BlackcofferIntern

October 4, 2025

[30]:
```
# Imports & Configuration

import pandas as pd
import numpy as np
import requests
from bs4 import BeautifulSoup as bs
import re
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
import os

# Download necessary NLTK resources
for resource in ["punkt", "stopwords"]:
    try:
        nltk.data.find(f"tokenizers/{resource}" if resource == "punkt" else␣
 ↪f"corpora/{resource}")
    except LookupError:
        nltk.download(resource)

# Confirm working directory
print("Current working directory:", os.getcwd())
```

```
Current working directory: C:\Users\sunik
```

[29]:
```
# Load Input file directly from your Jupyter home directory
data = pd.read_excel("Input.xlsx")

print(f"Loaded {len(data)} URLs from Input.xlsx")
data.head()
```

```
Loaded 147 URLs from Input.xlsx
```

[29]:

	URL_ID	URL
0	bctech2011	https://insights.blackcoffer.com/ml-and-ai-bas…
1	bctech2012	https://insights.blackcoffer.com/streamlined-i…
2	bctech2013	https://insights.blackcoffer.com/efficient-dat…
3	bctech2014	https://insights.blackcoffer.com/effective-man…
4	bctech2015	https://insights.blackcoffer.com/streamlined-t…

[10]:
```
# Create empty lists for scraped results
titles = []
texts = []
failed_urls = []

for index, row in data.iterrows():
    url_id = row["URL_ID"]
    url = row["URL"]
    print(f"Processing {index+1}/{len(data)}: {url_id}")

    try:
        # Send request
        response = requests.get(url, timeout=30)
        response.raise_for_status()
        bsoup = bs(response.text, "html.parser")

        # Extract title
        title_tag = bsoup.find("h1", class_="entry-title") or bsoup.
 ↪find("title")
        article_title = title_tag.get_text(strip=True) if title_tag else "TITLE␣
 ↪NOT FOUND"

        # Extract article text
        content_tag = bsoup.find("div", class_="td-post-content tagdiv-type")␣
 ↪or bsoup.find("div", class_="tdb-block-inner td-fix-index")
        article_text = content_tag.get_text(separator=" ", strip=True) if␣
 ↪content_tag else "ARTICLE TEXT NOT FOUND"

        titles.append(article_title)
        texts.append(article_text)

    except requests.exceptions.RequestException as e:
        print(f" Failed to fetch URL_ID {url_id}: {e}")
        titles.append("ERROR")
        texts.append("ERROR")
        failed_urls.append(url_id)

data["Article_Title"] = titles
data["Article_Text"] = texts
print(f"Scraping complete - {len(failed_urls)} URLs failed.")
```

```
Processing 1/147: bctech2011
Processing 2/147: bctech2012
Processing 3/147: bctech2013
Processing 4/147: bctech2014
Processing 5/147: bctech2015
Processing 6/147: bctech2016
```

```
Processing 7/147: bctech2017
Processing 8/147: bctech2018
Processing 9/147: bctech2019
Processing 10/147: bctech2020
Processing 11/147: bctech2021
Processing 12/147: bctech2022
Processing 13/147: bctech2023
Processing 14/147: bctech2024
Processing 15/147: bctech2025
Processing 16/147: bctech2026
Processing 17/147: bctech2027
Processing 18/147: bctech2028
Processing 19/147: bctech2029
Processing 20/147: bctech2030
Processing 21/147: bctech2031
Processing 22/147: bctech2032
Processing 23/147: bctech2033
Processing 24/147: bctech2034
Processing 25/147: bctech2035
Processing 26/147: bctech2036
Processing 27/147: bctech2037
Processing 28/147: bctech2038
Processing 29/147: bctech2039
Processing 30/147: bctech2040
Processing 31/147: bctech2041
Processing 32/147: bctech2042
Processing 33/147: bctech2043
Processing 34/147: bctech2044
Processing 35/147: bctech2045
Processing 36/147: bctech2046
Processing 37/147: bctech2047
Processing 38/147: bctech2048
Processing 39/147: bctech2049
Processing 40/147: bctech2050
Processing 41/147: bctech2051
Processing 42/147: bctech2052
Processing 43/147: bctech2053
Processing 44/147: bctech2054
Processing 45/147: bctech2055
Processing 46/147: bctech2056
Processing 47/147: bctech2057
Processing 48/147: bctech2058
Processing 49/147: bctech2059
Processing 50/147: bctech2060
Processing 51/147: bctech2061
Processing 52/147: bctech2062
