新闻文本分类项目报告

1. 数据获取及预处理

1.1 数据来源

近年来，随着人工智能的发展，其在语音识别、自然语言处理、图像与视频分析等诸多领域取得了巨大成功。本次项目将聚焦在新闻文本的分类，利用人工智能技术，对新闻文本进行分类。要求给出一个算法或模型，对于给定的文本，检测出文本所属类别。

从天池官网下载到完整的新闻数据集，地址为[天池-新闻文本分类]

1.2 数据说明

（1）数据集分为1个训练集和测试集A,B

- 训练集含数据样本**20万条**
- 测试集A含数据样本**5万条**
- 测试集B是网站用于对提交模型的精度进行评估的，不支持下载

（2）训练集原始字段为label（新闻类别），text（匿名处理后的新闻文本）

	label	text
0	2	2967 6758 339 2021 1854 3731 4109 3792 4149 1519 2058 3912 2465 2410 1219 6654 7539 264 2456 4811 1292 2109 6905 552...
1	11	4464 486 6352 5619 2465 4802 1452 3137 5778 5445 26 6663 5530 4149 2986 1746 5491 3659 2662 3002 2986 6182 4603 3220...
2	3	7346 4068 5074 3747 5681 6093 1777 2226 7354 6301 2465 6088 5858 4333 1386 1401 5780 290 541 6350 5491 1580 2662 533...
3	2	7159 948 4866 2109 5520 2490 211 3956 5520 5492 7006 316 2828 2058 2331 2465 5410 4646 6983 6781 6902 4211 7394 7495...
4	3	3646 3055 3055 2490 4659 6065 3370 5814 2465 5192 4876 3585 4298 5659 2109 1043 3994 7393 5099 3725 2109 6088 299 52...
...	...	...
199995	2	307 4894 7539 4853 5330 648 6038 4409 3764 6038 1722 6407 2465 462 3018 5298 5949 94 3099 4411 7159 6248 4603 7495 2...
199996	2	3792 2983 355 1070 4464 5050 6298 3782 3130 6898 5589 7154 2592 2465 2289 1605 1379 5538 5589 847 3692 3661 3605 332...
199997	11	6811 1580 7539 1252 1899 5139 1386 3870 4124 1946 955 3359 5445 26 2804 361 1271 1519 5788 2073 648 1070 1036 1903 5...
199998	2	6405 3203 6644 983 794 1913 1678 5736 1397 1913 5221 1722 2410 5816 7444 2465 3440 6298 7148 5005 2683 2109 4269 754...
199999	3	4350 3878 3268 1699 6909 5505 2376 2465 6088 2756 1734 4464 6250 1324 7371 6301 5491 1580 2662 5999 4853 4811 4559 2...

200000 rows × 2 columns

（3）测试集A仅包含text（匿名处理后的新闻文本）字段

	text
0	5399 3117 1070 4321 4568 2621 5466 3772 4516 2990 3618 2456 3335 3272 7495 2435 4568 5915 134 2465 5284 1779 5445 19...
1	2491 4109 1757 7539 648 3695 3038 4490 23 7019 3731 4109 3792 2465 2893 7212 5296 1667 3618 7044 1519 5413 1283 6122...
2	2673 5076 6835 2835 5948 5677 3247 4124 2465 5192 6101 913 4326 3615 442 4409 2466 6552 465 7399 1859 7449 5192 6101...
3	4562 4893 2210 4761 3659 1324 2595 5949 4583 2465 4893 1699 2461 3585 4583 2859 3254 3772 1465 3329 3659 1324 2595 5...
4	4269 7134 2614 1724 4464 1324 3370 3370 2106 2465 88 7400 5602 3370 4876 5574 6649 5780 623 6637 2708 2400 910 2722 ...
...	...
49995	3725 4498 2282 1647 6293 4245 4498 3615 1141 2828 2205 6453 2169 1726 3364 6289 281 810 343 2119 3586 2465 2465 6770...
49996	4811 465 3800 1394 3038 2376 2327 5165 3070 5744 5491 1580 2662 5917 1610 5028 5744 1500 4958 5693 2210 3750 2827 15...
49997	5338 1952 3117 4109 299 6656 6654 3792 6831 2151 4409 5430 922 433 5410 764 922 433 465 2004 5338 3117 3659 6065 485...
49998	893 3469 5775 584 2490 4223 6569 6663 2124 1684 2465 7346 2310 656 3747 5631 641 307 5051 4205 3750 893 3469 2490 61...
49999	2400 4409 4412 2210 5122 4464 7186 2465 1327 94 669 307 2109 2407 5631 1854 4811 3937 6357 307 5176 623 3397 648 240...

