In [4]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import re
import string
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

file_path = r"C:\Users\AkshS\OneDrive\Desktop\data_news.xlsx"
df_news = pd.read_excel(file_path)

print("Columns:")
print(df_news.columns)

# View data
print(df_news.head())

# Check for nulls
print(df_news.isnull().sum())

# Drop rows with missing short descriptions
df_news.dropna(subset=['short_description'], inplace=True)

# Text length
df_news['text_length'] = df_news['short_description'].apply(lambda x: len(str(x)
print("\nText Length Stats:")
print(df_news['text_length'].describe())

# Category distribution
print(df_news['category'].value_counts())

# Plot
sns.countplot(data=df_news, y='category', order=df_news['category'].value_counts
plt.title("Article Category Distribution")
plt.show()

# Text cleaning
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'<.*?>', '', text)
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = re.sub(r'\d+', '', text)
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_wo
    return ' '.join(words)

df_news['clean_text'] = df_news['short_description'].apply(clean_text)
print(df_news[['short_description', 'clean_text']].head())
```

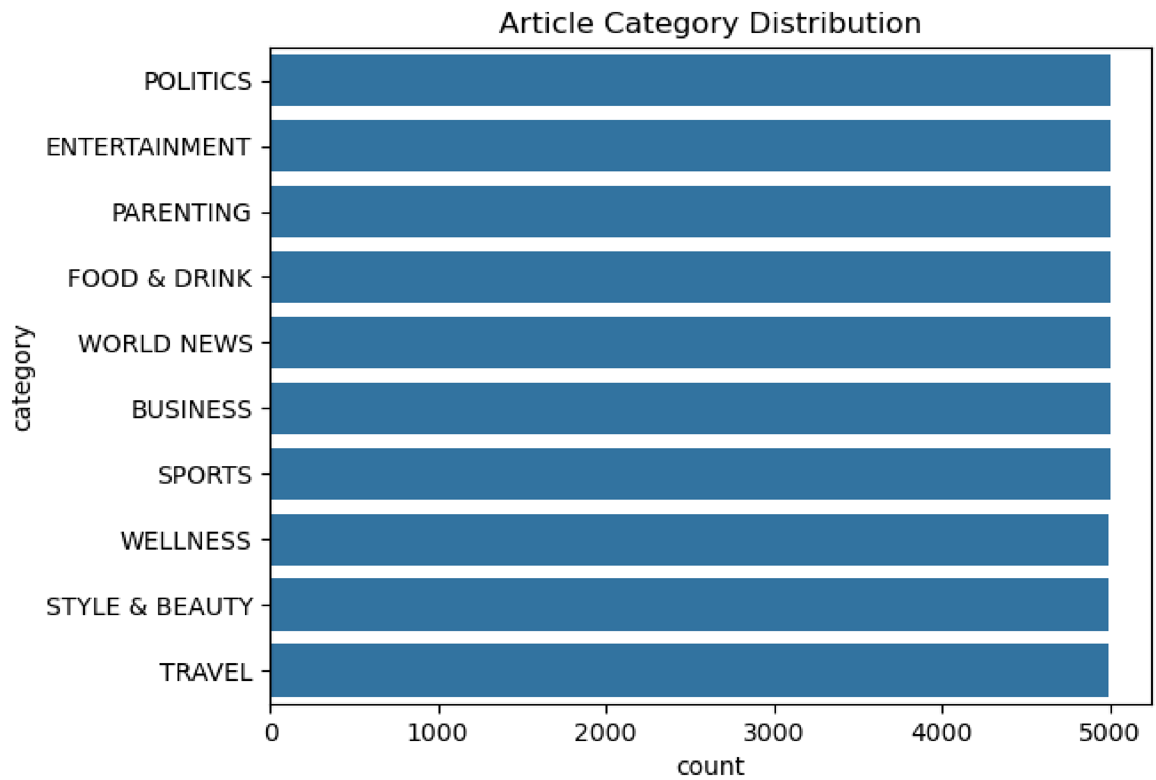
```
Columns:
Index(['category', 'headline', 'links', 'short_description', 'keywords'], dtype
='object')
   category                                           headline  \
0  WELLNESS              143 Miles in 35 Days: Lessons Learned
1  WELLNESS       Talking to Yourself: Crazy or Crazy Helpful?
2  WELLNESS  Crenezumab: Trial Will Gauge Whether Alzheimer...
3  WELLNESS                     Oh, What a Difference She Made
4  WELLNESS                                   Green Superfoods

                                               links  \
0  https://www.huffingtonpost.com/entry/running-l...
1  https://www.huffingtonpost.com/entry/talking-t...
2  https://www.huffingtonpost.com/entry/crenezuma...
3  https://www.huffingtonpost.com/entry/meaningfu...
4  https://www.huffingtonpost.com/entry/green-sup...

