In [1]:
```python
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
# Load the dataset
newsgroups = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)
newsgroups
```

Out[2]:

```
{'data': ["From: Mamatha Devineni Ratnam <mr47+@andrew.cmu.edu>\nSubject: Pens fans reactions\nOrganization: Post O
ffice, Carnegie Mellon, Pittsburgh, PA\nLines: 12\nNNTP-Posting-Host: po4.andrew.cmu.edu\n\n\n\nI am sure some bash
ers of Pens fans are pretty confused about the lack\nof any kind of posts about the recent Pens massacre of the Dev
ils. Actually,\nI am  bit puzzled too and a bit relieved. However, I am going to put an end\nto non-PIttsburghers'
relief with a bit of praise for the Pens. Man, they\nare killing those Devils worse than I thought. Jagr just showe
d you why\nhe is much better than his regular season stats. He is also a lot\nfo fun to watch in the playoffs. Bowm
an should let JAgr have a lot of\nfun in the next couple of games since the Pens are going to beat the pulp out of
Jersey anyway. I was very disappointed not to see the Islanders lose the final\nregular season game.          PENS
RULE!!!\n\n",
  'From: mblawson@midway.ecn.uoknor.edu (Matthew B Lawson)\nSubject: Which high-performance VLB video card?\nSummar
y: Seek recommendations for VLB video card\nNntp-Posting-Host: midway.ecn.uoknor.edu\nOrganization: Engineering Com
puter Network, University of Oklahoma, Norman, OK, USA\nKeywords: orchid, stealth, vlb\nLines: 21\n\n  My brother i
s in the market for a high-performance video card that supports\nVESA local bus with 1-2MB RAM.  Does anyone have s
uggestions/ideas on:\n\n  - Diamond Stealth Pro Local Bus\n\n  - Orchid Farenheit 1280\n\n  - ATI Graphics Ultra Pr
o\n\n  - Any other high-performance VLB card\n\n\nPlease post or email.  Thank you!\n\n  - Matt\n\n-- \n    |  Matt
hew B. Lawson <------------> (mblawson@essex.ecn.uoknor.edu)  |   \n  --+-- "Now I, Nebuchadnezzar, praise and exal
t and glorify the King  --+-- \n    |   of heaven, because everything he does is right and all his ways  |   \n
|   are just." - Nebuchadnezzar, king of Babylon, 562 B.C.           |   \n',
  'From: hilmi-er@dsv.su.se (Hilmi Eren)\nSubject: Re: ARMENIA SAYS IT COULD SHOOT DOWN TURKISH PLANES (Henrik)\nLi
```


In [3]:
```python
text_categories = newsgroups.target_names
text_categories

# Print the text categories
print("Text Categories:")
for i, category in enumerate(text_categories):
    print(f"{i}: {category}")

print("Number of unique classes: {}".format(len(text_categories)))
```

```
Text Categories:
0: alt.atheism
1: comp.graphics
2: comp.os.ms-windows.misc
3: comp.sys.ibm.pc.hardware
4: comp.sys.mac.hardware
5: comp.windows.x
6: misc.forsale
7: rec.autos
8: rec.motorcycles
9: rec.sport.baseball
10: rec.sport.hockey
11: sci.crypt
12: sci.electronics
13: sci.med
14: sci.space
15: soc.religion.christian
16: talk.politics.guns
17: talk.politics.mideast
18: talk.politics.misc
19: talk.religion.misc
Number of unique classes: 20
```

In [4]:
```python
# Split the dataset
train_data, test_data, y_train, y_test = train_test_split(newsgroups.data, newsgroups.target, test_size=0.2, random_
```

In [5]:
```python
# Create a pipeline that combines the TfidfVectorizer and the MultinomialNB classifier
model = make_pipeline(TfidfVectorizer(stop_words='english'), MultinomialNB())
model
```

Out[5]:
```
▸ Pipeline
  ▸ TfidfVectorizer
  ▸ MultinomialNB
```

In [6]:
```
# Train the model
model.fit(train_data, y_train)
```

Out[6]:
```
▸ Pipeline
▸ TfidfVectorizer
▸ MultinomialNB
```

In [7]:
```
# Make predictions
y_pred = model.predict(test_data)
y_pred
```

Out[7]:
```
array([ 9, 12, 14, ...,  0,  0, 14])
```

In [8]:
```
# Evaluate the model
accuracy = metrics.accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 87.85%
```

In [9]:
```
# Print classification report
print(metrics.classification_report(y_test, y_pred, target_names=newsgroups.target_names))
```

