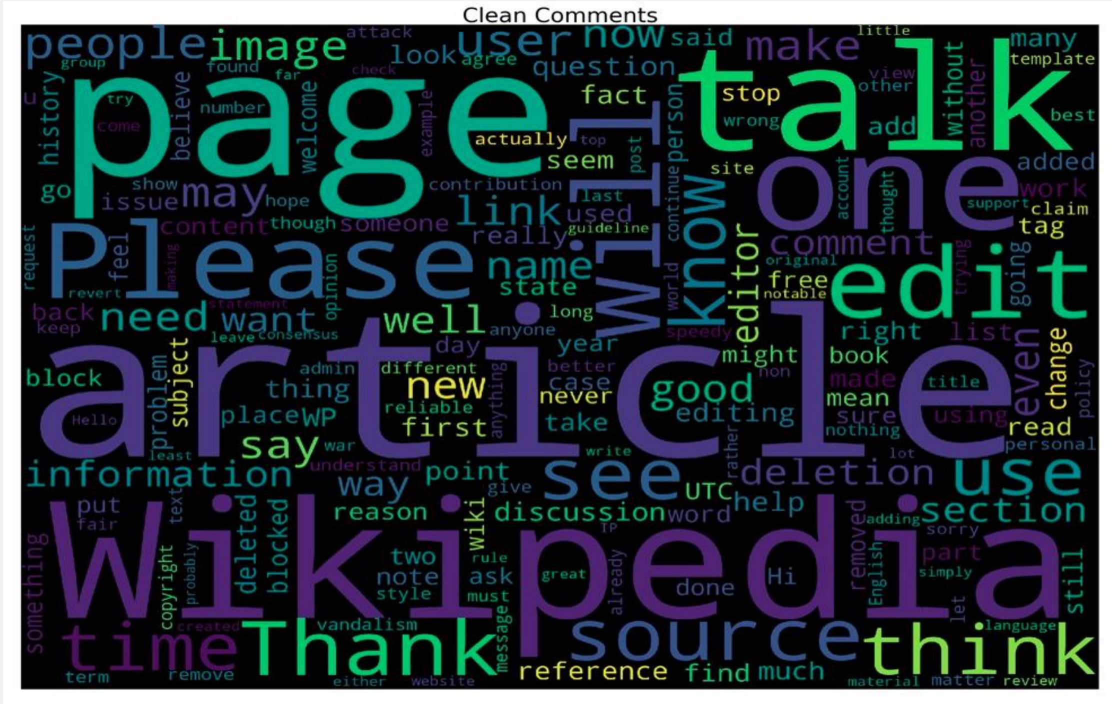
```
rowSums = data_raw.iloc[:,2:].sum(axis=1)

multiLabel_counts = rowSums.value_counts()

multiLabel_counts = multiLabel_counts.iloc[1:]

sns.set(font_scale = 2)
```

```
plt.figure(figsize=(15,8))
ax = sns.barplot(multiLabel_counts.index,
multiLabel_counts.values)
plt.title("Comments having multiple labels ")
plt.ylabel(&apos;Number of comments&apos;, fontsize=18)
plt.xlabel(&apos;Number of labels&apos;, fontsize=18)
#adding the text labels
rects = ax.patches
labels = multiLabel_counts.values
for rect, label in zip(rects, labels):
height = rect.get_height()
ax.text(rect.get_x() + rect.get_width()/2, height + 5, label,
ha=&apos;center&apos;, va=&apos;bottom&apos;)
plt.show()
```



5. 每个注释类别中最常用的单词的 WordCloud 表示形式。

```
from wordcloud

import WordCloud,STOPWORDSplt.figure(figsize=(40,25))# clean

subset = data_raw[data_raw.clean==True]

text = subset.comment_text.values

cloud_toxic = WordCloud(

                stopwords=STOPWORDS,

                background_color='black',

                collocations=False,

                width=2500,

                height=1800

).generate(" ".join(text))

plt.axis('off')

plt.title("Clean",fontsize=40)

plt.imshow(cloud_clean)# Same code can be used to generate

wordclouds of other categories.
```



6. 数据预处理：

我们首先将注释转换为小写，然后使用库函数从注释中删除 html 标签，标点符号和非字母字符。(处理非法数据)

```
import nltk

from nltk.corpus import stopwords

                from nltk.stem.snowball import SnowballStemmer

                import re

                import sys

                import warningsdata = data_rawif not sys.warnoptions:

warnings.simplefilter("ignore")def cleanHtml(sentence):

cleanr = re.compile('<.*?>')

cleantext = re.sub(cleanr, ' ', str(sentence))

return cleantextdef cleanPunc(sentence): #function to clean the word

of any punctuation or special characters

cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
```

```
cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)

cleaned = cleaned.strip()

cleaned = cleaned.replace("\n"," ")

return cleaneddef keepAlpha(sentence):

alpha_sent = ""

for word in sentence.split():

        alpha_word = re.sub('[^a-z A-Z]+', ' ', word)

alpha_sent += alpha_word

        alpha_sent += " "

alpha_sent = alpha_sent.strip()

return alpha_sentdata['comment_text'] =

data['comment_text'].str.lower()

data['comment_text'] = data['comment_text'].apply(cleanHtml)

data['comment_text'] = data['comment_text'].apply(cleanPunc)

data['comment_text'] = data['comment_text'].apply(keepAlpha)
```

接下来，我们使用可以从 NLTK 库下载的默认停用词集删除注释中存在的停用词。我们再向列表中添加一些停用词。停用词基本上是一组使用任何语言（而不只是英语）的常用词。停用词对许多应用程序至关重要的原因是，如果我们删除给定语言中非常常用的词，则可以将注意力集中在重要的词上。

```
stop_words = set(stopwords.words('english'))

stop_words.update(['zero','one','two','three','four','five','six','seven','ei
ght','nine','ten','may','also','across','among','beside','however','yet','wit
hin'])

re_stop_words = re.compile(r"\b(" + "|".join(stop_words) + ")\\W",
```

```
re.I)
def removeStopWords(sentence):
global re_stop_words
return re_stop_words.sub(" ", sentence)data['comment_text'] =
data['comment_text'].apply(removeStopWords)
```

接下来，我们要筛除词根意思相同的词汇，将具有大致相同语义的单词转换为一种标准形式。例如娱乐和欢乐，其词根相同。

```
stemmer = SnowballStemmer("english")
def stemming(sentence):
stemSentence = ""
for word in sentence.split():
        stem = stemmer.stem(word)
stemSentence += stem
        stemSentence += " "
stemSentence = stemSentence.strip()
return stemSentencedata['comment_text'] =
data['comment_text'].apply(stemming)
```

将数据集按照 8：2 进行拆分，前 80%作为训练集，后 20%作为测试集，并转化为数值向量并根据单词出现的频度进行标注，利用 TDF-If 进行计算

```
from sklearn.model_selection import train_test_splittrain, test =
train_test_split(data, random_state=42, test_size=0.30,
```

```
shuffle=True)from sklearn.feature_extraction.text import
TfidfVectorizer
        vectorizer = TfidfVectorizer(strip_accents='unicode',
analyzer='word', ngram_range=(1,3), norm='l2')
vectorizer.fit(train_text)
vectorizer.fit(test_text)x_train = vectorizer.transform(train_text)
y_train = train.drop(labels = ['id','comment_text'], axis=1)x_test =
vectorizer.transform(test_text)
y_test = test.drop(labels = ['id','comment_text'], axis=1)
```

至此数据预处理完成

7．多个二分类

首先考虑 6 种标签的识别是属于多分类问题的，我们的第一个方向是将多分类转化成多个二分类问题，建立多个二分类器，先利用线性回归模型，将二分类做好，然后多个二分类器进行划分,准确率为 0.856666666667

```
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.multiclass import OneVsRestClassifier
# Using pipeline for applying logistic regression and one vs rest
classifier
        LogReg_pipeline = Pipeline([
        ('clf',
OneVsRestClassifier(LogisticRegression(solver='sag'), n_jobs=-1)),
```

```
])for category in categories:
print('**Processing {} comments...**'.format(category))
# Training logistic regression model on train data
LogReg_pipeline.fit(x_train, train[category])
# calculating test accuracy
prediction = LogReg_pipeline.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(test[category],
prediction)))
# using binary relevance
from skmultilearn.problem_transform import BinaryRelevance
from sklearn.naive_bayes import GaussianNB# initialize binary
relevance multi-label classifier
# with a gaussian naive bayes base classifier
classifier = BinaryRelevance(GaussianNB())# train
        classifier.fit(x_train, y_train)# predict
        predictions = classifier.predict(x_test)# accuracy
        print("Accuracy = ",accuracy_score(y_test,predictions))
```

Output: *Accuracy = 0.856666666667*

8.ML-KNN 算法

利用 ML-KNN 进行分类，并得到准确率准确率：0.88166666667

```
from skmultilearn.adapt import MLkNN
from scipy.sparse import csr_matrix, lil_matrixclassifier_new =
MLkNN(k=10)# Note that this classifier can throw up errors when
```

```
handling sparse matrices.x_train = lil_matrix(x_train).toarray()
y_train = lil_matrix(y_train).toarray()
x_test = lil_matrix(x_test).toarray()# train
        classifier_new.fit(x_train, y_train)# predict
        predictions_new = classifier_new.predict(x_test)#
accuracy
                print("Accuracy =
",accuracy_score(y_test,predictions_new))
print("\n")
```

Output: *Accuracy = 0.88166666667*。

4.总结与展望

总结：

解决多标签分类问题的主要方法有两种：**转换为多个二分类**和**常用的多分类方法**。

问题转换方法将多标签问题转换为多个二分类问题，然后可以使用单分类器对其进行处理。而多分类方法则使算法直接进行行多标签分类。换句话说，他们没有尝试将问题转换为更简单的问题，而是尝试以完整的形式解决该问题。在此数据集上运行时，ML-KNN 和多个二分类都需要花费大量时间，因此对训练数据的进行随机样本实验。

改进：

- 在深度学习中使用 LSTM 可以解决相同的问题。
- 为了获得更高的速度，我们可以使用决策树，并且为了在速度和准确性之间进行合理的权衡，我们还可以选择集成模型。

- 其他框架（例如 MEKA）可用于处理多标签分类问题