Web Scraping Documentation

1. **Architecture:** The solution is built around three main components:
    - Data Loading: Excel file handling using pandas
    - Web Scraping: Using requests and BeautifulSoup4 for content extraction
    - File Management: Organized storage of extracted content

2. **Extraction Strategy**
    - Targets specific HTML elements (h1, h3, p, li)
    - Removes duplicate content using sets
    - Maintains content hierarchy (title + body)
    - Handles encoding for international character support

3. **Dependencies**
   Required Python packages:
    - Python
    - Pandas
    - Requests
    - beautifulsoup4

4. **Code Structure:**
   Key Functions:
    - extract_data(url)
    - Parameters: URL string
    - Returns: Tuple of (title, content)
    - Purpose: Scrapes webpage and extracts relevant text
    - save_to_file(url_id, title, content)
    - Parameters: URL ID, title text, content text
    - Purpose: Saves extracted content to text file
    - Output: Creates .txt file named with URL_ID

5. **Running in Jupyter Notebook**
    - Create new notebook
    - Copy the code
    - Update file paths
    - Run all cells

6. **Error Handling:**
   The script includes robust error handling for:
    - Failed HTTP requests
    - Missing webpage elements
    - Network connectivity issues
    - File writing errors
    - Character encoding issues

7. **Special Considerations**
    - Duplicate Content: Uses set data structure to prevent duplicate text. Maintains original content order
    - Character Encoding: Files are saved in UTF-8 format. Handles international characters
    - Content Extraction: Targets main content areas. Excludes navigation, footers, and sidebars
8. **Best Practices**
   Rate Limiting: Consider adding delays between requests
   Respect website robots.txt

9. **Troubleshooting**
   Common issues and solutions:
    - Connection Errors
      Check internet connectivity
      Verify URL accessibility
      Consider using proxies
    - File Permission Issues
      Verify write permissions
      Check folder existence
      Ensure proper file path
    - Content Extraction Problems
      Website might have different structure

Data Analysis Documentation

Solution Approach

1. Architecture

- **TextAnalyzer Class**: Core class that handles all text processing operations
- **analyze_files Function**: Wrapper function to process multiple files and generate output

2. Key Optimizations

- Set data structures for O(1) lookup of stop words and sentiment words
- Single-pass processing for multiple metrics
- Generator expressions for memory efficiency
- Efficient file handling with proper encoding

3. Text Analysis Process

1. **Text Preprocessing**
    - Remove digits and special characters
    - Convert to lowercase

- Tokenize into sentences and words

2. **Metrics Calculation**
    - Sentiment Scores (Positive/Negative)
    - Readability Metrics (FOG Index, Complex Words)
    - Statistical Measures (Word Count, Average Lengths)
    - Linguistic Features (Personal Pronouns)
    - And many more

3. Dependencies

Required Python packages:

python

pandas

nltk

4. Running the Code

Using Jupyter Notebook/Colab:

1. Open Jupyter Notebook
2. Create new notebook or open existing one
3. Copy the code into a cell
4. Update file paths:

Expected Output

Generates 'Output Data Structure.xlsx' in the working directory
Excel file contains columns:

- URL_ID
- URL
- POSITIVE SCORE
- NEGATIVE SCORE
- POLARITY SCORE
- And other metrics...

5. Troubleshooting

Common issues and solutions:

- FileNotFoundError: Verify file paths are correct, Check if all required files exist, Ensure proper file permissions
- UnicodeDecodeError: Files are handled with UTF-8 encoding and 'ignore' error mode. If issues persist, check file encodings
- ImportError: Verify all dependencies are installed. Check Python version compatibility. Ensure virtual environment is activated (if using)

- Memory Issues: Code uses efficient processing methods. For very large datasets, consider processing in batches