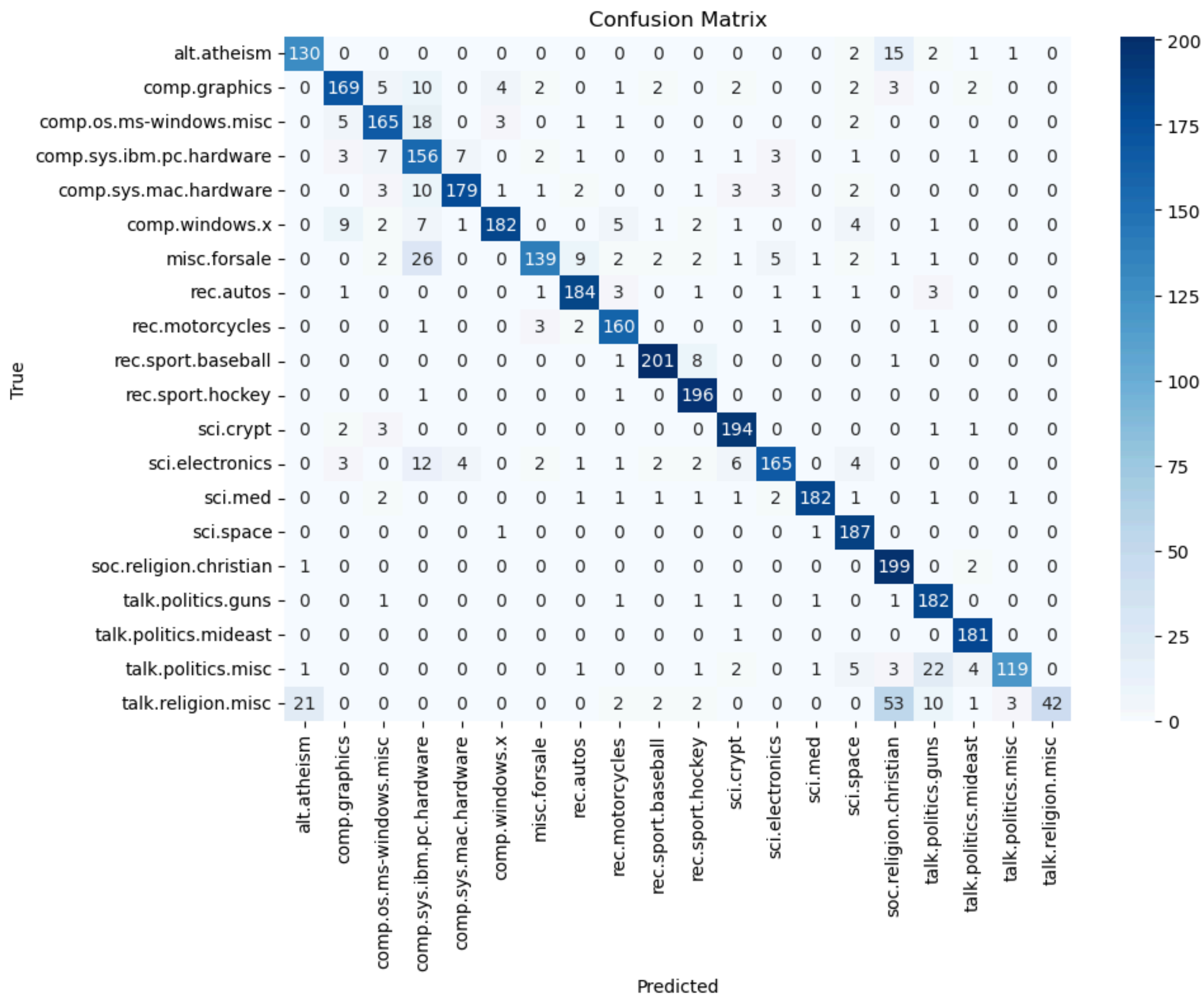
```
                          precision    recall  f1-score   support

             alt.atheism       0.85      0.86      0.86       151
           comp.graphics       0.88      0.84      0.86       202
 comp.os.ms-windows.misc       0.87      0.85      0.86       195
comp.sys.ibm.pc.hardware       0.65      0.85      0.74       183
   comp.sys.mac.hardware       0.94      0.87      0.90       205
          comp.windows.x       0.95      0.85      0.90       215
            misc.forsale       0.93      0.72      0.81       193
               rec.autos       0.91      0.94      0.92       196
         rec.motorcycles       0.89      0.95      0.92       168
      rec.sport.baseball       0.95      0.95      0.95       211
        rec.sport.hockey       0.90      0.99      0.94       198
               sci.crypt       0.91      0.97      0.94       201
         sci.electronics       0.92      0.82      0.86       202
                 sci.med       0.97      0.94      0.96       194
               sci.space       0.88      0.99      0.93       189
  soc.religion.christian       0.72      0.99      0.83       202
      talk.politics.guns       0.81      0.97      0.88       188
   talk.politics.mideast       0.94      0.99      0.97       182
      talk.politics.misc       0.96      0.75      0.84       159
      talk.religion.misc       1.00      0.31      0.47       136

                accuracy                           0.88      3770
               macro avg       0.89      0.87      0.87      3770
            weighted avg       0.89      0.88      0.87      3770
```

In [10]:
```
# Compute confusion matrix
conf_matrix = metrics.confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=newsgroups.target_names, yticklabels=newsgro
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```



In [11]:
```python
# Print train and test data with their respective categories
print("\nTrain Data Categories:")
for i in range(len(train_data)):
    print(f"Document {i}: Category - {text_categories[y_train[i]]}")

print("\nTest Data Categories:")
for i in range(len(test_data)):
    print(f"Document {i}: Category - {text_categories[y_test[i]]}")
```

```
Document 806: Category - comp.windows.x
Document 807: Category - comp.sys.mac.hardware
Document 808: Category - sci.med
Document 809: Category - rec.motorcycles
Document 810: Category - alt.atheism
Document 811: Category - comp.os.ms-windows.misc
Document 812: Category - talk.politics.guns
Document 813: Category - comp.sys.ibm.pc.hardware
Document 814: Category - rec.sport.hockey
Document 815: Category - misc.forsale
Document 816: Category - sci.space
Document 817: Category - sci.electronics
Document 818: Category - comp.sys.mac.hardware
Document 819: Category - rec.sport.hockey
Document 820: Category - comp.graphics
Document 821: Category - talk.politics.mideast
Document 822: Category - talk.politics.misc
Document 823: Category - talk.politics.mideast
Document 824: Category - talk.politics.guns
Document 825: Category - rec.sport.hockey
```

In [ ]: