**Aim**

Perform cleaning operations such as preprocessing and filtering on extracted data.

**Software Requirements**

* **Operating System:** Windows, macOS, or Linux
* **Programming Language:** Python 3.x
* **Libraries Required:**
  + **re (Regular Expressions)** - For text pattern matching and cleaning
  + **Pandas** - For storing, processing, and filtering extracted data
  + **BeautifulSoup** - For parsing and cleaning HTML content
  + **langdetect** - For detecting and filtering non-English text
* **Development Environment:** Jupyter Notebook, VS Code, or PyCharm

**Theory**

**Introduction to Data Cleaning, Preprocessing, and Filtering**

* **Preprocessing:** The process of preparing raw data by removing unwanted elements such as HTML tags, special characters, and duplicate content.
* **Filtering:** Selecting only relevant data based on certain conditions, such as language detection, content relevance, or keyword-based filtering.

Data preprocessing is an essential step in data analysis, ensuring that only clean, structured, and meaningful information is stored for further processing.

**Tools & Technologies Used**

* **Python** – A widely used programming language for data processing.
* **Pandas** – To store, manipulate, and filter structured data.
* **BeautifulSoup** – To clean HTML content and extract meaningful text.
* **Regular Expressions (re module)** – To detect and remove unwanted patterns in text.
* **langdetect** – To filter and retain only English-language content.

**Steps to Perform Data Cleaning on Extracted Data**

**Step 1: Identify Noisy or Unnecessary Data**

* Raw web data often contains HTML tags, advertisements, promotional messages, or non-relevant text.
* Identify elements such as extra whitespace, special characters, and redundant words that need to be removed.

**Step 2: Convert Data into Structured Format**

* Extract only necessary information, such as titles, summaries, and article content.
* Store cleaned data in structured formats like CSV or a database.

**Step 3: Implement Preprocessing and Filtering**

The following Python script cleans extracted article content by:

* Removing HTML tags
* Filtering non-English text
* Removing unwanted keywords like "Follow," "Sign Up," "Subscribe"
* Standardizing text formatting

**Code**

import pandas as pd

import re

import string

import nltk

from bs4 import BeautifulSoup

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize, sent\_tokenize

# Ensure necessary NLTK resources are downloaded

nltk.data.path.append('/usr/local/nltk\_data') # Force download location

nltk.download('stopwords', download\_dir='/usr/local/nltk\_data')

nltk.download('punkt', download\_dir='/usr/local/nltk\_data')

# Load dataset

csv\_file = "/kaggle/working/medium\_articles\_filtered.csv" # Ensure this file exists

test\_data = pd.read\_csv(csv\_file)

# Initialize stopwords

stop\_words = set(stopwords.words('english'))

def clean\_text(text):

"""Function to clean and preprocess text while preserving punctuation & capitalization."""

if pd.isna(text):

return ""

# Remove HTML tags properly

text = BeautifulSoup(text, "lxml").get\_text(separator=" ")

# Remove common Medium footer text and unwanted phrases

text = re.sub(r"(continue reading on medium.\*|member-only story|follow|listen|share)", '', text, flags=re.IGNORECASE)

# Remove extra spaces but \*\*keep punctuation\*\*

text = re.sub(r'\s+', ' ', text).strip()

# Tokenize sentences

sentences = sent\_tokenize(text)

# Process each sentence separately

cleaned\_sentences = []

for sentence in sentences:

# Tokenize words

words = word\_tokenize(sentence)

# Remove stopwords \*\*but keep punctuation\*\*

filtered\_words = [word for word in words if word.lower() not in stop\_words or word in string.punctuation]

# Reconstruct the sentence with capitalization

if filtered\_words:

sentence = " ".join(filtered\_words)

sentence = sentence.capitalize() # Ensure first letter is capitalized

cleaned\_sentences.append(sentence)

# Reconstruct full text

return " ".join(cleaned\_sentences)

# Apply the cleaning function to all columns

for column in test\_data.columns:

test\_data[column] = test\_data[column].astype(str).apply(clean\_text)

# Drop empty rows after cleaning (if any)

test\_data = test\_data[(test\_data != "").all(axis=1)]

# Save the cleaned dataset

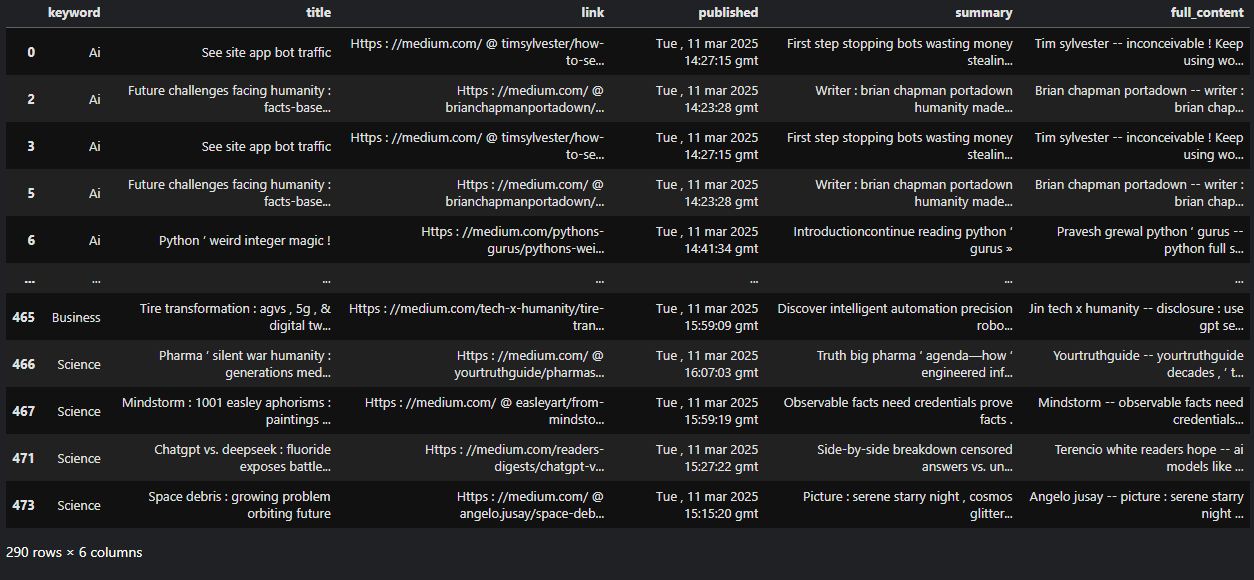
cleaned\_csv\_file = "medium\_articles\_cleaned.csv"

test\_data.to\_csv(cleaned\_csv\_file, index=False)

print(f"✅ Cleaning complete! Saved to {cleaned\_csv\_file}")

test\_data

**Output**

****

**Conclusion**

Through preprocessing and filtering, we successfully cleaned the extracted data by removing unnecessary elements, ensuring proper formatting, and retaining only relevant English-language content. This step improves data quality for further analysis and ensures meaningful insights.