In [1]:
```python
import pandas as pd
import glob
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
plt.style.use('ggplot')
```

In [2]:
```python
category_list = ["sport", "world", "us", "business", "health", "entertainment", "sci_tech"]
directory_list = ["data/sport/*.txt",
                  "data/world/*.txt",
                   "data/us/*.txt",
                  "data/business/*.txt",
                  "data/health/*.txt",
                  "data/entertainment/*.txt",
                  "data/sci_tech/*.txt",]

text_files = list(map(lambda x: glob.glob(x), directory_list))
text_files = [item for sublist in text_files for item in sublist]
len(text_files)
```

Out[2]:
```
1433
```

In [3]:
```python
training_data = []


for t in text_files:
    f = open(t, 'r')
    f = f.read()
    t = f.split('\n')
    training_data.append({'data' : t[0] + ' ' + t[1], 'flag' : category_list.index(t[6]),'category' : t[6]})

training_data[0]
```

Out[3]:
```
{'data': "court agrees to expedite n.f.l.'s appeal the decision means a ruling could be made nearly two months
before the regular season begins, time for the sides to work out a deal without delaying the season.",
 'flag': 0,
 'category': 'sport'}
```

In [4]:
```python
training_data[0:3]
```

Out[4]:
```
[{'data': "court agrees to expedite n.f.l.'s appeal the decision means a ruling could be made nearly two months
before the regular season begins, time for the sides to work out a deal without delaying the season.",
  'flag': 0,
  'category': 'sport'},
 {'data': 'no tsunami but fifa\'s corruption storm rages on though jack warner\'s threatened soccer "tsunami" r
emains stuck in the doldrums, the corruption storm raging around fifa shows no sign of abating after another ex
traordinary week for the game\'s governing body.',
  'flag': 0,
  'category': 'sport'},
 {'data': 'chung backs bin hammam in fifa presidential race mohamed bin hammam received a boost to his campaign
to become fifa president when he was backed by his former foe and ex-fifa executive committee member chung mong
-joon tuesday.',
  'flag': 0,
  'category': 'sport'}]
```

Create Data Frame

In [5]:
```python
training_data = pd.DataFrame(training_data, columns=['data', 'flag','category'])

print(training_data.shape)
```

```
(1433, 3)
```

In [6]:
```python
training_data.head()
```

Out[6]:

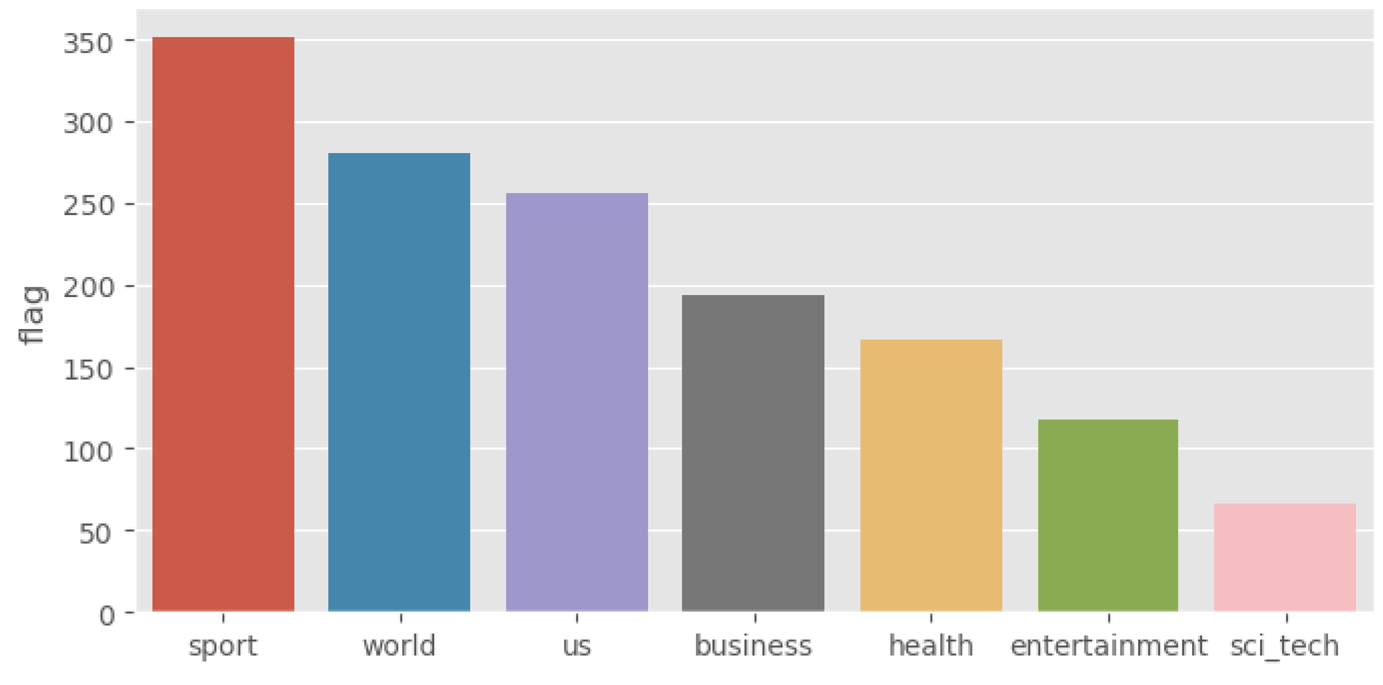
	data	flag	category
0	court agrees to expedite n.f.l.'s appeal the d...	0	sport
1	no tsunami but fifa's corruption storm rages o...	0	sport
2	chung backs bin hammam in fifa presidential ra...	0	sport
3	rory mcilroy hangs on to slim lead at augusta ...	0	sport
4	figure skating: south korean skater stumbles, ...	0	sport

Count the frequency of each news articles in diff categories

In [7]:
```python
training_data.flag.value_counts()
```

Out[7]:
```
0    352
1    280
3    256
2    194
5    167
6    118
4     66
Name: flag, dtype: int64
```

In [8]:
```python
plt.figure(figsize=(8, 4), dpi=100, facecolor='w', edgecolor='k')
sns.barplot(x=category_list, y=training_data.flag.value_counts())
plt.show()
```



Feature extraction

In [9]:
```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(training_data.data, training_data.flag, random_state=1)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(1074,) (359,) (1074,) (359,)
```

In [10]:
```python
import pickle
from sklearn.feature_extraction.text import CountVectorizer


#GET VECTOR COUNT
count_vect = CountVectorizer(strip_accents='ascii', token_pattern=u'(?ui)\\b\\w*[a-z]+\\w*\\b', lowercase=True,
X_train_cv = count_vect.fit_transform(X_train)
X_test_cv = count_vect.transform(X_test)

#SAVE WORD VECTOR
#pickle.dump(count_vect.vocabulary_, open("count_vector.pkl","wb"))
```

In [11]:
```python
X_train_cv
```

Out[11]:
```
<1074x7156 sparse matrix of type '<class 'numpy.int64'>'
	with 19847 stored elements in Compressed Sparse Row format>
```

Fit the model and make predictions

In [12]:
```python
from sklearn.naive_bayes import MultinomialNB
naive_bayes = MultinomialNB()
naive_bayes.fit(X_train_cv, y_train)
```

Out[12]:
```
MultinomialNB()
```

Predictions

In [13]:
```python
predictions_nb = naive_bayes.predict(X_test_cv)
print(predictions_nb)
```

```
[5 1 6 1 3 3 2 6 5 0 3 1 3 3 1 0 0 0 0 0 5 3 6 0 0 5 6 0 3 2 0 2 0 0 5 1
 5 3 3 5 0 0 1 0 0 1 5 5 1 1 3 1 0 6 0 0 1 3 3 0 2 5 1 6 0 3 1 3 0 3 3 2 5
 1 0 0 6 0 5 5 2 0 2 0 3 2 0 1 3 1 0 3 3 1 3 0 0 0 5 4 4 4 2 3 6 4 4 0 3 1
 3 3 3 1 1 2 2 2 5 0 0 5 1 1 0 0 2 3 0 3 1 2 5 0 1 1 3 1 6 3 6 0 3 4 1 0 1
 2 1 2 1 0 1 1 0 0 5 1 3 2 0 3 5 1 3 3 5 3 1 1 2 5 0 0 6 3 5 2 3 5 0 0 0 6
 1 3 1 2 0 3 3 3 1 1 1 2 2 3 4 0 2 0 5 2 1 3 2 4 0 5 0 3 0 0 5 1 6 3 1 0 5
 0 0 0 0 2 0 3 0 5 0 6 1 0 0 1 0 3 1 0 5 1 2 1 5 2 0 1 2 0 0 1 0 3 0 3 0 2
 1 2 4 3 3 0 6 6 1 0 2 0 0 1 0 1 5 1 3 1 0 6 3 2 1 0 3 2 3 3 2 1 5 1 2 0 3
 6 0 1 6 0 0 3 1 3 0 5 2 2 6 3 5 1 0 2 0 3 2 1 1 5 1 6 0 2 2 0 1 1 1 0 1 1
 0 3 4 0 5 0 6 5 0 3 2 0 1 0 5 0 3 6 5 0 3 5 0 3 5 0]
```

In [14]:
```python
df = pd.DataFrame({"Actual":y_test, "Predicted":predictions_nb})
df.head()
```

Out[14]:

	Actual	Predicted
1195	5	5
494	1	1
820	2	6
547	1	1
1035	3	3

In [15]:
```python
category_list
```

Out[15]:
```
['sport', 'world', 'us', 'business', 'health', 'entertainment', 'sci_tech']
```

In [16]:
```python
df1 = df.replace(0,category_list[0])
df2 = df1.replace(1,category_list[1])
df3 = df2.replace(2,category_list[2])
df4 = df3.replace(3,category_list[3])
df5 = df4.replace(4,category_list[4])
df6 = df5.replace(5,category_list[5])
final_df = df6.replace(6,category_list[6])
final_df.head(10)
```

Out[16]:

	Actual	Predicted
1195	entertainment	entertainment
494	world	world
820	us	sci_tech
547	world	world
1035	business	business
48	sport	business
824	us	us
1335	sci_tech	sci_tech
867	business	entertainment
204	sport	sport

KNN Algorithm

In [17]:
```python
import pickle
# make a k- nearest model
from sklearn.neighbors import KNeighborsClassifier
clf_k = KNeighborsClassifier(n_neighbors=32)

model2 = clf_k.fit(X_train_cv, y_train)
predictions = model2.predict(X_test_cv)
print(predictions)
#save model
#pickle.dump(model2, open("model2.pkl", "wb"))
```

```
[5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 0 5 5
 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 1 5 5 5 5 5 5 5 3 5 5
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 5 5 5 5 5 1 5 5 5 5 5 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5
 5 5 5 0 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5
 5 5 5 5 5 5 5 5 5 5 5 1 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 5 5 5 5 5 0 5 5 5 0 5 5 0 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0]
```

Check the results for `Naive-Bayes`

In [18]:
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score,confusion_matrix

print("Accuracy score: ", accuracy_score(y_test, predictions_nb))
print("Precision score: ", precision_score(y_test, predictions_nb,
                                           pos_label='positive',average='micro'))
print("Recall score: ", recall_score(y_test, predictions_nb,
                                           pos_label='positive',average='micro'))
```