Processing 53/147: bctech2063
Processing 54/147: bctech2064
```

```
Processing 55/147: bctech2065
Processing 56/147: bctech2066
Processing 57/147: bctech2067
Processing 58/147: bctech2068
Processing 59/147: bctech2069
Processing 60/147: bctech2070
Processing 61/147: bctech2071
Processing 62/147: bctech2072
Processing 63/147: bctech2073
Processing 64/147: bctech2074
Processing 65/147: bctech2075
Processing 66/147: bctech2076
Processing 67/147: bctech2077
Processing 68/147: bctech2078
Processing 69/147: bctech2079
Processing 70/147: bctech2080
Processing 71/147: bctech2081
Processing 72/147: bctech2082
Processing 73/147: bctech2083
Processing 74/147: bctech2084
Processing 75/147: bctech2085
Processing 76/147: bctech2086
Processing 77/147: bctech2087
Processing 78/147: bctech2088
Processing 79/147: bctech2089
Processing 80/147: bctech2090
Processing 81/147: bctech2091
Processing 82/147: bctech2092
Processing 83/147: bctech2093
Processing 84/147: bctech2094
Processing 85/147: bctech2095
Processing 86/147: bctech2096
Processing 87/147: bctech2097
Processing 88/147: bctech2098
Processing 89/147: bctech2099
Processing 90/147: bctech2100
Processing 91/147: bctech2101
Processing 92/147: bctech2102
Processing 93/147: bctech2103
Processing 94/147: bctech2104
Processing 95/147: bctech2105
Processing 96/147: bctech2106
Processing 97/147: bctech2107
Processing 98/147: bctech2108
Processing 99/147: bctech2109
Processing 100/147: bctech2110
Processing 101/147: bctech2111
Processing 102/147: bctech2112
```

```
Processing 103/147: bctech2113
Processing 104/147: bctech2114
Processing 105/147: bctech2115
Processing 106/147: bctech2116
Processing 107/147: bctech2117
Processing 108/147: bctech2118
Processing 109/147: bctech2119
Processing 110/147: bctech2120
Processing 111/147: bctech2121
Processing 112/147: bctech2122
Processing 113/147: bctech2123
Processing 114/147: bctech2124
Processing 115/147: bctech2125
Processing 116/147: bctech2126
Processing 117/147: bctech2127
Processing 118/147: bctech2128
Processing 119/147: bctech2129
Processing 120/147: bctech2130
Processing 121/147: bctech2131
Processing 122/147: bctech2132
Processing 123/147: bctech2133
Processing 124/147: bctech2134
Processing 125/147: bctech2135
Processing 126/147: bctech2136
Processing 127/147: bctech2137
Processing 128/147: bctech2138
Processing 129/147: bctech2139
Processing 130/147: bctech2140
Processing 131/147: bctech2141
Processing 132/147: bctech2142
Processing 133/147: bctech2143
Processing 134/147: bctech2144
Processing 135/147: bctech2145
Processing 136/147: bctech2146
Processing 137/147: bctech2147
Processing 138/147: bctech2148
Processing 139/147: bctech2149
Processing 140/147: bctech2150
Processing 141/147: bctech2151
Processing 142/147: bctech2152
Processing 143/147: bctech2153
Processing 144/147: bctech2154
Processing 145/147: bctech2155
Processing 146/147: bctech2156
Processing 147/147: bctech2157
Scraping complete - 0 URLs failed.
```

[12]:
```
# Load stopword files
def load_wordlist(path):
    words = set()
    with open(path, "r", encoding="utf-8", errors="ignore") as f:
        for line in f:
            w = line.strip()
            if w and not w.startswith(";"):
                words.add(w.lower())
    return words

stop_words = set(stopwords.words("english"))

# Adding custom stopword lists (merge all uploaded ones)
custom_stop_files = [
    "StopWords_Auditor.txt", "StopWords_Currencies.txt",␣
 ↪"StopWords_DatesandNumbers.txt",
    "StopWords_Generic.txt", "StopWords_GenericLong.txt", "StopWords_Geographic.
 ↪txt",
    "StopWords_Names.txt"
]
for file in custom_stop_files:
    stop_words.update(load_wordlist(file))

# Load positive and negative words
positive_words = load_wordlist("positive-words.txt")
negative_words = load_wordlist("negative-words.txt")

print(f"Loaded {len(stop_words)} stopwords, {len(positive_words)} positive,␣
 ↪{len(negative_words)} negative words.")
```

```
Loaded 12797 stopwords, 2006 positive, 4783 negative words.
```

