# NLP Text Analysis Project - Complete Instructions Documentation

## Project Overview

This project performs comprehensive Natural Language Processing (NLP) text analysis on web articles. It extracts textual content from URLs and computes various linguistic and sentiment metrics including sentiment scores, readability metrics, and complex linguistic features.

## Table of Contents

1. [Solution Approach](#solution-approach)
2. [System Requirements](#system-requirements)
3. [Installation Guide](#installation-guide)
4. [File Structure](#file-structure)
5. [How to Run the Program](#how-to-run-the-program)
6. [Output Description](#output-description)
7. [Troubleshooting](#troubleshooting)
8. [Technical Details](#technical-details)

## Solution Approach

### 1. Problem Analysis

The project addresses the challenge of automated text analysis for web articles by:

* **Web Scraping**: Extracting clean article content from URLs
* **Sentiment Analysis**: Determining positive/negative sentiment using word dictionaries
* **Readability Assessment**: Computing FOG Index, complexity measures, and linguistic features
* **Data Processing**: Handling stopwords, text cleaning, and normalization

### 2. Architecture Design

The solution follows a modular object-oriented approach with the ComprehensiveNLPAnalyzer class containing:

* **Data Loading Module**: Loads positive/negative word dictionaries and stopword lists
* **Web Scraping Engine**: Uses BeautifulSoup for robust HTML content extraction
* **Text Processing Pipeline**: Handles tokenization, cleaning, and stopword removal
* **Sentiment Analysis Engine**: Calculates sentiment scores using dictionary-based approach
* **Readability Calculator**: Computes FOG Index, syllable counts, and complexity metrics
* **Results Generation**: Outputs structured Excel files with all computed metrics

### 3. Key Algorithms

* **Syllable Counting**: Vowel-based algorithm with special handling for silent ‘e’
* **Complex Word Detection**: Words with >2 syllables classified as complex
* **FOG Index Calculation**: 0.4 × (Average Sentence Length + Percentage Complex Words)
* **Sentiment Scoring**: Dictionary-based positive/negative word matching
* **Polarity Score**: (Positive - Negative) / (Positive + Negative + ε)

## System Requirements

### Operating System

* Windows 10/11
* macOS 10.14+
* Linux (Ubuntu 18.04+)

### Python Version

* Python 3.9 or higher (recommended: Python 3.11)

### Hardware Requirements

* RAM: Minimum 4GB (8GB recommended for large datasets)
* Storage: 500MB free space
* Internet connection (required for web scraping)

## Installation Guide

### Step 1: Install Python

Download and install Python from [python.org](https://python.org) - ✅ Check “Add Python to PATH” during installation

### Step 2: Install Required Dependencies

Open Command Prompt/Terminal and run:

pip install pandas numpy requests beautifulsoup4 openpyxl

**Individual package installation:**

pip install pandas # Data manipulation and analysis  
pip install numpy # Numerical computing  
pip install requests # HTTP library for web scraping  
pip install beautifulsoup4 # HTML parsing  
pip install openpyxl # Excel file handling

### Step 3: Verify Installation

Test your installation:

python -c "import pandas, numpy, requests, bs4, openpyxl; print('All dependencies installed successfully!')"

## File Structure

Your project directory should contain:

NLP\_Project/  
│  
├── complete\_nlp\_analyzer.py # Main analysis script  
├── Input.xlsx # Input URLs file  
│  
├── MasterDictionary/  
│ ├── positive-words.txt # Positive sentiment words  
│ └── negative-words.txt # Negative sentiment words  
│  
├── StopWords/  
│ ├── StopWords\_Auditor.txt  
│ ├── StopWords\_DatesandNumbers.txt  
│ ├── StopWords\_Generic.txt  
│ ├── StopWords\_GenericLong.txt  
│ ├── StopWords\_Geographic.txt  
│ └── StopWords\_Names.txt  
│  
└── Results/ # Output directory (auto-created)  
 ├── Output.xlsx # Analysis results  
 └── extracted\_articles/ # Individual article text files

**⚠️ Important Path Configuration:**

The script expects files at specific paths. Update these paths in complete\_nlp\_analyzer.py to match your setup:

Update these paths in the load\_word\_lists() method:  
"C:/Users/HP/Downloads/NLP Project/MasterDictionary/positive-words.txt"  
"C:/Users/HP/Downloads/NLP Project/MasterDictionary/negative-words.txt"  
"C:/Users/HP/Downloads/NLP Project/StopWords/StopWords\_\*.txt"  
  
Update input file path in process\_all\_articles() method:  
input\_file = "C:/Users/HP/Downloads/NLP Project/Input.xlsx"

## How to Run the Program

### Method 1: Direct Execution (Recommended)

1. **Open Command Prompt/Terminal**
2. **Navigate to your project directory:**

* cd "C:/path/to/your/NLP\_Project"

1. **Run the script:**

* python complete\_nlp\_analyzer.py

### Method 2: Using Python IDE (VS Code, PyCharm)

1. **Open complete\_nlp\_analyzer.py in your IDE**
2. **Update file paths** to match your system
3. **Run the script**

### Method 3: Custom Input Files

To use different input files, modify the script:

if \_\_name\_\_ == "\_\_main\_\_":  
 analyzer = ComprehensiveNLPAnalyzer()  
 # Specify custom input and output files  
 results = analyzer.process\_all\_articles(  
 input\_file="path/to/your/Input.xlsx",  
 output\_file="CustomOutput.xlsx"  
 )

