NLP Assignment -- Text Preprocessing (example using a simple Naive Bayes classifier)

SUPRATIM NAG (CSE/22/057)

In [1]:
```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.
1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (fro
m nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (fr
om nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-pac
kages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from
nltk) (4.66.6)
```

In [ ]:
```
import nltk
nltk.download('all')
```

In [3]:
```
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import string
```

In [ ]:
```
# Initialize the lemmatizer and download required NLTK data
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
```

In [5]:
```
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words("english"))
```

In [6]:
```
# Training data (using a toy dataset for illustration purposes)
training_data = [
    ("It was a great movie.", "pos"),
    ("I hated the book.", "neg"),
    ("The book was okay.", "pos"),
    ("I absolutely loved the performance!", "pos"),
    ("The plot was boring and predictable.", "neg"),
    ("What an amazing experience!", "pos"),
    ("The food was terrible.", "neg"),
    ("I enjoyed every minute of it.", "pos"),
    ("The service was really bad.", "neg"),
    ("It was an unforgettable adventure!", "pos"),
    ("I don't think I'll watch it again.", "neg"),
    ("The soundtrack was beautiful.", "pos"),
    ("I regret buying this product.", "neg"),
    ("The view was breathtaking!", "pos"),
    ("The story lacked depth.", "neg"),
    ("I am so happy with my purchase.", "pos"),
    ("The customer service was disappointing.", "neg"),
```

```
    ("It was worth every penny.", "pos"),
    ("The quality is not up to the mark.", "neg"),
    ("I'm thrilled with the results!", "pos")
]
```

In [7]:
```
# Function to preprocess and extract features from text
def extract_features(text):
    # Convert text to lowercase
    text = text.lower()
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove punctuation and stopwords, and apply lemmatization
    features = {
        lemmatizer.lemmatize(word): True
        for word in tokens if word not in stop_words and word not in string.punctua
    }
    return features
```

In [8]:
```
# Create a list of feature sets and labels
feature_sets = [(extract_features(text), label) for (text, label) in training_data]
```

In [9]:
```
# Train the classifier
classifier = NaiveBayesClassifier.train(feature_sets)
```

In [10]:
```
# Test the classifier on a new example
test_text = "The movie was great."
print("Sentiment:", classifier.classify(extract_features(test_text)))
```

```
Sentiment: pos
```