[13]:
```
test_url = 'https://insights.blackcoffer.com/
 ↪efficient-aws-infrastructure-setup-and-management-addressing-security-scalability-and-compl
 ↪'
print(url)
```

```
https://insights.blackcoffer.com/amazon-buy-bot-an-automation-ai-tool-to-auto-
checkouts/
```

[19]:
```
# Full scraping loop for all articles
import requests
from bs4 import BeautifulSoup as bs
from tqdm import tqdm

extracted_titles = []
extracted_texts = []
failed_urls = []
```

```
for index, row in tqdm(data.iterrows(), total=len(data)):
    url_id = row['URL_ID']
    url = row['URL']

    try:
        response = requests.get(url, timeout=60)
        response.raise_for_status()
        bsoup = bs(response.text, 'html.parser')

        title_tag = bsoup.find('h1', class_='entry-title') or bsoup.
 ↪find('title')
        article_title = title_tag.get_text().strip() if title_tag else "TITLE␣
 ↪NOT FOUND"

        content_tag = bsoup.find('div', class_='td-post-content tagdiv-type')␣
 ↪or \
                      bsoup.find('div', class_='tdb-block-inner td-fix-index')
        article_text = content_tag.get_text(separator="\n").strip() if␣
 ↪content_tag else "ARTICLE TEXT NOT FOUND"

        extracted_titles.append(article_title)
        extracted_texts.append(article_text)

    except requests.exceptions.RequestException as e:
        print(f" Failed URL_ID {url_id} ({url}): {e}")
        extracted_titles.append("ERROR")
        extracted_texts.append("ERROR")
        failed_urls.append(url_id)

data["Article_Title"] = extracted_titles
data["Article_Text"] = extracted_texts

print(f" Scraping complete. {len(failed_urls)} URLs failed.")
```

```
100%|
   | 147/147 [04:33<00:00,  1.86s/it]

 Scraping complete. 0 URLs failed.
```

[20]:
```
print(data.head())
```

```
       URL_ID                                                URL  \
0  bctech2011  https://insights.blackcoffer.com/ml-and-ai-bas…
1  bctech2012  https://insights.blackcoffer.com/streamlined-i…
2  bctech2013  https://insights.blackcoffer.com/efficient-dat…
3  bctech2014  https://insights.blackcoffer.com/effective-man…
```

```
4  bctech2015  https://insights.blackcoffer.com/streamlined-t…

                                        Article_Title  \
0  ML and AI-based insurance premium model to pre…
1  Streamlined Integration: Interactive Brokers A…
2  Efficient Data Integration and User-Friendly I…
3  Effective Management of Social Media Data Extr…
4  Streamlined Trading Operations Interface for M…

                                        Article_Text
0  Client Background\n\n\nClient:\n A leading ins…
1  Client Background\n\n\nClient:\n A leading fin…
2  Client Background\n\n\nClient:\n A leading tec…
3  Client Background\n\n\nClient:\n A leading tec…
4  Client Background\n\n\nClient:\n A leading fin…
```

[25]:
```python
import re
from nltk.tokenize import sent_tokenize, word_tokenize

def count_syllables(word):
    """Estimate syllables in a word."""
    word = word.lower()
    vowels = "aeiou"
    count, prev_vowel = 0, False
    for ch in word:
        if ch in vowels:
            if not prev_vowel:
                count += 1
            prev_vowel = True
        else:
            prev_vowel = False
    if word.endswith(("es", "ed")):
        count -= 1
    return max(1, count)
```

