In [1]:

```
# 读取训练集
import pandas as pd
train_df = pd.read_csv('./data/train_set.csv', sep='\t')
```

In [2]:

```
# 查看训练集
train_df
```

Out[2]:

| | label | text |
|---|---|---|
| **0** | 2 | 2967 6758 339 2021 1854 3731 4109 3792 4149 15... |
| **1** | 11 | 4464 486 6352 5619 2465 4802 1452 3137 5778 54... |
| **2** | 3 | 7346 4068 5074 3747 5681 6093 1777 2226 7354 6... |
| **3** | 2 | 7159 948 4866 2109 5520 2490 211 3956 5520 549... |
| **4** | 3 | 3646 3055 3055 2490 4659 6065 3370 5814 2465 5... |
| **...** | ... | ... |
| **199995** | 2 | 307 4894 7539 4853 5330 648 6038 4409 3764 603... |
| **199996** | 2 | 3792 2983 355 1070 4464 5050 6298 3782 3130 68... |
| **199997** | 11 | 6811 1580 7539 1252 1899 5139 1386 3870 4124 1... |
| **199998** | 2 | 6405 3203 6644 983 794 1913 1678 5736 1397 191... |
| **199999** | 3 | 4350 3878 3268 1699 6909 5505 2376 2465 6088 2... |

200000 rows × 2 columns

In [3]:

```
# 统计每条样本的字符个数并做统计
%pylab inline
train_df['text_len'] = train_df['text'].apply(lambda x: len(x.split(' ')))
print(train_df['text_len'].describe())
```

```
Populating the interactive namespace from numpy and matplotlib
count    200000.000000
mean        907.207110
std         996.029036
min           2.000000
25%         374.000000
50%         676.000000
75%        1131.000000
max       57921.000000
Name: text_len, dtype: float64
```

由统计结果可知，文本最长的字符为57921个，最短的为2个，平均字符长度为907个

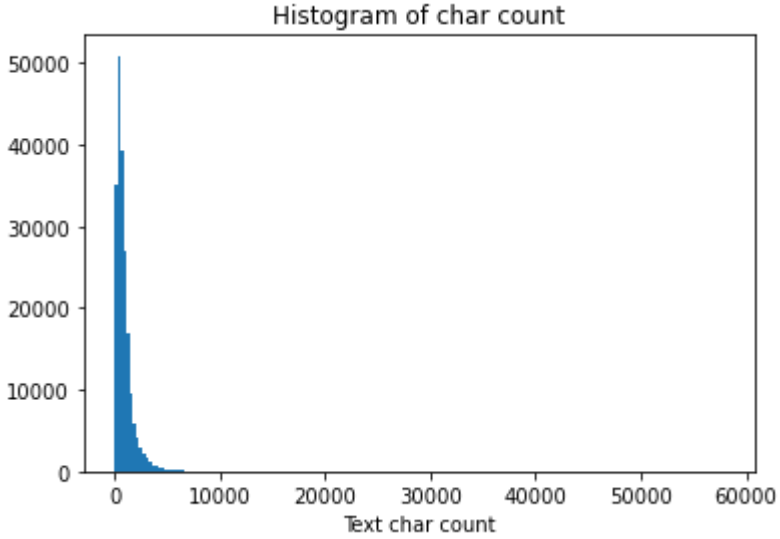
In [4]:

```
# 查看
_ = plt.hist(train_df['text_len'], bins=200)
plt.xlabel('Text char count')
plt.title("Histogram of char count")

```

Out[4]:

```
Text(0.5, 1.0, 'Histogram of char count')
```



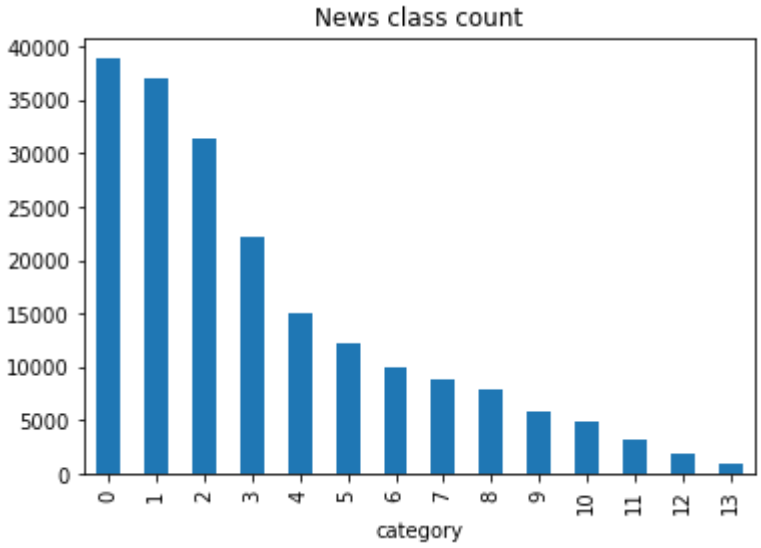

由直方图可知，文本字符长度集中在2000左右

In [5]:

```
train_df['label'].value_counts().plot(kind='bar')
plt.title('News class count')
plt.xlabel("category")
```

Out[5]:

```
Text(0.5, 0, 'category')
```




{'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': 11, '彩票': 12, '星座': 13}
从统计结果看，科技类新闻的样本数量最多，星座类新闻样本数最少，样本比例不均衡

In [6]:

```
from collections import Counter #counter 统计完为字典形式

```

In [ ]:

```

```

In [7]:

```
all_data=[]
train_data=train_df['text'].values.tolist()
for i,data in enumerate(train_data):
    data=data.split()
    if data:
        if i%10000==0:
            print("第%d条数据"%(i+1))
        all_data.extend(data)
```

```
第1条数据
第10001条数据
第20001条数据
第30001条数据
第40001条数据
第50001条数据
第60001条数据
第70001条数据
第80001条数据
第90001条数据
第100001条数据
第110001条数据
第120001条数据
第130001条数据
第140001条数据
第150001条数据
第160001条数据
第170001条数据
第180001条数据
第190001条数据
```

In [8]:

```
counter=Counter(all_data)
```

In [9]:

```
a=counter.most_common(3)
a
```

Out[9]:

```
[('3750', 7482224), ('648', 4924890), ('900', 3262544)]
```

In [10]:

```
_,s=zip(*a)
sum(s)/len(all_data)
```

Out[10]:

```
0.0863620766816962
```

In [11]:

```
len(all_data)
```

Out[11]:

181441422

In [14]:

```
sum(s)/len(train_df)
```

Out[14]:

78.34829

作业1：假设3750 、900和648 都是标点符号，那么假设一个标点符号前面一个句子，每篇新闻的平均有78个句子

In [72]:

```
del all_data
```

In [82]:

```

def split_data(train_data):
    all_data=[]
    for i,data in enumerate(train_data):
        data=data.split()
        if data:
            all_data.extend(data)
    return all_data
```

In [83]:

```
for i in range(14):
    data=train_df.loc[train_df['label']==i]
    data=data.text.values.tolist()
    alldata=split_data(data)
    count=Counter(alldata).most_common(4)
    print("第%d类新闻的频率最高的前4个为：%(i+1),count)

```

```
第1类新闻的频率最高的前4个为： [('3750', 1267331), ('648', 967653), ('900', 577742),
('3370', 503768)]
第2类新闻的频率最高的前4个为： [('3750', 1200686), ('648', 714152), ('3370', 62670
8), ('900', 542884)]
第3类新闻的频率最高的前4个为： [('3750', 1458331), ('648', 974639), ('900', 618294),
('7399', 351894)]
第4类新闻的频率最高的前4个为： [('3750', 774668), ('648', 494477), ('900', 298663),
('6122', 187933)]
第5类新闻的频率最高的前4个为： [('3750', 360839), ('648', 231863), ('900', 190842),
('4411', 120442)]
第6类新闻的频率最高的前4个为： [('3750', 715740), ('648', 329051), ('900', 305241),
('6122', 159125)]
第7类新闻的频率最高的前4个为： [('3750', 469540), ('648', 345372), ('900', 222488),
('6248', 193757)]
第8类新闻的频率最高的前4个为： [('3750', 428638), ('648', 262220), ('900', 184131),
('3370', 159156)]
第9类新闻的频率最高的前4个为： [('3750', 242367), ('648', 202399), ('900', 92207),
('6122', 57345)]
第10类新闻的频率最高的前4个为： [('3750', 178783), ('648', 157291), ('900', 70680),
('7328', 46477)]
第11类新闻的频率最高的前4个为： [('3750', 180259), ('648', 114512), ('900', 75185),
('3370', 67780)]
第12类新闻的频率最高的前4个为： [('3750', 83834), ('648', 67353), ('900', 37240),
('4939', 18591)]
第13类新闻的频率最高的前4个为： [('3750', 87412), ('4464', 51426), ('3370', 45815),
('648', 37041)]
第14类新闻的频率最高的前4个为： [('3750', 33796), ('648', 26867), ('900', 11263),
('4939', 9651)]
```

In [4]:

```
all_data_unique=[]
train_data_unique=train_df['text_unique'].values.tolist()
for i,data in enumerate(train_data_unique):
    data=data.split()
    if data:
#         if i%10000==0:
#             print("第%d条数据"%(i+1))
        all_data_unique.extend(data)
```

In [6]:

```
len(all_data_unique)
```

Out[6]:

```
56074040
```

In [10]:

```
from collections import Counter
counter=Counter(all_data_unique)
```

In [11]:

```
counter.most_common(3)
```

Out[11]:

```
[('3750', 197997), ('900', 197653), ('648', 191975)]
```

去重前后， 3750 、900和648 都是占比最大的前三位，所以极有可能是标点符号。