# PE02: Basic NLP Analysis

## Setup

First, install and import the required libraries. Make sure to download necessary NLTK data and spaCy models.

# Required libraries  
import nltk  
import spacy   
import numpy as np  
import pandas as pd  
from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer  
from sklearn.naive\_bayes import MultinomialNB  
from sklearn.linear\_model import LogisticRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report  
  
# Hint: You'll need to download specific NLTK packages  
# nltk.download(...) # What packages do you need for tokenization and stopwords?  
  
# You'll also need to download a spaCy model  
# !python -m spacy download ...

## Tasks

### Exercise 1: Enhanced Text Preprocessing

**Initial Setup:** 1. Create a small dataset of movie reviews (4-5 samples) with sentiment labels 2. Labels should be binary: 1 for positive, 0 for negative 3. Each review should be 1-2 sentences long

**Tasks:**

1. Compare NLTK vs spaCy preprocessing:

* from nltk.tokenize import word\_tokenize  
  from nltk.corpus import stopwords  
    
  # Implement two functions:  
  def nltk\_preprocess(text):  
   # Hint: Use word\_tokenize() and stopwords  
   pass  
    
  def spacy\_preprocess(text):  
   # Hint: Use nlp() and token.is\_stop  
   pass

1. Analyze preprocessing steps:
   * Show how the text changes after each step:
     + Basic tokenization
     + Case normalization
     + Stopword removal
   * Print statistics about token counts at each stage
2. Document and analyze differences between NLTK and spaCy results:
   * Compare outputs
   * Analyze which tokens are kept/removed by each
   * Consider which might be better for different applications

### Exercise 2: Basic Feature Extraction

1. Compare CountVectorizer and TfidfVectorizer:

* # Initialize both vectorizers  
  count\_vec = CountVectorizer()  
  tfidf\_vec = TfidfVectorizer()  
    
  # Hint: Use fit\_transform() and get\_feature\_names\_out()  
  # Compare the results - what's different about the outputs?

1. Experiment with n-gram ranges:
   * Try these combinations: (1,1), (1,2), (2,2)
   * Document how feature space size changes
   * Analyze the types of features captured
2. Analyze TF-IDF benefits and limitations:
   * Create a function to find top terms by TF-IDF score
   * Compare how different terms are weighted
   * Consider why certain terms get higher scores

### Exercise 3: Text Classification

1. Implement and compare classifiers:

* # Prepare your data  
  X = tfidf\_vec.fit\_transform(reviews)  
  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
   X, labels, test\_size=0.2, random\_state=42  
  )  
    
  # Implement both classifiers  
  models = {  
   'Naive Bayes': MultinomialNB(),  
   'Logistic Regression': LogisticRegression()  
  }  
    
  # Hint: Use classification\_report to compare results

1. Study preprocessing impact:
   * Create a function to evaluate model performance with different preprocessing
   * Compare no preprocessing vs NLTK vs spaCy
   * Document impact on accuracy
2. Feature selection analysis:
   * Use SelectKBest from sklearn.feature\_selection
   * Try different numbers of features (e.g., 5, 10, 15)
   * Analyze which features are selected and why

## Submission Requirements

1. Jupyter notebook with:
   * All code properly executed
   * Clear markdown explanations
   * Output showing results of each step
2. Brief report (2-3 pages) analyzing:
   * Preprocessing impact on text analysis
   * Feature extraction findings
   * Classification performance
   * Key challenges and solutions

## Evaluation Criteria

* Code implementation (30%)
* Results analysis (30%)
* Documentation (20%)
* Report quality (20%)