[26]:
```python
def analyze_text(text):
    """Compute sentiment and readability metrics for one article."""
    if not isinstance(text, str) or text.strip() in ["ERROR", "ARTICLE TEXT NOT␣
 ↪FOUND"]:
        return {col: 0 for col in [
            "POSITIVE SCORE", "NEGATIVE SCORE", "POLARITY SCORE", "SUBJECTIVITY␣
 ↪SCORE",
            "AVG SENTENCE LENGTH", "PERCENTAGE OF COMPLEX WORDS", "FOG INDEX",
            "AVG NUMBER OF WORDS PER SENTENCE", "COMPLEX WORD COUNT",
            "WORD COUNT", "SYLLABLE PER WORD", "PERSONAL PRONOUNS", "AVG WORD␣
 ↪LENGTH"
        ]}
```

```
    sentences = sent_tokenize(text)
    words = [w for w in word_tokenize(text) if w.isalpha()]
    words_clean = [w.upper() for w in words if w.lower() not in stop_words]

    # Sentiment
    pos = sum(1 for w in words_clean if w in positive_words)
    neg = sum(1 for w in words_clean if w in negative_words)
    polarity = (pos - neg) / ((pos + neg) + 1e-6)
    subjectivity = (pos + neg) / (len(words_clean) + 1e-6)

    # Readability
    avg_sentence_len = len(words_clean) / max(1, len(sentences))
    complex_words = [w for w in words_clean if count_syllables(w) > 2]
    percent_complex = len(complex_words) / max(1, len(words_clean))
    fog_index = 0.4 * (avg_sentence_len + percent_complex)

    # Other metrics
    word_count = len(words_clean)
    syllables = sum(count_syllables(w) for w in words_clean)
    syllable_per_word = syllables / max(1, len(words_clean))
    pronouns = len(re.findall(r"\b(I|we|my|ours|us)\b", text, flags=re.I))
    avg_word_len = sum(len(w) for w in words_clean) / max(1, len(words_clean))

    return {
        "POSITIVE SCORE": pos,
        "NEGATIVE SCORE": neg,
        "POLARITY SCORE": polarity,
        "SUBJECTIVITY SCORE": subjectivity,
        "AVG SENTENCE LENGTH": avg_sentence_len,
        "PERCENTAGE OF COMPLEX WORDS": percent_complex,
        "FOG INDEX": fog_index,
        "AVG NUMBER OF WORDS PER SENTENCE": avg_sentence_len,
        "COMPLEX WORD COUNT": len(complex_words),
        "WORD COUNT": word_count,
        "SYLLABLE PER WORD": syllable_per_word,
        "PERSONAL PRONOUNS": pronouns,
        "AVG WORD LENGTH": avg_word_len
    }
```

[22]:
```
print(" Running NLP analysis on all scraped articles...")
analysis_results = data["Article_Text"].apply(analyze_text).apply(pd.Series)

# Merge with main DataFrame
data = pd.concat([data, analysis_results], axis=1)

print(" NLP analysis complete.")
```

```
Running NLP analysis on all scraped articles…
NLP analysis complete.
```

[23]:
```
output_path = "Output_Data_Structure_Completed.xlsx"
data.to_excel(output_path, index=False)
print(f" Final file saved successfully as: {output_path}")
```

```
Final file saved successfully as: Output_Data_Structure_Completed.xlsx
```

[24]:
```
data.head(3)
```

[24]:

	URL_ID	URL	Article_Title	Article_Text	POSITIVE SCORE	NEGATIVE SCORE	POLARITY SCORE	SUBJECTIVITY SCORE	AVG SENTENCE LENGTH	PERCENTAGE OF COMPLEX WORDS	FOG INDEX	AVG NUMBER OF WORDS PER SENTENCE	COMPLEX WORD COUNT	WORD COUNT	SYLLABLE PER WORD	PERSONAL PRONOUNS	AVG WORD LENGTH
0	bctech2011	https://insights.blackcoffer.com/ml-and-ai-bas…	ML and AI-based insurance premium model to pre…	Client Background\n\n\nClient:\n A leading ins…	0.0	0.0	0.0	0.0	9.463277	0.507463	3.988296	9.463277	850.0	1675.0	2.660299	2.0	7.948060
1	bctech2012	https://insights.blackcoffer.com/streamlined-i…	Streamlined Integration: Interactive Brokers A…	Client Background\n\n\nClient:\n A leading fin…	0.0	0.0	0.0	0.0	6.269231	0.438650	2.683152	6.269231	143.0	326.0	2.631902	1.0	7.846626
2	bctech2013	https://insights.blackcoffer.com/efficient-dat…	Efficient Data Integration and User-Friendly I…	Client Background\n\n\nClient:\n A leading tec…	0.0	0.0	0.0	0.0	11.542857	0.349010	4.756747	11.542857	141.0	404.0	2.306931	1.0	7.264851

[33]:
```
OUTPUT_SAVE = DATA_DIR / "Output_Data_Structure_Completed.xlsx"
OUTPUT_CSV = DATA_DIR / "Output_Data_Structure_Completed.csv"
```

```
print(OUTPUT_CSV)
```

```
C:\Users\sunik\Output_Data_Structure_Completed.csv
```

[32]:

[ ]: