Project Overview

This project demonstrates **automated news topic classification** using NLP techniques. It processes raw newsgroup text, transforms it into structured features, and trains a model to categorize articles into 4 topics:

1. comp.graphics (Computer Graphics)
2. rec.sport.baseball (Baseball)
3. sci.space (Space Science)
4. talk.politics.mideast (Middle East Politics)

Key Components Explained

1. **Data Source**

- Uses the **20 Newsgroups Dataset** (via fetch_20newsgroups from scikit-learn).
- Filters 4 specific topics to create a focused classification task.
- Removes metadata (headers, footers, quotes) to force the model to analyze core content.

2. **Text Preprocessing**

- **Lowercasing**: Standardizes text (text.lower()).
- **Punctuation Removal**: Eliminates symbols (string.punctuation).
- **Tokenization**: Splits text into words (word_tokenize).
- **Stopword Removal**: Drops common words (e.g., "the", "and") using NLTK's stopwords.
- **Lemmatization**: Reduces words to root forms (e.g., "running" → "run") via WordNetLemmatizer.
- **Output**: Cleaned text as space-separated tokens.

3. **Feature Engineering**

- **TF-IDF Vectorization**: Converts text to numerical features.
    - Weighs words by importance (frequent in a document but rare across the corpus).
    - Outputs a sparse matrix of TF-IDF scores.

4. **Model**

- **Pipeline**: TfidfVectorizer → MultinomialNB (Naive Bayes classifier).
- **Why Naive Bayes?**
    - Efficient for high-dimensional text data.
    - Works well with TF-IDF features despite feature independence assumptions.

5. **Evaluation**

- **Metrics**:
    - Precision/Recall/F1-score per class.
    - Overall accuracy (89.22% in this run).
- **Performance Highlights**:
    - Best for talk.politics.mideast (F1=0.92).
    - Weakest for sci.space (F1=0.86).

Real-World Use Cases

- **News Aggregators**: Auto-tag articles for personalized feeds.
- **Content Moderation**: Flag off-topic/inappropriate content.
- **Knowledge Management**: Organize internal documents by topic.
- **SEO Optimization**: Analyze topic trends in user-generated content.

Conclusion

This project showcases a **fundamental NLP workflow** for topic classification. It highlights how combining linguistic preprocessing (NLTK) with statistical feature engineering (TF-IDF) and machine learning (Naive Bayes) can solve real-world text categorization problems efficiently. The 89% accuracy demonstrates viability for practical applications, with clear paths for further optimization.

NewsTopicClassifictionUsingNLP

June 1, 2025

[1]:
```
import nltk
import string
import numpy as np
import pandas as pd

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 Pipeline
from sklearn.metrics import classification_report, accuracy_score

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize
```

[2]:
```
# Download necessary NLTK data
nltk.download("punkt")
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("omw-1.4")
import nltk
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\itzsh\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\itzsh\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\itzsh\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\itzsh\AppData\Roaming\nltk_data…
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\itzsh\AppData\Roaming\nltk_data…
```

```
[nltk_data]   Package punkt_tab is already up-to-date!
```

[2]:
```
True
```

[3]:
```
# Load 20 Newsgroups Dataset
categories = ['sci.space', 'rec.sport.baseball', 'comp.graphics', 'talk.
 ↪politics.mideast']
newsgroups = fetch_20newsgroups(subset='all', categories=categories,␣
 ↪remove=('headers', 'footers', 'quotes'))
```

[4]:
```
# Preprocessing function
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
```

[5]:
```
def preprocess(text):
    # Lowercase
    text = text.lower()
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Tokenize
    tokens = word_tokenize(text)
    # Remove stopwords and lemmatize
    cleaned = [lemmatizer.lemmatize(word) for word in tokens if word not in␣
 ↪stop_words and word.isalpha()]
    return ' '.join(cleaned)
```

[6]:
```
# Preprocess all data
print("Preprocessing texts...")
cleaned_data = [preprocess(doc) for doc in newsgroups.data]
```

```
Preprocessing texts…
```

[7]:
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(cleaned_data, newsgroups.
 ↪target, test_size=0.2, random_state=42)
```

[8]:
```
# Build a pipeline: TF-IDF + Naive Bayes Classifier
model = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', MultinomialNB())
])
```

[9]:
```
# Train the model
print("Training the model...")
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
```

```python
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=newsgroups.
  ↪target_names))
print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
```

```
Training the model…

Classification Report:
                        precision    recall  f1-score   support

         comp.graphics       0.93      0.87      0.90       220
   rec.sport.baseball        0.85      0.94      0.89       178
             sci.space       0.88      0.84      0.86       189
talk.politics.mideast        0.91      0.93      0.92       192

              accuracy                           0.89       779
             macro avg       0.89      0.89      0.89       779
          weighted avg       0.89      0.89      0.89       779

Accuracy: 0.8922
```