达观杯文本智能挑战赛-逻辑回归

任务背景

竞赛网址

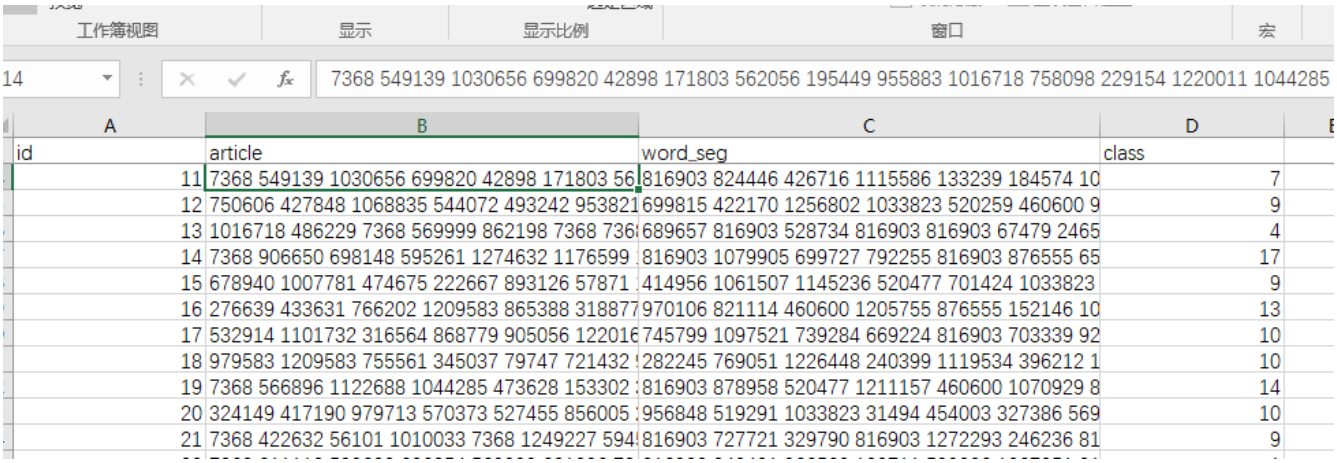
这个比赛的任务就是**文本分类**，是自然语言处理（NLP）领域里一项最最基本的任务。但这个任务的难点就是在于，文本的长度非常长，大约3000个词，一般任务也就300词。而文本的长度过长对文本的智能解析带来了很多挑战。

用传统的监督学习模型对一段文本进行分类的基本过程：

一段原始文本通过数据预处理得到处理后的文本，然后进行特征工程，得到Features，之后输入模型 $y = f(x_1, x_2, \ldots, x_n)$，得到输出类别。

数据集

训练数据集包括id，article，word_seg和class，总计102277行数据



id	article	word_seg	class
11	7368 549139 1030656 699820 42898 171803 56	816903 824446 426716 1115586 133239 184574 10	7
12	750606 427848 1068835 544072 493242 953821	699815 422170 1256802 1033823 520259 460600 9	9
13	1016718 486229 7368 569999 862198 7368 736	689657 816903 528734 816903 816903 67479 2465	4
14	7368 906650 698148 595261 1274632 1176599	816903 1079905 699727 792255 816903 876555 65	17
15	678940 1007781 474675 222667 893126 57871	414956 1061507 1145236 520477 701424 1033823	9
16	276639 433631 766202 1209583 865388 318877	970106 821114 460600 1205755 876555 152146 10	13
17	532914 1101732 316564 868779 905056 122016	745799 1097521 739284 669224 816903 703339 92	10
18	979583 1209583 755561 345037 79747 721432	282245 769051 1226448 240399 1119534 396212 1	10
19	7368 566896 1122688 1044285 473628 153302	816903 878958 520477 1211157 460600 1070929 8	14
20	324149 417190 979713 570373 527455 856005	956848 519291 1033823 31494 454003 327386 569	10
21	7368 422632 56101 1010033 7368 1249227 594	816903 727721 329790 816903 1272293 246236 81	9

测试数据集除了没有class之外和训练数据集相同，总计102277行数据。

数据中的数字都代表的是脱敏后的字或者词。article是一篇由字组成的文章，word_seg是由词组成的文章，class代表对应的类别。

逻辑回归解决方案

我们提供了一个简单的逻辑回归方案。

首先，我们进行第一步——数据预处理：

```
df_train = pd.read_csv("./train_set.csv")
df_test = pd.read_csv("./test_set.csv")
df_train.drop(columns=['article', 'id'], inplace=True)
df_test.drop(columns=['article'], inplace=True)
```

分析比赛数据我们可以看出数据非常的干净，没有空缺值等，我们只需要直接上手即可。使用pandas读入数据后，我们删除了article和id这两列不需要的信息。article是由字组成的，单个字很多是不存在意义的，会对模型产生误导信息，我们简单的不进行处理，直接删除。

第二步——特征工程：

```
vectorizer = CountVectorizer(ngram_range=(1, 2), min_df=3, max_df=0.9, max_features=100000)
vectorizer.fit(df_train['word_seg'])
x_train = vectorizer.transform(df_train['word_seg'])
x_test = vectorizer.transform(df_test['word_seg'])
y_train = df_train['class'] - 1
```

我们使用scikit-learn中的CountVectorizer从文章中提取出特征信息，CountVectorizer只考虑词汇在文本中出现的频率。除此之外，我们还可以使用TfidfVectorizer，除了考量某一词汇在当前训练文本中出现的频率之外，同时关注包含这个词汇的其它训练文本数目的倒数，能够削减高频没有意义的词汇出现带来的影响, 挖掘更有意义的特征。

在这个阶段，我们会将字符文本转化成数字向量，以便计算机能够进行处理。

第三步——构建模型进行训练：

```
lg = LogisticRegression(C=4, dual=True)
lg.fit(x_train, y_train)
```

我们使用scikit-learn中的LogisticRegression模型，之后我们会使用其他的模型，以便大家进行比较。

模型中的c指定了惩罚函数的倒数，值越小，正则化越大。此外，模型还有一些其他的参数：penalty指定正则化策略，solver求解最优化问题的算法等，大家可以自己多多尝试。

第四步——输出：

```
y_test = lg.predict(x_test)

df_test['class'] = y_test.tolist()
df_test['class'] = df_test['class'] + 1
df_result = df_test.loc[:, ['id', 'class']]
df_result.to_csv('./result.csv', index=False)
```