50000 rows × 1 columns

1.3 数据预处理

（1）查看数据缺失情况和重复情况

- 无缺失值，无需处理
- 需处理重复数据

```
#查看数据集的数据缺失和重复情况
print('数据缺失情况：')
print(train_df.isnull().sum())
print('数据重复情况：')
print(train_df.duplicated().value_counts())
```

```
数据缺失情况：
label    0
text     0
dtype: int64
数据重复情况：
False    199911
True         89
dtype: int64
```

(2) 数据去重

```
#去除重复值
train_df=train_df.drop_duplicates(keep='last')
train_df.duplicated().value_counts()
```

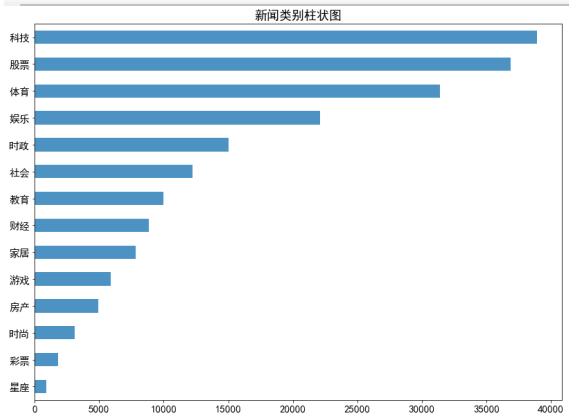
```
False    199911
dtype: int64
```

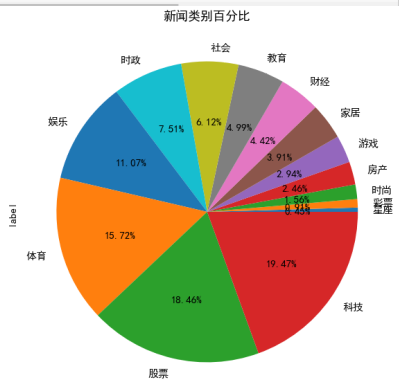
2. 数据分析与可视化

2.1查看新闻类别的柱状图和占比可知：

- 数量排名前4的科技、股票、体育、娱乐类占据了总体60%以上的新闻
- 数量倒数后5的游戏、房产、时尚、彩票、星座类合计占10%左右
- 整体类别分布不均匀

```
#划分训练数据集和标签集
train_data = train_df['text']
train_label = train_df['label']
#设置签字典
label_map={'科技': 0, '股票': 1, '体育': 2, '娱乐': 3, '时政': 4, '社会': 5, '教育': 6, '财经': 7, '家居': 8, '游戏': 9, '房产': 10, '时尚
label_map={value:key for key, value in label_map.items()}
#可视化查看新闻文本类型比例
import matplotlib.pyplot as plt
import numpy as np
plt.rc("font",family='SimHei',size=15)
fig = plt.figure(figsize=(30,10))
fig.add_subplot(1,2,1)
train_df['label'].value_counts().rename(index=label_map).sort_values(ascending=True).plot(kind='barh',alpha=0.8)
plt.title('新闻类别柱状图')
fig.add_subplot(1,2,2)
train_df['label'].value_counts().rename(index=label_map).sort_values(ascending=True).plot(kind='pie',autopct='%.2f%%')
plt.title('新闻类别百分比')
plt.show()
```





2.2.查看新闻文本的字符长短情况：

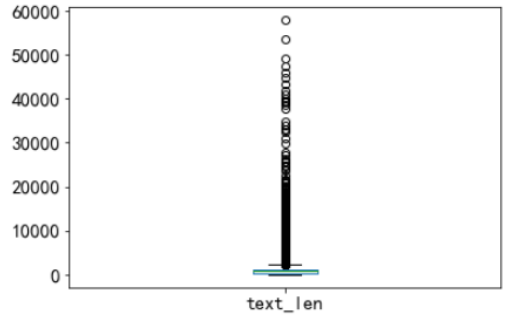
- 由对新闻文本字符数目的五数概括描述可知：
  - 平均长度为907个字符
  - 最短2个字符

- 25%在374个字符以内
- 50%在647个字符以内
- 75%在1131个字符以内
- 最长57921个字符
- 由新闻文本字符数目箱型图可观察到较多离群点，整体新闻文本字符数分布不太均匀，结合直方图可分析其绝大部分整体长度都在将近2000以下

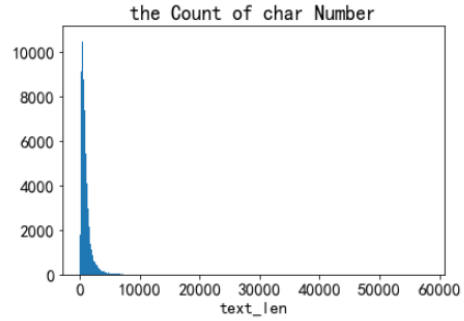
```
#查看新闻文本的字符数的5数概括和箱图
train_df['text_len']=train_df['text'].apply(lambda x: len(x.split(' ')))
print(train_df['text_len'].describe())
train_df['text_len'].plot(kind='box')
```

```
count    199911.000000
mean        907.460460
std         996.110506
min           2.000000
25%         375.000000
50%         676.000000
75%        1131.000000
max       57921.000000
Name: text_len, dtype: float64

<AxesSubplot:>
```



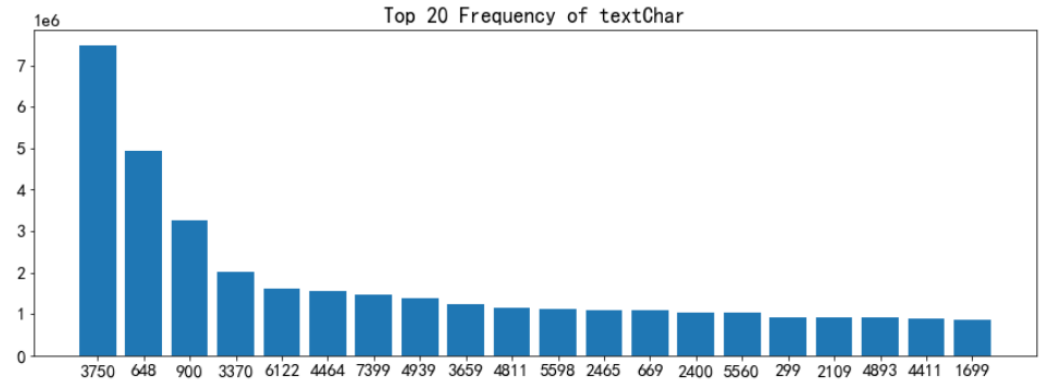

```
#文本字符数量直方图
plt.hist(train_df['text_len'], bins=1000)
plt.xlabel('text_len')
plt.title("the Count of char Number")
plt.show()
```



2.3 查看所有新闻文本中出现频率最高的前20个字符：

- 出现频率最高的是字符'3750'，748万次
- 排名前3字符之后基本都低于200万次

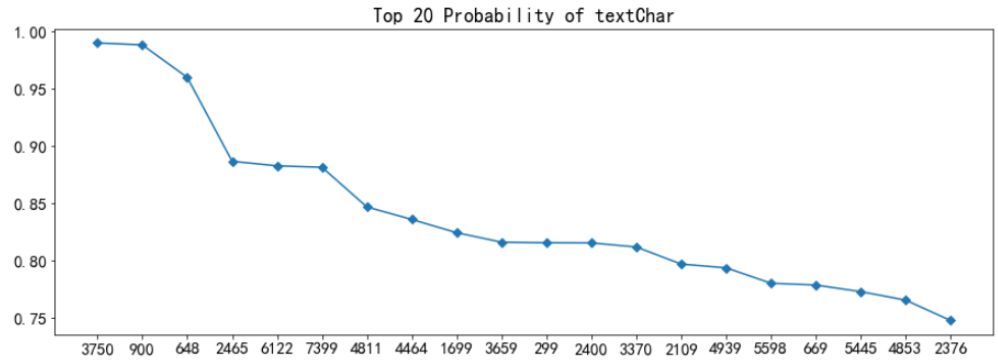
```
# 统计并可视化文本中出现频率前20的字符
from collections import Counter
all_lines = ' '.join(list(train_df['text']))
word_count = Counter(all_lines.split(" "))
word_count = sorted(word_count.items(), key=lambda d:d[1], reverse = True)
ch=[]
ch_count=[]
for i in range(0,20):
    ch.append(int(word_count[i][0]))
    ch_count.append(word_count[i][1])
#可视化
plt.figure(figsize=(15,5))
plt.bar(range(len(ch_count)), ch_count, tick_label=ch)
plt.title('Top 20 Frequency of textChar')
plt.show()
```



2.4 统计各字符在20万条新闻中的出现概率：

- 前3的字符“3750”，“900”，“648”的**在20万条新闻中每条出现概率超过95%**，猜测可能是标点符号(符合上条前三字符均超过200万次)

```
# 统计并可视化各字符在20万条新闻中的出现概率（前20的）
train_df['text_unique'] = train_df['text'].apply(lambda x: ' '.join(list(set(x.split(' ')))))
all_lines = ' '.join(list(train_df['text_unique']))
unique_word_count = Counter(all_lines.split(" "))
unique_word_count = sorted(unique_word_count.items(), key=lambda d:int(d[1]), reverse = True)
ch1=[]
ch1_count=[]
for i in range(0,20):
    ch1.append(unique_word_count[i][0])
    ch1_count.append(unique_word_count[i][1]/200000)
#折线图可视化
plt.figure(figsize=(15,5))
plt.plot(ch1,ch1_count,marker='D', markersize=5)
plt.title('Top 20 Probability of textChar')
plt.show()
```



3. 模型选取

3.1 TFIDF+SVM分类

3.1.1 模型选取缘由：

词频-逆向文档频率(Term Frequency-Inverse Document Frequency,TFIDF) 是一种基于统计的数学方法，其通过统计当前样本中每个词的 TFIDF 值作为特征向量的每个元素，从而可以判断出一个词语或短语在样本集中的重要程度。TFIDF 方法计算公式如下:

$$F_{\mathrm{TF}_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (1)$$

$$F_{\mathrm{IDF}_{i,j}} = \log \frac{|D|}{1 + |\{j:\ n_i \in d_j\}|} \quad (2)$$

$$F_{\mathrm{TFIDF}_{i,j}} = F_{\mathrm{TF}_{i,j}} \times F_{\mathrm{IDF}_{i,j}} \quad (3)$$

其中 $n_{i,j}$ 为某词条 n_i 在样本 j 中出现次数，F_{TF} 为词频，D 为样本总数，$|\{j:\ n_i \in d_j\}|$ 为包含某词条 n_i 的样本总数，F_{IDF} 为逆文档频率。

由上述定义，FTFIDFi，j越大，则表示某词条 ni 对样本 j 越重要，TFIDF方法在文本特征提取上有广泛的应用，且在文本分类任务中效果显著，因此本项目用TFIDF提取文本特征。

支持向量机（Support Vector Machine，SVM）是作为是一类按监督学习方式对数据进行二元分类的线性分类器，它具有泛化错误率低、计算开销不大和结果易理解等特点，同时像本实验数据集样本总数（本实验20万样本）比文本特征词数量（本实验最大文本特征数量为1万）比较大时，更适合用基于线性核函数的SVM（或logistic回归）来完成分类任务，因此本项目用SVM线性分类器完成文本分类任务。

3.1.2 实验步骤

- 首先，利用TFIDF提取文本词语的特征信息；
- 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（1，3），即最少提取单个词语作为文本特征，最多会提取3个词语作为文本特征词；
- 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——支持向量机（SVM），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。

3.1.3 分类结果

由于天池新闻文本分类赛题提供的测试集A中仅包含需要预测的新闻文本，不包含其标签，因此我们无法对测试集的结果进行直观的分析。为了能对新闻文本分类结果做一个比较直观的分析，我们从赛题提供的训练集中抽取部分数据集作为线下实验的验证集，我们把训练数据集分成5个子数据集，通过K折交叉验证的方式进行试验，其中4个子数据集作为训练数据集，1个子数据集作为验证集，从而使训练集与验证集的比值达到8:2（16万：4万），实验结果为如下，其中混淆矩阵及其他结果中的每一行从左到右（第0列到13列），从上到下（第0行到13行）分别代表：'科技'，'股票'，,'体育'， '娱乐'， '时政'， '社会'， '教育'， '财经'， '家居'， '游戏'，'房产'， '时尚'， '彩票'， '星座'等类别，其中混淆矩阵中行代表其真实值，列代表其预测值：

（1）第 1 折交叉验证的实验结果

混淆矩阵为如下：

```
混淆矩阵输出：
 [[7401  102   11   45   68   93   20   15   24   59    3    2    0    0]
 [ 100 7088    2    7   81    6    1  125    5    1    4    0    1    0]
 [   8    4 6213   19    2   11    3    0    4    0    1    4    2    0]
 [  40    7   22 4318   25   33    5    1   13    5    1   12    0    0]
 [  51   51   14   17 2785   48   19    8    3    1    1    6    1    1]
 [  62    4   10   33   76 2214   33    6    5    0    6    0    7    0]
 [  18    5    6    4   29   43 1843    4    7    2    1    1    0    1]
 [  22  161    0    5   10    7    3 1476    6    0    2    1    1    0]
 [  31   12    3   19    1    6    3    3 1476    1    0    8    0    1]
 [  79    1    1    6    2    7    1    0    1 1046    0    0    0    0]
 [   2    5    0    3    3    5    0    1    5    0  958    2    0    0]
 [   8    1    0   16    5    3    0    0   11    0    0  589    0    3]
 [   0    0   18    0    0    8    0    0    0    0    0    0  325    0]
 [   0    0    1    2    0    2    3    0    1    0    0    3    0  172]]
```

每一个新闻类别的precision、recall、f1-score以及总的accuracy为如下：

	precision	recall	f1-score	support
0	0.95	0.94	0.94	7843
1	0.95	0.96	0.95	7421
2	0.99	0.99	0.99	6271
3	0.96	0.96	0.96	4482
4	0.90	0.93	0.91	3006
5	0.89	0.90	0.90	2456
6	0.95	0.94	0.95	1964
7	0.90	0.87	0.89	1694
8	0.95	0.94	0.94	1564
9	0.94	0.91	0.93	1144
10	0.98	0.97	0.98	984
11	0.94	0.93	0.93	636
12	0.96	0.93	0.94	351
13	0.97	0.93	0.95	184
accuracy			0.95	40000
macro avg	0.94	0.94	0.94	40000
weighted avg	0.95	0.95	0.95	40000

（2）第 2 折交叉验证的实验结果

混淆矩阵为如下：

```
混淆矩阵输出:
[[7468  104    7   42   70   79   21   23   22   47    0    1    3    1]
 [  89 7125    4    5   66    8    3  135    9    0    2    1    0    0]
 [   8    3 6261   32   10    9    5    0    1    2    0    2    3    0]
 [  37    6   25 4131   18   29    5    4    7    2    0   10    0    0]
 [  55   55    8   13 2742   51   18    8    2    2    2    4    2    0]
 [  60    7    7   30   70 2165   38   10    4    1    1    2   19    0]
 [   9    4    3   11   23   30 1874    3    9    0    0    3    0    0]
 [  26  156    0   12   14   17    0 1576    6    1    1    0    0    0]
 [  28    7    3   20    8   15    4    3 1442    1    2    5    0    0]
 [  84    6    0    7    2    0    0    1    1 1089    0    1    0    1]
 [   1    5    1    1    4    7    0    5    6    0  993    1    0    0]
 [   4    0    2   19    2    2    4    0   16    5    0  549    0    0]
 [   1    0   15    0    1    8    0    0    0    0    0    1  334    0]
 [   2    0    0    0    0    1    4    0    1    0    0    7    0  169]]
```

