# PE03: Text Representation and Feature Extraction

## Overview

This exercise focuses on creating different text representations using the NLTK and scikit-learn libraries, working with the Reuters news article dataset.

### Prerequisites

pip install nltk scikit-learn pandas numpy

## Exercise and Questions

### Task 1: Basic Text Representation

Start by importing the necessary libraries and setting up the Reuters corpus:

import nltk  
from nltk.corpus import reuters  
from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer  
  
# Don't forget to download the Reuters corpus!  
nltk.download('reuters')  
  
# Read Reuters corpus  
documents = reuters.fileids()  
corpus = [reuters.raw(doc\_id) for doc\_id in documents]

Hint: After loading the corpus, print some basic information to understand its structure. How many documents are there? What does a sample document look like?

### Task 2: Create Bag-of-Words Representation

Implement the bag-of-words representation using CountVectorizer. Consider: - How to create and fit the vectorizer - How to examine the vocabulary size - How to inspect the shape of the resulting matrix - How to look at the first few terms in the vocabulary

Hint: The vocabulary\_ attribute of CountVectorizer contains a dictionary mapping terms to indices. How might you sort this to see terms in order?

### Task 3: Create TF-IDF Representation

Create a TF-IDF representation and compare it with the bag-of-words approach: - Implement TfidfVectorizer - Compare the shapes of both representations - Examine how the same document is represented in both formats

Hint: The nonzero() method on sparse matrices can help you identify which features are present in a document.

### Task 4: Stop Words Analysis

Modify your TF-IDF implementation to remove English stop words: - Create a new TfidfVectorizer with stop\_words=‘english’ - Compare vocabulary sizes before and after stop word removal - Analyze the impact on document features

Questions to consider: 1. How much does the vocabulary size change? 2. How does the number of features per document change? 3. What’s the impact on the representation quality?

### Task 5: New Document Processing

Create a function that can process new documents using your trained vectorizers:

def process\_new\_document(document, vectorizer):  
 """  
 Process a new document using trained vectorizer  
   
 Args:  
 document (str): Text to process  
 vectorizer: Trained CountVectorizer or TfidfVectorizer  
   
 Returns:  
 sparse matrix: Document vector  
 """  
 # Your implementation here  
 pass

Test your function with a sample news article about financial markets or economics.

Hints: - Remember that vectorizer.transform() expects a list of documents - You can use the vocabulary\_ attribute to see which terms were found - Consider comparing the number of features in both representations

### Required Libraries and Resources

* NLTK with Reuters corpus
* scikit-learn
* pandas
* numpy

### Analysis Questions

1. Compare the vocabulary sizes and matrix shapes you obtained. What do these numbers tell you about the Reuters corpus?
2. How many features (non-zero elements) did you find in your sample documents before and after stop word removal?
3. When processing new documents, what happens to terms that weren’t in the original training vocabulary?
4. Looking at the terms found in your test document, do you notice any patterns in what type of words are most common?

### Submission Requirements

1. Complete Jupyter notebook with all code cells executed
2. Comments explaining your implementation choices
3. Printed output showing:
   * Corpus statistics
   * Vocabulary information
   * Matrix shapes and characteristics
   * Feature comparisons
   * Test document analysis
4. Answers to all analysis questions with specific examples from your results

Remember: Document each step and include print statements to demonstrate your understanding of the transformations being applied.