Crawler Design for Sanskrit data

Introduction

Web crawlers are used for tasks like web indexing, data mining,contents scraping, and monitoring website changes. They embark on their exploration from seed URLs and meticulously follow hyperlinks in an automated and methodical manner. Challenges in web crawling are numerous, spanning issues such as handling duplicate content, navigating dynamic web pages, managing authentication mechanisms, and dealing with diverse types of media, including text, images, and videos. Web crawlers stand as the stalwart automation agents of internet data gathering, rendering digital content accessible and searchable for a plethora of purposes.

API’s used are:

1. **WikipediaAPI**: The WikipediaAPI library is used to interact with Wikipedia and access Wikipedia articles. It allows you to retrieve information from Wikipedia, including page content and links. In this script, it is used to create a Wikipedia API object, retrieve the seed page, and access links from that page. Explanation of wikipediaapi is used in this script:
2. **Creating a Wikipedia API object**: wikipediaapi.Wikipedia class is used to create a Wikipedia API object named wiki\_wiki. This object serves as the interface for interacting with Wikipedia. When creating the object, you specify the language code for the Wikipedia edition you want to access.
3. **Retrieving the Seed Page**: The script defines a seed\_page\_title variable, which contains the title of the seed page you want to access. In this script, it's set to "Om," which is the starting point for the analysis.
4. **Accessing Links**: Once the seed\_page object is obtained, you can access various attributes and methods associated with it. . These links represent references to other Wikipedia articles related to the seed page. The links variable now contains a list of titles of articles that are linked from the seed page.
5. **requests**: The requests library is a popular Python library for making HTTP requests to web servers. In this script, it is used to send HTTP requests to fetch the content of Wikipedia pages and other web resources. Explanation of requests used in the script:
6. **Importing the requests Library**: At the beginning of the script, the requests library is imported to make it available for use in the code.
7. **HTTP GET Requests**: The primary function of the requests library is to send HTTP GET requests to web servers. In the script, it is used to send GET requests to Wikipedia's servers to retrieve the content of Wikipedia pages and requests.get() is used to fetch the content of those pages.
8. **Handling Responses**: After sending an HTTP GET request, the requests library handles the server's response. It provides access to the response content, status code, headers, and more. In the script, the response content (HTML content of Wikipedia pages) is accessed and processed.
9. **Error Handling**: The requests library also includes mechanisms for error handling. It can handle various HTTP status codes and exceptions, allowing you to respond appropriately when requests fail or encounter issues.
10. **BeautifulSoup**: BeautifulSoup is a Python library used for web scraping. It parses HTML and XML documents, making it easier to extract specific information from web pages. Explanation of BeautifulSoup is used in this script:
11. **Importing the BeautifulSoup Library**: At the beginning of the script, the BeautifulSoup class from the bs4 module is imported to make it available for use in the code.
12. **Parsing HTML Content**: The BeautifulSoup constructor is used to create a BeautifulSoup object, passing the HTML content as its input.
13. **Extracting Textual Content**: With the BeautifulSoup object in hand, you can easily extract textual content from the HTML document. In the script, this is used to extract the textual content of Wikipedia pages.
14. **Additional Parsing and Navigation**: Functions like find(), find\_all(), and various CSS selector-based methods to locate and extract specific elements of interest.
15. **math**: The math library in Python provides mathematical functions and constants. In this script, it is likely imported for potential mathematical operations, although it is not actively used in the code.
16. **Importing the math Library**: At the start of the script, the math library is imported to make its mathematical functions and constants available for potential use.
17. **Potential Mathematical Operations**: The inclusion of the math library suggests that the script might have been designed with the possibility of performing mathematical calculations or operations at some point.
18. **Not Actively Used**: However, in the provided script, the math library is not actively used. It is included as part of the script's library imports but does not contribute to the script's current functionality.
19. **Flexibility and Readability:** Importing libraries like math even if they are not immediately used can provide flexibility for future code enhancements or modifications.
20. scikit-learn: Scikit-learn is a popular machine learning library in Python. Specifically, it is used for text processing and similarity analysis in this script. Two modules from scikit-learn are utilized:
21. TfidfVectorizer: This module is used to convert text data into TF-IDF (Term Frequency-Inverse Document Frequency) vectors, a common representation for text in natural language processing tasks.
22. cosine\_similarity: The cosine\_similarity function from scikit-learn calculates the cosine similarity between two TF-IDF vectors. It's used to measure the similarity between the content of the "Om" page and linked pages.