                                   short_description  \
0  Resting is part of training. I've confirmed wh...
1  Think of talking to yourself as a tool to coac...
2  The clock is ticking for the United States to ...
3  If you want to be busy, keep trying to be perf...
4  First, the bad news: Soda bread, corned beef a...

                              keywords
0                      running-lessons
1            talking-to-yourself-crazy
2  crenezumab-alzheimers-disease-drug
3                      meaningful-life
4                     green-superfoods
category                0
headline                0
links                   0
short_description       6
keywords             2706
dtype: int64

Text Length Stats:
count    49994.000000
mean        22.984538
std         13.589016
min          1.000000
25%         14.000000
50%         21.000000
75%         29.000000
max        222.000000
Name: text_length, dtype: float64
category
POLITICS          5000
ENTERTAINMENT     5000
PARENTING         5000
FOOD & DRINK      5000
WORLD NEWS        5000
BUSINESS          5000
SPORTS            5000
WELLNESS          4999
STYLE & BEAUTY    4999
TRAVEL            4996
Name: count, dtype: int64
```




```
                              short_description  \
0  Resting is part of training. I've confirmed wh...
1  Think of talking to yourself as a tool to coac...
2  The clock is ticking for the United States to ...
3  If you want to be busy, keep trying to be perf...
4  First, the bad news: Soda bread, corned beef a...

                                         clean_text
0  resting part training ive confirmed sort alrea...
1  think talking tool coach challenge narrate exp...
2  clock ticking united state find cure team work...
3  want busy keep trying perfect want happy focus...
4  first bad news soda bread corned beef beer hig...
```

In [6]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_news = TfidfVectorizer(max_features=5000)
X_news = vectorizer_news.fit_transform(df_news['clean_text'])

print("TF-IDF Matrix Shape:", X_news.shape)

# Additional features
df_news['char_count'] = df_news['short_description'].apply(lambda x: len(str(x)))
df_news['avg_word_length'] = df_news['char_count'] / df_news['text_length']

print(df_news[['text_length', 'char_count', 'avg_word_length']].head())
```

```
TF-IDF Matrix Shape: (49994, 5000)
   text_length  char_count  avg_word_length
0           49         280         5.714286
1           39         216         5.538462
2           24         120         5.000000
3           22         106         4.818182
4           24         125         5.208333
```

In [8]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC

# Target labels
y_news = df_news['category']

# Train-test split
X_train_news, X_test_news, y_train_news, y_test_news = train_test_split(X_news,

# Logistic Regression
lr_news = LogisticRegression(max_iter=1000)
lr_news.fit(X_train_news, y_train_news)

# Naive Bayes
nb_news = MultinomialNB()
nb_news.fit(X_train_news, y_train_news)

# SVM
svm_news = LinearSVC()
svm_news.fit(X_train_news, y_train_news)
```

Out[8]:
```
▾ LinearSVC
LinearSVC()
```

In [10]:
```
from sklearn.metrics import classification_report, accuracy_score, confusion_mat

def evaluate_model_news(model, name):
    print(f"\n{name} Evaluation:")
    y_pred = model.predict(X_test_news)
    print(classification_report(y_test_news, y_pred))
    print("Accuracy:", accuracy_score(y_test_news, y_pred))
    cm = confusion_matrix(y_test_news, y_pred)
    plt.figure(figsize=(8,6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{name} Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

# Evaluate all models
evaluate_model_news(lr_news, "Logistic Regression")
evaluate_model_news(nb_news, "Naive Bayes")
evaluate_model_news(svm_news, "SVM")
```

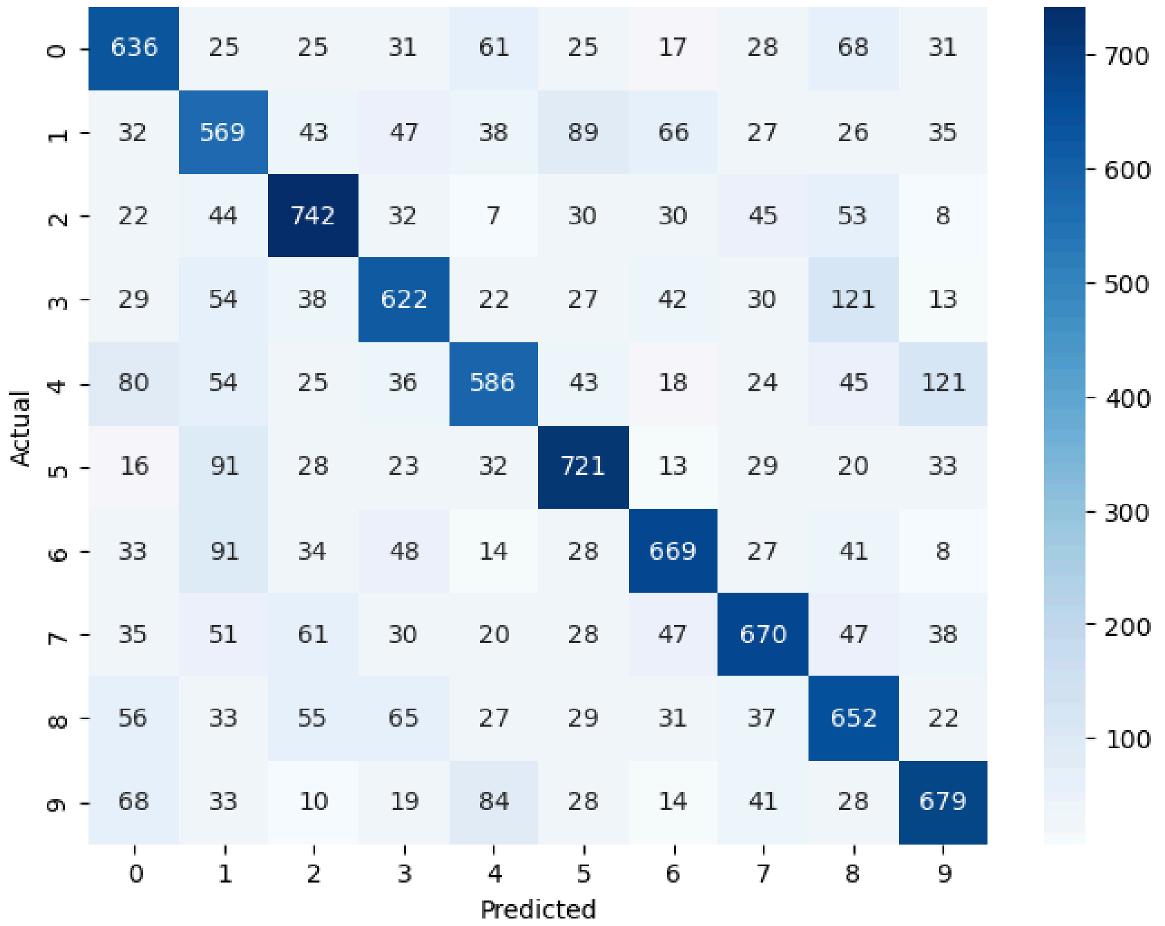
```
Logistic Regression Evaluation:
                precision    recall  f1-score   support

      BUSINESS       0.63      0.67      0.65       947
 ENTERTAINMENT       0.54      0.59      0.56       972
  FOOD & DRINK       0.70      0.73      0.72      1013
     PARENTING       0.65      0.62      0.64       998
      POLITICS       0.66      0.57      0.61      1032
        SPORTS       0.69      0.72      0.70      1006
STYLE & BEAUTY       0.71      0.67      0.69       993
        TRAVEL       0.70      0.65      0.68      1027
      WELLNESS       0.59      0.65      0.62      1007
    WORLD NEWS       0.69      0.68      0.68      1004

      accuracy                           0.65      9999
     macro avg       0.66      0.65      0.65      9999
  weighted avg       0.66      0.65      0.65      9999