每一个新闻类别的precision、recall、f1-score以及总的accuracy为如下：

	precision	recall	f1-score	support
0	0.95	0.95	0.95	7888
1	0.95	0.96	0.95	7447
2	0.99	0.99	0.99	6336
3	0.96	0.97	0.96	4274
4	0.90	0.93	0.92	2962
5	0.89	0.90	0.90	2414
6	0.95	0.95	0.95	1969
7	0.89	0.87	0.88	1809
8	0.94	0.94	0.94	1538
9	0.95	0.91	0.93	1192
10	0.99	0.97	0.98	1024
11	0.94	0.91	0.92	603
12	0.93	0.93	0.93	360
13	0.99	0.92	0.95	184
accuracy			0.95	40000
macro avg	0.94	0.93	0.94	40000
weighted avg	0.95	0.95	0.95	40000

（3）第 3折交叉验证的实验结果

混淆矩阵为如下：

```
混淆矩阵输出:
 [[7423  108    6   42   87   92   18   17   17   52    1    5    1    0]
 [ 109 7030    0    7   60    2    2  142    5    0    3    0    0    0]
 [   7    0 6060   27   14    8    0    0    2    1    1    5    2    0]
 [  31    7   20 4283   29   28    1    1    2    3    0    4    0    3]
 [  59   69   12   20 2760   59   29    6    1    0    1    6    0    1]
 [  84    6    6   25   70 2240   27    8    5    1    2    1    3    0]
 [  23    3    1   14   23   46 1954    2   15    2    0    2    0    2]
 [  31  159    0    9   14   20    2 1562    0    0    2    0    1    2]
 [  38   10   10   18    5    6    3    1 1470    1    1   11    0    0]
 [  80    2    7    5    2    4    0    0    1 1053    1    2    1    0]
 [   1    8    0    1    2    6    0    0    3    0  926    0    0    0]
 [   7    0    1   21    3    6    3    0   12    1    0  560    0    2]
 [   3    1   17    2    1    7    0    0    0    0    0    0  335    0]
 [   0    1    0    2    2    1    2    1    1    0    0    2    0  169]]
```

每一个新闻类别的precision、recall、f1-score以及总的accuracy为如下：

	precision	recall	f1-score	support
0	0.94	0.94	0.94	7869
1	0.95	0.96	0.95	7360
2	0.99	0.99	0.99	6127
3	0.96	0.97	0.96	4412
4	0.90	0.91	0.91	3023
5	0.89	0.90	0.90	2478
6	0.96	0.94	0.95	2087
7	0.90	0.87	0.88	1802
8	0.96	0.93	0.95	1574
9	0.95	0.91	0.93	1158
10	0.99	0.98	0.98	947
11	0.94	0.91	0.92	616
12	0.98	0.92	0.94	366
13	0.94	0.93	0.94	181
accuracy			0.95	40000
macro avg	0.94	0.93	0.94	40000
weighted avg	0.95	0.95	0.95	40000

（4）第 4折交叉验证的实验结果

混淆矩阵为如下：

```
混淆矩阵输出:
 [[7205   83    6   35   61   93   18   12   21   38    0    3    1    1]
 [  77 7137    0    5   62    6    2  108    4    1    5    1    1    0]
 [   3    1 6334   29   11   12    2    0    2    1    0    0    2    0]
 [  35    5   20 4383   13   30    2    2    9    2    0    8    0    1]
 [  60   55   12   19 2727   63   20    8    5    2    2    7    1    0]
 [  70    7    2   26   65 2182   28   11    6    0    3    3    8    0]
 [  18    5    2    5   28   36 1860    0    4    1    0    3    0    1]
 [  19  183    2    6   11   19    3 1542    3    0    3    0    1    0]
 [  26   13    7   24    4   11    4    1 1475    2    2   12    0    2]
 [ 103    2    2    5    2    2    0    0    1 1065    0    2    0    1]
 [   1   12    1    0    1    5    1    0    7    1  935    1    0    0]
 [   5    0    1   28    5    7    3    2   14    1    0  581    0    4]
 [   2    0   19    1    0    4    0    0    0    0    0    0  342    0]
 [   0    0    1    1    0    1    3    0    6    1    0    4    0  191]]
```

每一个新闻类别的precision、recall、f1-score以及总的accuracy为如下：

	precision	recall	f1-score	support
0	0.95	0.95	0.95	7577
1	0.95	0.96	0.96	7409
2	0.99	0.99	0.99	6397
3	0.96	0.97	0.97	4510
4	0.91	0.91	0.91	2981
5	0.88	0.91	0.89	2411
6	0.96	0.95	0.95	1963
7	0.91	0.86	0.89	1792
8	0.95	0.93	0.94	1583
9	0.96	0.90	0.93	1185
10	0.98	0.97	0.98	965
11	0.93	0.89	0.91	651
12	0.96	0.93	0.94	368
13	0.95	0.92	0.93	208
accuracy			0.95	40000
macro avg	0.95	0.93	0.94	40000
weighted avg	0.95	0.95	0.95	40000

（5）第 5折交叉验证的实验结果

混淆矩阵为如下：

```
混淆矩阵输出：
 [[7335   96    8   34   72   81   15   18   26   52    2    1    1    0]
 [  87 7003    1    5   75    2    6  116    7    3    3    0    0    0]
 [   8    3 6228   27   11    4    3    0    4    0    0    1    5    0]
 [  26    3   22 4320   14   25    8    1   16    4    0   13    0    3]
 [  56   59   14   15 2800   52   21    6    8    2    3    7    1    0]
 [  67    6    6   31   63 2222   43   12    6    1    3    3    9    1]
 [  13    5    6    3   27   32 1893    1   10    2    0    9    1    0]
 [  19  197    0    5    8   17    4 1488    3    0    1    0    1    1]
 [  29   11    2   28    5   14    3    2 1475    2    2   15    0    0]
 [  77    4    4    8    1    3    1    0    0 1101    0    0    0    0]
 [   0   11    0    0    4    6    1    1    2    0  975    0    0    0]
 [   6    2    1   14    3    0    1    0    7    1    0  588    0    2]
 [   1    1   14    1    0    9    0    0    0    0    0    0  350    0]
 [   3    0    0    4    1    1    1    0    3    0    0    3    0  135]]
```

每一个新闻类别的precision、recall、f1-score以及总的accuracy为如下：

	precision	recall	f1-score	support
0	0.95	0.95	0.95	7741
1	0.95	0.96	0.95	7308
2	0.99	0.99	0.99	6294
3	0.96	0.97	0.97	4455
4	0.91	0.92	0.91	3044
5	0.90	0.90	0.90	2473
6	0.95	0.95	0.95	2002
7	0.90	0.85	0.88	1744
8	0.94	0.93	0.94	1588
9	0.94	0.92	0.93	1199
10	0.99	0.97	0.98	1000
11	0.92	0.94	0.93	625
12	0.95	0.93	0.94	376
13	0.95	0.89	0.92	151
accuracy			0.95	40000
macro avg	0.94	0.93	0.94	40000
weighted avg	0.95	0.95	0.95	40000

其中accuracy指标最高值为0.95，最低值为0.95；precision（macro）指标最高值为0.95最，低值为0.94；recall（macro）指标最高值为0.94，最低值为0.93；f1-score（macro）指标最高值为0.95，最低值为0.94。

3.2 Bert+softmax

3.2.1 模型选取理由

为了对新闻文本进行分类，我们也可以将句子进行编码，构建属于句子的表示，即词向量，随后通过softmax进行分类。

为了将句子编码成词向量，我们可以考虑许多模型，如word2vec等，但考虑到需要考虑整句的前后联系与上下文，采用BERT编码成上下文相关词向量会使得编码词向量的代表性更强。因此我们采用BERT编码文本，随后通过softmax进行多分类。