Program Overview:

1. **Importing Required Libraries**: The script begins by importing several Python libraries necessary for its operations. These libraries include wikipediaapi for interacting with Wikipedia, requests for making HTTP requests, BeautifulSoup for parsing HTML content, math for mathematical functions (although not actively used), and two modules from scikit-learn (TfidfVectorizer and cosine\_similarity) for text analysis and similarity calculation.
2. **Creating a Wikipedia API Object**: An instance of the wikipediaapi.Wikipedia class is created and named wiki\_wiki. This object serves as the interface for interacting with Wikipedia. The language code 'sa' is specified to access Sanskrit-language Wikipedia.
3. **Selecting the Seed Page:** The script specifies the title of the seed page to be accessed, which is set to "Om".
4. **Retrieving the Seed Page**: The script uses the wiki\_wiki.page() method to retrieve the Wikipedia page object for the seed page title ("Om").
5. **Accessing Links on the Seed Page**: The links on the seed page, which represent references to other Wikipedia articles related to "Om," are obtained and stored in the links variable.
6. **Calculating Cosine Similarity**: The script defines a function, calculate\_cosine\_similarity(), that calculates the cosine similarity between two text vectors. This function uses scikit-learn's TfidfVectorizer to convert text data into TF-IDF vectors and computes cosine similarity.
7. **Analyzing and Prioritizing Links**: For each linked page, it retrieves the page object, extracts the content, and calculates the cosine similarity between the "Om" page and the linked page's content.The URL of the linked page and its cosine similarity score are added to the url\_similarity\_list.
8. **Sorting and Displaying Results:** The list of URLs and their cosine similarity scores is sorted in descending order based on the similarity scores .The script retrieves and prints the top 5 URLs along with their cosine similarity scores, effectively ranking them based on similarity to the "Om" page.

Program

import wikipediaapi

import requests

from bs4 import BeautifulSoup

import math

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

# Create a Wikipedia API object with a user agent

wiki\_wiki = wikipediaapi.Wikipedia(

language='sa', # Set the language code for Sanskrit

user\_agent='YourAppName/1.0'

)

# Specify the title of the seed page you want to access (a Sanskrit page)

seed\_page\_title = "Om"

# Retrieve the seed page object

seed\_page = wiki\_wiki.page(seed\_page\_title)

# Access the links on the seed page

links = seed\_page.links

# Create a list to store URLs and their corresponding cosine similarity scores

url\_similarity\_list = []

# Define a function to calculate cosine similarity between two text vectors

def calculate\_cosine\_similarity(text1, text2):

vectorizer = TfidfVectorizer()

tfidf\_matrix = vectorizer.fit\_transform([text1, text2])

cosine\_score = cosine\_similarity(tfidf\_matrix[0], tfidf\_matrix[1])

return cosine\_score[0][0]

# Calculate cosine similarity scores and prioritize URLs based on similarity

for link\_title in links:

link\_url = "https://sa.wikipedia.org/wiki/" + link\_title

# Retrieve the page object for the linked page

linked\_page = wiki\_wiki.page(link\_title)

# Extract the content of the linked page

linked\_page\_content = linked\_page.text

# Calculate cosine similarity between the "Om" page and the linked page

similarity\_score = calculate\_cosine\_similarity(seed\_page.text, linked\_page\_content)

# Add the URL and cosine similarity score to the list

url\_similarity\_list.append((link\_url, similarity\_score))

# Sort the list of URLs based on cosine similarity scores (higher score first)

url\_similarity\_list.sort(key=lambda x: x[1], reverse=True)

# Retrieve and print the top 5 URLs along with cosine similarity scores

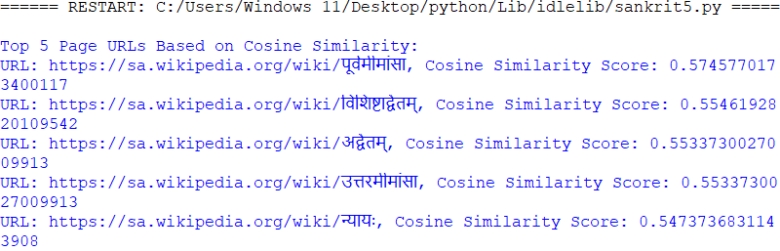
top\_5\_urls = url\_similarity\_list[:5]

print("\nTop 5 Page URLs Based on Cosine Similarity:")

for url, similarity\_score in top\_5\_urls:

print(f"URL: {url}, Cosine Similarity Score: {similarity\_score}")

Output



Conclusion

This program has been designed to conduct a systematic analysis and ranking of web pages sourced from Sanskrit Wikipedia. The central criterion for this ranking is the textual similarity to a designated seed page, namely, "Om." The code makes use of essential Python libraries, including "wikipediaapi" for accessing Wikipedia content, "requests" for web-based requests, "BeautifulSoup" for parsing HTML documents, and "scikit-learn" for text analysis.