### Expected Runtime

* **147 articles**: ~15-25 minutes (depending on internet speed and website response times)

## Output Description

### Primary Output File: Output.xlsx

The results Excel file contains these columns for each analyzed URL:

| Column Name | Description | Example Value |
| --- | --- | --- |
| URL\_ID | Unique identifier for each article | Netclan20241017 |
| URL | Original article URL | https://insights.blackcoffer.com/… |
| POSITIVE SCORE | Count of positive sentiment words | 15 |
| NEGATIVE SCORE | Count of negative sentiment words | 8 |
| POLARITY SCORE | Sentiment polarity: (Pos-Neg)/(Pos+Neg) | 0.304 |
| SUBJECTIVITY SCORE | Opinion density: (Pos+Neg)/TotalWords | 0.062 |
| AVG SENTENCE LENGTH | Average words per sentence | 18.5 |
| PERCENTAGE OF COMPLEX WORDS | % of words with >2 syllables | 25.3 |
| FOG INDEX | Readability measure (higher = harder) | 17.8 |
| AVG NUMBER OF WORDS PER SENTENCE | Same as avg sentence length | 18.5 |
| COMPLEX WORD COUNT | Number of complex words | 89 |
| WORD COUNT | Total words in article | 456 |
| SYLLABLE PER WORD | Average syllables per word | 1.87 |
| PERSONAL PRONOUNS | Count of I, we, my, ours, us | 5 |
| AVG WORD LENGTH | Average characters per word | 5.2 |

### Secondary Outputs

1. **extracted\_articles/ directory**: Contains individual text files for each scraped article
2. **Console output**: Real-time progress updates and summary statistics
3. **Error logs**: Details of any failed URL extractions

## Troubleshooting

### Common Issues and Solutions

#### **Issue 1: ModuleNotFoundError**

ModuleNotFoundError: No module named 'pandas'

**Solution:**

pip install pandas numpy requests beautifulsoup4 openpyxl

#### **Issue 2: FileNotFoundError**

FileNotFoundError: [Errno 2] No such file or directory: 'positive-words.txt'

**Solution:** Update file paths in the script to match your directory structure.

#### **Issue 3: Connection Errors**

Connection error: Failed to establish connection

**Solutions:** - Check internet connection - Disable firewall/antivirus temporarily - Run as Administrator - Try using mobile hotspot

#### **Issue 4: Permission Denied**

PermissionError: [Errno 13] Permission denied

**Solution:** Run Command Prompt as Administrator

#### **Issue 5: SSL Certificate Errors**

**Solution:** Update certificates or add this to your script:

import ssl  
ssl.\_create\_default\_https\_context = ssl.\_create\_unverified\_context

### Performance Optimization

1. **For faster processing:** Reduce sleep time between requests
2. **For better success rate:** Increase timeout values
3. **For large datasets:** Process in batches
4. **For memory efficiency:** Clear variables after processing

## Technical Details

### Dependencies Explained

* **pandas (1.3.0+)**: Data manipulation and Excel file handling
* **numpy (1.21.0+)**: Numerical computations and array operations
* **requests (2.25.0+)**: HTTP library for web scraping
* **beautifulsoup4 (4.9.0+)**: HTML parsing and content extraction
* **openpyxl (3.0.0+)**: Excel file reading and writing

### Text Processing Pipeline

1. **URL Request**: Send HTTP GET request with browser headers
2. **HTML Parsing**: Extract content using BeautifulSoup selectors
3. **Text Cleaning**: Remove scripts, styles, and normalize whitespace
4. **Tokenization**: Split text into words and sentences
5. **Stopword Removal**: Filter out common words and noise
6. **Feature Extraction**: Calculate all linguistic metrics
7. **Result Compilation**: Structure data for Excel output

### Sentiment Analysis Methodology

The system uses a **dictionary-based approach**: - **Positive Words**: ~2000 words indicating positive sentiment - **Negative Words**: ~4800 words indicating negative sentiment  
- **Scoring**: Simple count of matching words after stopword removal - **Normalization**: Scores normalized by total word count

### Readability Metrics

* **FOG Index**: Measures text complexity (12+ = college level)
* **Complex Words**: Words with more than 2 syllables
* **Average Sentence Length**: Words per sentence indicator
* **Syllable Counting**: Vowel-based algorithm with edge case handling

## Usage Examples

### Basic Usage

python complete\_nlp\_analyzer.py

### Processing Specific Number of Articles

Modify the main section:

if \_\_name\_\_ == "\_\_main\_\_":  
 analyzer = ComprehensiveNLPAnalyzer()  
 input\_df = pd.read\_excel("Input.xlsx").head(10) # Process first 10 only  
 # ... rest of processing

### Custom Output Directory

my\_directory = "C:/Users/YourName/Desktop/Results"  
os.makedirs(my\_directory, exist\_ok=True)  
results\_df.to\_excel(f"{my\_directory}/Custom\_Output.xlsx", index=False)

## Support and Maintenance

### For Technical Issues:

1. Check Python and package versions
2. Verify file paths and permissions
3. Test internet connectivity
4. Review error messages carefully

### For Data Issues:

1. Validate input Excel file format
2. Check URL accessibility
3. Verify dictionary files are properly formatted
4. Monitor web scraping success rates

### Performance Monitoring:

* Track processing time per article
* Monitor memory usage for large datasets
* Log failed extractions for analysis
* Validate output data quality

**Project Status**: Production Ready ✅  
**Last Updated**: September 2025  
**Python Compatibility**: 3.7+  
**Platform Support**: Windows, macOS, Linux