3.2.2 实验步骤

（1）读取训练集。由于BERT的最大输入长度是512，而样本中最长的样本远超这个数字，因此我们需要对文本进行裁剪得到可以供BERT使用的数据集。本次实验取文本的头与尾的部分字符，加上拼接的token，长度为256。

（2）将得到数据集的输入BERT模型后，通过softmax进行训练与测试。

3.2.3 实验结果

分别在大小为20000的训练集和200000的训练集上分别进行了训练，并进行预测，最后的结果如下所示：

上方为在全部训练集上进行训练的模型得到的结果，下方为在大小为20000的训练集上进行训练的模型得到的结果。

由此可见，该方法对于数据集的大小有一定的要求，数据集足够大能够得到很好的结果。

4. 挖掘实验的结果

TFIDF+SVM线性模型的分类结果如上面所示。

4.1 线上测试结果

我们用代替提供的20万训练集训练完我们的模型后对对赛题提供的大小为5万的测试集A进行了测试并提交结果到天池赛题官网，测试结果为如下（评价指标为F1值）：

4.2 实验结果分析

从第三小节中5次验证集的实验结果很接近，没有很大差别，所以我们选取第1折的实验结果对它进行分析。

混淆矩阵输出：

```
[[7401  102   11   45   68   93   20   15   24   59    3    2    0    0]
 [ 100 7088    2    7   81    6    1  125    5    1    4    0    1    0]
 [   8    4 6213   19    2   11    3    0    4    0    1    4    2    0]
 [  40    7   22 4318   25   33    5    1   13    5    1   12    0    0]
 [  51   51   14   17 2785   48   19    8    3    1    1    6    1    1]
 [  62    4   10   33   76 2214   33    6    5    0    6    0    7    0]
 [  18    5    6    4   29   43 1843    4    7    2    1    1    0    1]
 [  22  161    0    5   10    7    3 1476    6    0    2    1    1    0]
 [  31   12    3   19    1    6    3    3 1476    1    0    8    0    1]
 [  79    1    1    6    2    7    1    0    1 1046    0    0    0    0]
 [   2    5    0    3    3    5    0    1    5    0  958    2    0    0]
 [   8    1    0   16    5    3    0    0   11    0    0  589    0    3]
 [   0    0   18    0    0    8    0    0    0    0    0    0  325    0]
 [   0    0    1    2    0    2    3    0    1    0    0    3    0  172]]
```

从第1折的混淆矩阵图中看到很类别被错误地预测成'科技'类（红色框线），同时'科技'类相对于其他类预测错误（蓝色框线）的数量也较多，我们认为出现这一情况的原因有

（i）'科技'类的新闻文本数量多且特征词更广泛且经典，因此某些跟'科技'类有类似特征项的被错误地预测；

（ii）'科技' 类的新闻文本数目最多，因此个别文本没有科技类典型特征而错误地被预测 。

同时从混淆矩阵中看到'财经'被误预测成'体育'类的（橙色框线）相对很突出，我们认为这是因为通常报道财经相关的新闻或多或少很跟体育竞技相关，因此'财经'类中的部分文本跟体育相关而预测成'体育'类。

	precision	recall	f1-score	support
0	0.95	0.94	0.94	7843
1	0.95	0.96	0.95	7421
2	0.99	0.99	0.99	6271
3	0.96	0.96	0.96	4482
4	0.90	0.93	0.91	3006
5	0.89	0.90	0.90	2456
6	0.95	0.94	0.95	1964
7	0.90	0.87	0.89	1694
8	0.95	0.94	0.94	1564
9	0.94	0.91	0.93	1144
10	0.98	0.97	0.98	984
11	0.94	0.93	0.93	636
12	0.96	0.93	0.94	351
13	0.97	0.93	0.95	184
accuracy			0.95	40000
macro avg	0.94	0.94	0.94	40000
weighted avg	0.95	0.95	0.95	40000

我们从第1折的评估指标中看到，'体育'类新闻文本的分类结果最客观，而'财经'类别的新闻文本的分类结果相对来说较差，我们假设'体育'类别的特征项比较特有且经典，而'财经'类别设计的范围比较广，因此有了上述的实验结果。