```
Accuracy score:  0.7409470752089137
Precision score:  0.7409470752089137
Recall score:  0.7409470752089137
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1295: UserWarning: Note that pos_l
abel (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to s
pecify a single positive class.
  warnings.warn("Note that pos_label (set to %r) is ignored when "
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1295: UserWarning: Note that pos_l
abel (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to s
pecify a single positive class.
  warnings.warn("Note that pos_label (set to %r) is ignored when "
```

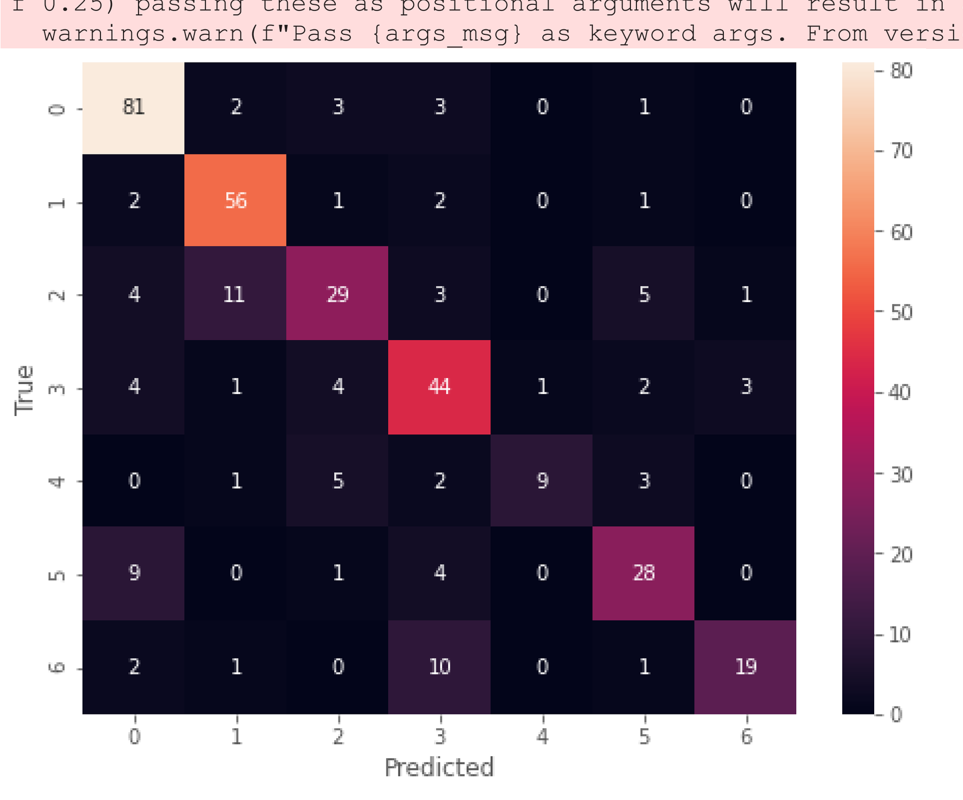
Confusion Matrix

In [19]:
```python
conf_mat = confusion_matrix(
        final_df.Actual, final_df.Predicted,labels=['sport', 'world', 'us', 'business', 'health', 'entertainmen
conf_mat
```

Out[19]:
```
array([[81,  2,  3,  3,  0,  1,  0],
       [ 2, 56,  1,  2,  0,  1,  0],
       [ 4, 11, 29,  3,  0,  5,  1],
       [ 4,  1,  4, 44,  1,  2,  3],
       [ 0,  1,  5,  2,  9,  3,  0],
       [ 9,  0,  1,  4,  0, 28,  0],
       [ 2,  1,  0, 10,  0,  1, 19]], dtype=int64)
```

In [20]:
```python
labels=['sport', 'world', 'us', 'business', 'health', 'entertainment', 'sci_tech']
cm = confusion_matrix(final_df.Actual, final_df.Predicted, labels)
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\utils\validation.py:70: FutureWarning: Pass labels=['sport',
'world', 'us', 'business', 'health', 'entertainment', 'sci_tech'] as keyword args. From version 1.0 (renaming o
f 0.25) passing these as positional arguments will result in an error
  warnings.warn(f"Pass {args_msg} as keyword args. From version "
```



Check the results for `KNN`

In [21]:
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score

print("Accuracy score: ", accuracy_score(y_test, predictions))
print("Precision score: ", precision_score(y_test, predictions,
                                           pos_label='positive',average='micro'))
print("Recall score: ", recall_score(y_test, predictions,
                                           pos_label='positive',average='micro'))
```

```
Accuracy score:  0.17270194986072424
Precision score:  0.17270194986072424
Recall score:  0.17270194986072424
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1295: UserWarning: Note that pos_l
abel (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to s
pecify a single positive class.
  warnings.warn("Note that pos_label (set to %r) is ignored when "
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1295: UserWarning: Note that pos_l
abel (set to 'positive') is ignored when average != 'binary' (got 'micro'). You may use labels=[pos_label] to s
pecify a single positive class.
  warnings.warn("Note that pos_label (set to %r) is ignored when "
```

* HAPPY LEARNINIG **

In [ ]: