【NLP】文本分类任务之逻辑回归

AI小白入门 1月17日

以下文章来源于AI算法之心 ， 作者何从庆

简介

在某些平台评论中会经常出现一些有毒评论（即一些粗鲁，不尊重或者可能让某人离开讨论的评论），这使得许多人不愿意再表达自己并放弃在平台中评论。因此，为了促进用户对话，提出一系列的方案，来缓解这一问题。我们将其看作一个**文本分类**问题，来介绍一系列的文本分类方案。

数据描述

数据分为训练集和测试集，训练集包含153165条样本，测试集包含153164条样本，标签分为6类，分别是toxic，severe_toxic，obscene，threat，insult，identity_hate。部分样本展示如下：

	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'awww! 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

评价指标

每类标签的AUC的平均值，作为评价指标。

方案

在这篇文章中，我将介绍最简单也是最常用的一种文本分类方法——从**TFIDF中提取文本的特征**，以**逻辑回归**作为分类器。

1、首先，我们利用TFIDF提取文本词语的信息：

```
word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents=unicode,
    analyzer=word,
    token_pattern=r\w{1,},
    stop_words=english,
    ngram_range=(1, 1),
    max_features=10000)
word_vectorizer.fit(all_text)
train_word_features = word_vectorizer.transform(train_text)
test_word_features = word_vectorizer.transform(test_text)
```

2、为了充分表征文本信息，我们也提取文本字的ngram信息，我们将ngram设置为（2，6），也就是说我们会最少提取两个字母作为单词的信息，最多会提取6个字母作为单词：

```
char_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents=unicode,
    analyzer=char,
    stop_words=english,
    ngram_range=(2, 6),
    max_features=50000)
char_vectorizer.fit(all_text)
train_char_features = char_vectorizer.transform(train_text)
test_char_features = char_vectorizer.transform(test_text)
```

3、接下来我们就开始合并文本词表征以及字表征：

```
train_features = hstack([train_char_features, train_word_features])
test_features = hstack([test_char_features, test_word_features])
```

4、所有的文本特征都提取完成后，我们就可以用机器学习的分类器——逻辑回归，训练模型。这是一个多标签问题，我们将其看作6个二分类问题求解，即我们假设两两标签是没有关系的。

```
scores = []
submission = pd.DataFrame.from_dict({id: test[id]})
for class_name in class_names:
    train_target = train[class_name]
    classifier = LogisticRegression(C=0.1, solver=sag)
    classifier.fit(train_features, train_target)
    submission[class_name] = classifier.predict_proba(test_features)[:, 1]
```

如果你想运行该代码，可参考完整代码如下：

```
import numpy as np
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from scipy.sparse import hstack

class_names = [toxic, severe_toxic, obscene, threat, insult, identity_hate]

train = pd.read_csv(../input/train.csv).fillna( )
test = pd.read_csv(../input/test.csv).fillna( )

train_text = train[comment_text]
test_text = test[comment_text]
all_text = pd.concat([train_text, test_text])

word_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents=unicode,
    analyzer=word,
    token_pattern=r\w{1,},
    stop_words=english,
    ngram_range=(1, 1),
    max_features=10000)
word_vectorizer.fit(all_text)
train_word_features = word_vectorizer.transform(train_text)
test_word_features = word_vectorizer.transform(test_text)

char_vectorizer = TfidfVectorizer(
    sublinear_tf=True,
    strip_accents=unicode,
    analyzer=char,
    stop_words=english,
    ngram_range=(2, 6),
    max_features=50000)
char_vectorizer.fit(all_text)
train_char_features = char_vectorizer.transform(train_text)
test_char_features = char_vectorizer.transform(test_text)

train_features = hstack([train_char_features, train_word_features])
test_features = hstack([test_char_features, test_word_features])

scores = []
submission = pd.DataFrame.from_dict({id: test[id]})
for class_name in class_names:
    train_target = train[class_name]
    classifier = LogisticRegression(C=0.1, solver=sag)

    cv_score = np.mean(cross_val_score(classifier, train_features, train_target, cv=3,
    scores.append(cv_score)
    print(CV score for class {} is {}.format(class_name, cv_score))
```

```
    classifier.fit(train_features, train_target)
    submission[class_name] = classifier.predict_proba(test_features)[:, 1]

print(Total CV score is {}.format(np.mean(scores)))

submission.to_csv(submission.csv, index=False)
```

上述代码可以在github上面找到：https://github.com/hecongqing/TextClassification

数据集可以在这里下载：https://share.weiyun.com/5c7KYLw

接下来的文章中我将介绍其他的机器学习方法和深度学习来解决有毒评论的文本分类问题。

参考

1、https://www.kaggle.com/tunguz/logistic-regression-with-words-and-char-n-grams

The End