**IMPORTING NECESSARY LIBRARIES :**

**A white background with black text

Description automatically generated**

Library Components:  
Import requests: A method of sending HTTP requests to web sites in order to receive their content.

Import BeautifulSoup: A tool for parsing HTML and XML documents and extracting data from web sites.

Natural Language Toolkit (nltk): A library used for different natural language processing activities.

Downloading NLTK Resources

nltk.download("punkt"): Downloads the tokenizer data necessary for word tokenization using the NLTK library.

nltk.download("stopwords"): Retrieves a list of commonly used stopwords for text processing operations.

Importing specific modules from NLTK

Stopwords: A list of frequent words such as "the," "a," "an," and so on that are often filtered out during text processing.

word\_tokenize: This function is used to tokenize text into individual words.

PorterStemmer: A stemming algorithm that converts words into their root or base form.

import defaultdict from collections: This imports the defaultdict class from the collections module, which provides an easy method to build a dictionary with default values.

Importing modules from tkinter

tk: The core module for designing graphical user interfaces.

Scrollbar: Adds scrollbars to widgets as needed.

A listbox is a widget that displays a list of things.

END: A constant in many tkinter widgets that represents the end index.

RIGHT: A constant that represents the correct alignment.

Y: A constant that represents vertical orientation.

**WEB SCRAPPING AND DATA EXTRACTION :**

**A computer screen shot of text

Description automatically generated**

create\_index(): This function accepts no parameters and returns an empty dictionary. For generating an index to hold data linked to publications. The dictionary returned is then used to store publishing data for later access or processing.

extract\_publications\_data(url): This function accepts a single parameter, url, which is the URL of the web page that contains a list of publications. It scrapes the specified URL to collect publication information such as title, publication date, authors, and links.

**CRAWLING WEBPAGES**

**A close-up of a computer screen

Description automatically generated**

The crawl\_and\_extract method makes it easier to crawl numerous pages of a website and aggregate publications data from each page into a single list. To extract data from each page, the programme calls the previously described extract\_publications\_data function.

**STEMMING AND FILTERING**

**A computer code on a white background

Description automatically generated**

preprocess\_text :function that accepts a text string as input and executes text preprocessing operations on it. The following stages are performed by the function using the Natural Language Toolkit (NLTK) library.Text normalisation, stopword removal, and word stemming are all popular text preparation techniques used in natural language processing and information retrieval tasks.

build\_inverted\_index(publications\_data): From a list of publishing data, this function creates an inverted index. An inverted index is a data structure that connects each term (or word) in the content of a publication to the documents (publications) in which it occurs. To normalise the terms, the function preprocesses the text content of each publication, deleting stop words and conducting word stemming. The dictionary term\_to\_documents is then created, where each term is a key that maps to a list of document IDs where the term appears. The function also adds a "Filtered Text" field to each publication, which contains the preprocessed content. The resultant term\_to\_documents dictionary is a quick method to find documents that include certain phrases.

A screen shot of a computer code

Description automatically generated

rank\_matched\_documents(query\_terms, term\_to\_documents): As input, this function accepts a list of query terms as well as an inverted index (in the form of the term\_to\_documents dictionary). It ranks the documents according to their relevancy to the query. The function searches up the term\_to\_documents dictionary for each query term to obtain a list of documents that include the word. It then counts the number of matches for each document. The greater a document's relevance score, the more frequently it matches query phrases. Finally, the method produces a list of document IDs sorted by relevance score in descending order. The resultant list displays the sorted papers that match the query keywords, with the most relevant document at the top.

**SEARCH FUNCTION**

**A screen shot of a computer code

Description automatically generated**

search\_publications(query, term\_to\_documents, publications\_data): This function is in charge into individual words, using Porter stemming to stem each word to its root form, and deleting frequent stop words. The rank\_matched\_documents function is then used to discover relevant documents that match the processed query words based on the inverted index.

A close-up of a computer code

Description automatically generated

\_\_main\_\_: This section of code acts as the program's entrance point. It specifies a base URL and the number of pages to crawl (num\_pages) in order to collect data from publications. It invokes the crawl\_and\_extract function (which is not displayed in the given code) to crawl the requested number of pages and extract the publication data from the website. It then uses the build\_inverted\_index function (not shown in the given code) to create an inverted index (term\_to\_documents) based on the extracted data. It starts a loop after establishing the appropriate data structures, allowing the user to submit search queries

***OUTPUT:***

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**