Accuracy: 0.6546654665466547
```

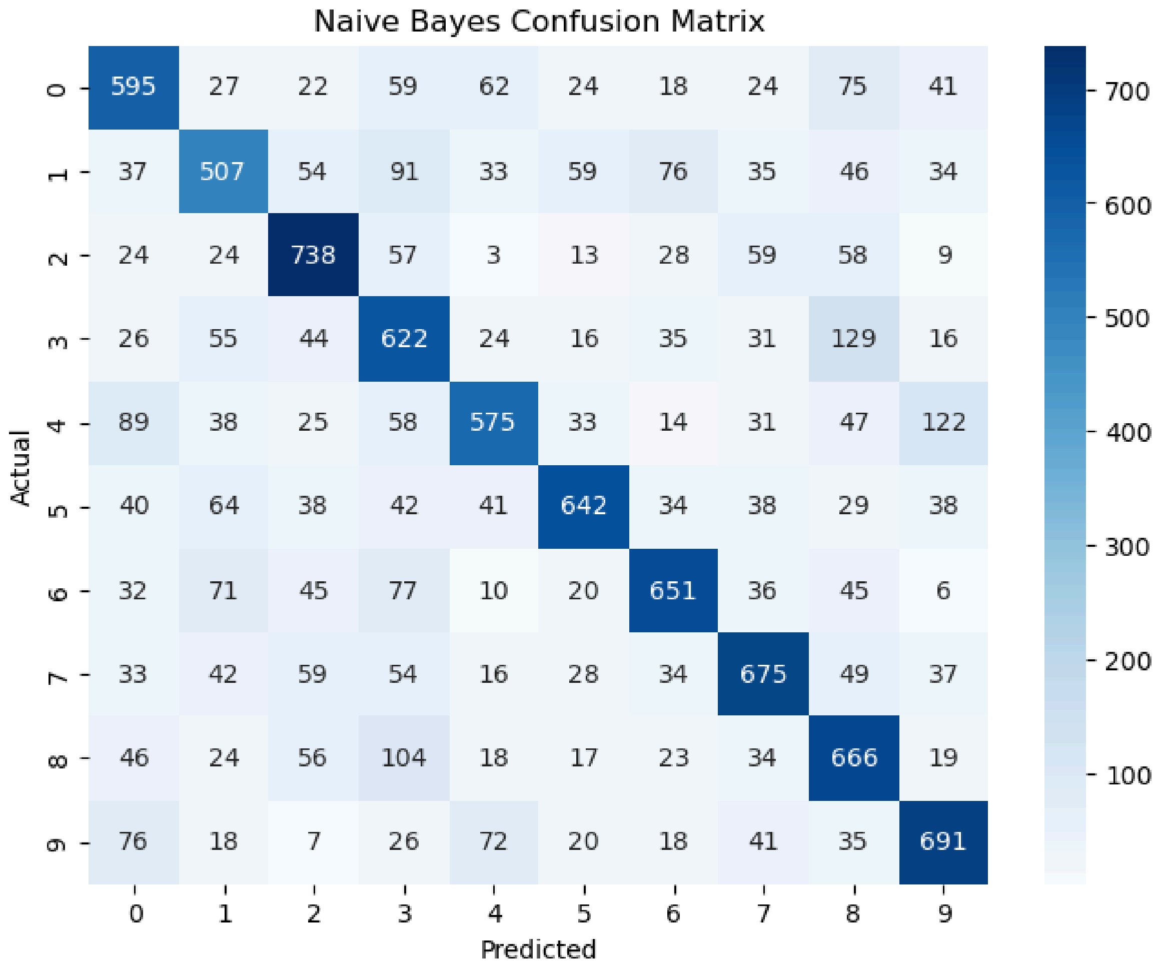
Logistic Regression Confusion Matrix



Naive Bayes Evaluation:

	precision	recall	f1-score	support
BUSINESS	0.60	0.63	0.61	947
ENTERTAINMENT	0.58	0.52	0.55	972
FOOD & DRINK	0.68	0.73	0.70	1013
PARENTING	0.52	0.62	0.57	998
POLITICS	0.67	0.56	0.61	1032
SPORTS	0.74	0.64	0.68	1006
STYLE & BEAUTY	0.70	0.66	0.68	993
TRAVEL	0.67	0.66	0.66	1027
WELLNESS	0.56	0.66	0.61	1007
WORLD NEWS	0.68	0.69	0.69	1004
accuracy			0.64	9999
macro avg	0.64	0.64	0.64	9999
weighted avg	0.64	0.64	0.64	9999

Accuracy: 0.6362636263626362

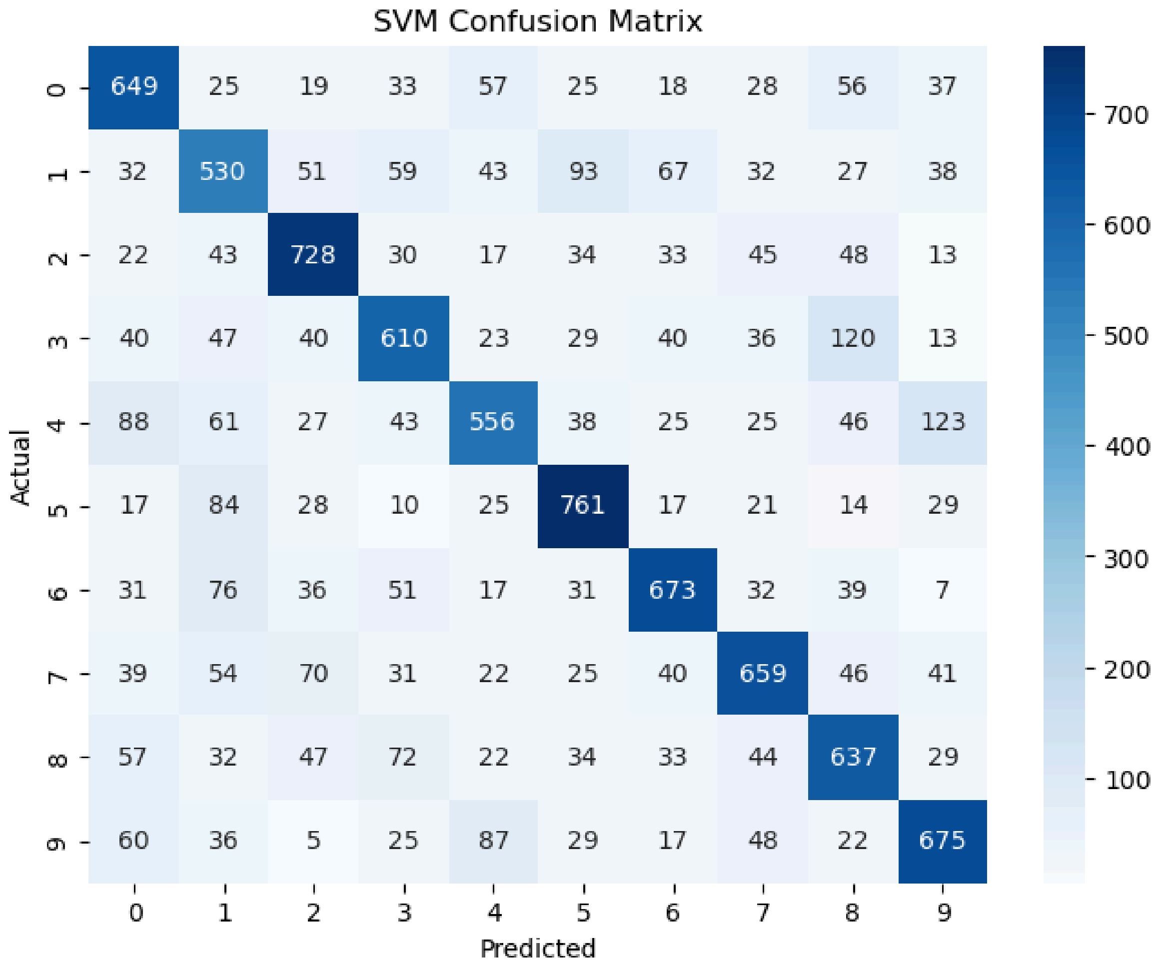


```
SVM Evaluation:
                precision    recall  f1-score   support

     BUSINESS       0.63      0.69      0.65       947
ENTERTAINMENT       0.54      0.55      0.54       972
 FOOD & DRINK       0.69      0.72      0.71      1013
    PARENTING       0.63      0.61      0.62       998
     POLITICS       0.64      0.54      0.58      1032
       SPORTS       0.69      0.76      0.72      1006
STYLE & BEAUTY      0.70      0.68      0.69       993
       TRAVEL       0.68      0.64      0.66      1027
     WELLNESS       0.60      0.63      0.62      1007
   WORLD NEWS       0.67      0.67      0.67      1004

     accuracy                           0.65      9999
    macro avg       0.65      0.65      0.65      9999
 weighted avg       0.65      0.65      0.65      9999

Accuracy: 0.6478647864786479
```



In [ ]:

In [ ]: