Scopus Scraper

Professor: Lorenzo Carnevale

Student: Zain Jamil

MAT: 531790



Table of Contents

* Introduction
* Create an api key
* Code explanation
* Running the code
* Refrences

INTRODUCTION

The purpose of this report is to present a Scopus Scraper application that extracts information about authors from the Scopus database using the Scopus API. The Scopus database is widely used in research, providing access to abstracts and citation information. This application aims to facilitate the retrieval of author information by utilizing the open-source APIs provided by Scopus.

The application allows users to input an author ID and retrieves relevant information such as the author's name, affiliation, and document count and …. By leveraging the Scopus API, the application automates the process of retrieving this information, saving researchers valuable time and effort.

And in the second step we extract the information within specific document of the Author by its associated it

Create an API key

before coding we should create an API key, An API key, is a unique identifier that grants access to an API (Application Programming Interface). APIs are used to allow different software systems to communicate with each other and exchange data.

In the context of the Scopus Scraper application, an API key is required to authenticate and authorize access to the Scopus API. The API key serves as a credential that identifies the application making the API request and ensures that only authorized users or applications can access the Scopus database.

A screenshot of a computer

Description automatically generated

The website URL is where you obtain information about the API and sign up to get an API

This is the URL that we use For the Scopus API, the website URL would be <https://api.elsevier.com/>

Code Explaintion

1.These lines import the necessary modules requests and json for making HTTP requests and parsing JSON data, respectively.

A screenshot of a computer

Description automatically generated

2.This try block contains the main logic of the code. It begins by prompting the user to enter an author ID and storing the input in the author\_id variable.

A black background with yellow text

Description automatically generated

3. first line constructs the URL for the API request. It uses an f-string to insert the author\_id variable into the URL query parameter. The constructed URL includes the Scopus API base URL (https://api.elsevier.com/content/search/author) followed by the query parameter query=AU-ID({author\_id}). The AU-ID is a specific search parameter used by the Scopus API to search for authors based on their unique identifier.

Then we define the headers to be included in the API request. Headers provide additional information about the request to the server. In this case, two headers are defined:

* "Accept": "application/json": This header specifies that the expected response format from the API is JSON. It tells the server to send the response in JSON format.
* "X-ELS-APIKey": it is a header that is used for authentication when making requests to the Scopus API.This header provides the API key as a means of authentication. It is required to access the Scopus API and ensure that only authorized users can make requests.

By including these headers in the API request, the code communicates to the Scopus API server the desired response format and authenticates the request using the provided API key.

A black background with orange text

Description automatically generated

4.This code sends an HTTP GET request to the specified URL (url) with the provided headers. The response from the API is stored in the response variable. raise\_for\_status() method is used to verify the success of the request by checking the status code

A black background with white text

Description automatically generated

5.This line parses the response text, which contains JSON data, into a Python dictionary. The json.loads() function is used to convert the JSON string into a dictionary. The parsed data is stored in the data variable.

A black background with colorful text

Description automatically generated

6.These lines extract specific information from the data variable. It retrieves the ORCID, author name, affiliation, document count, affiliation information (including city and country), and subject areas if available.

A screen shot of a computer code

Description automatically generated

7.These lines print the Author name, affiliation, and document count to the console.

A black background with colorful text

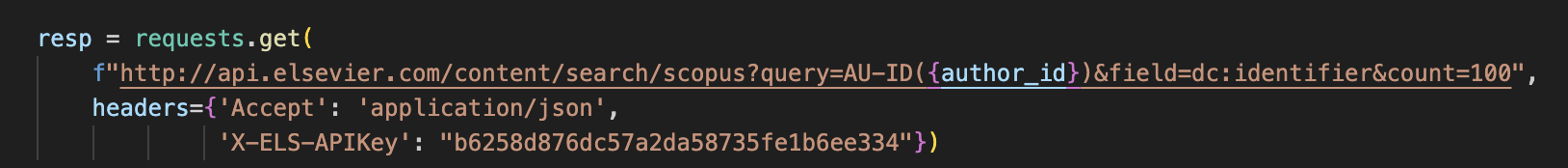
Description automatically generated

8.These lines check if certain information (ORCID, city, country, subject areas) is available and print them if present. If subject areas are available, it iterates over each area and prints the subject area, its abbreviation, and frequency.

A computer screen with text

Description automatically generated

9.This line makes another HTTP GET request to a different endpoint of the Elsevier API, retrieving additional metrics for a specific author that I will explain after running the code.



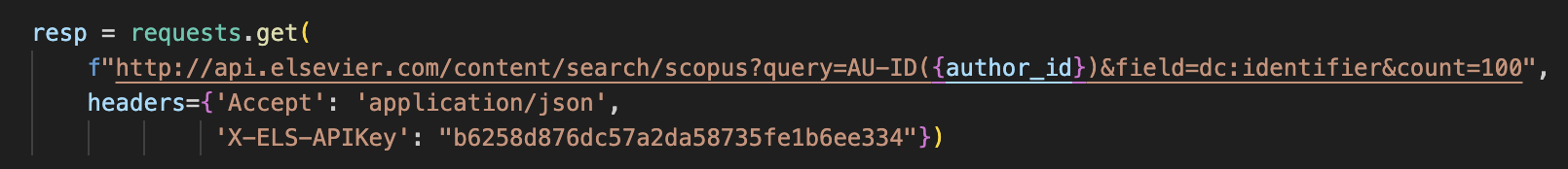
10.This line prints the JSON response of the additional author metrics, with pretty formatting (indented and sorted keys).

A black background with text

Description automatically generated

The json.dumps() function is used to serialize a Python object into a JSON-formatted string. The indent parameter specifies the number of spaces used for indentation in the output JSON string,sort\_keys=True argument instructs the function to sort the keys of the JSON object alphabetically,separators=(',', ': ') argument defines the separators used between the key-value pairs in the output JSON string. In this case, a comma (,) is used to separate the pairs, and a colon (:) is used to separate the key from the value.

11.This line makes a request to the Scopus API to search for documents associated with the provided author ID.



12.These lines parses the JSON response of the Scopus API request and store it in the results variable and then prints the Scopus document identifiers associated with the provided author ID.

A black background with colorful letters

Description automatically generated

The second line of code retrieves the identifier of each entry in the results dictionary and prints them as a list of lists, where each inner list contains a single identifier string.

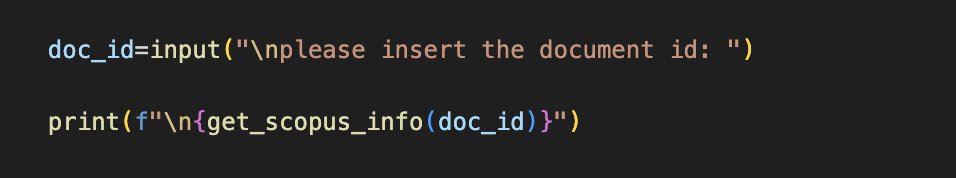
13.This code defines a function get Scopus info that retrieves information about a specific document based on its Scopus ID.

A screen shot of a computer program

Description automatically generated

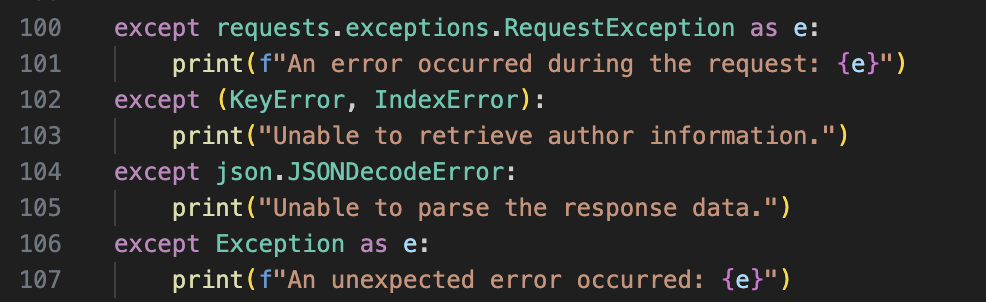
14. Next we prompt the user to insert the ID of the specific document of the writer that he wants to extract its information

And then we call the get\_scopus\_info function with a specific Scopus ID and prints the retrieved information about the document.



15. These except blocks handle different types of exceptions that may occur during the execution of the try block.

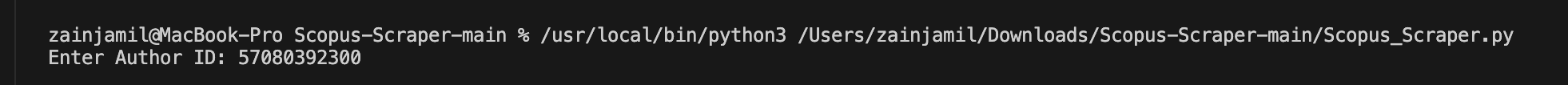
* requests.exceptions.RequestException handles exceptions related to the HTTP request, such as network connection issues.
* (KeyError, IndexError) handles exceptions that occur if the expected data structure is not found or the desired values are not present in the parsed JSON data.
* json.JSONDecodeError handles exceptions that occur if there is an error in parsing the response data as JSON.
* Exception is a generic exception that catches any unexpected errors during execution.



Running the code

1.The run terminal will prompt the user to insert an Author ID

so, we will insert : 57080392300



2.This is the extracted informations of the Author now i break down the output and explain each segment

A screenshot of a computer program

Description automatically generated

3.This is the JSON response received from the API. It contains detailed information about the author, such as their ID, name, document count, subject areas, and affiliation. It also includes links to related resources.

A screen shot of a computer

Description automatically generated

4.This part of the output is extracted from the data dictionary. In the code, after parsing the JSON response from the API into the data variable using json.loads(response.text), the relevant information is extracted from this dictionary.

A black screen with white text

Description automatically generated

As you can see this code will query the scopus database and extract Author’s name, Affiliation, Country, City, Orcid and Subject area and documents number

Subject area:These lines print the subject areas in which the author has published. It lists the abbreviations and frequencies of the subject areas.

Frequency, in this context, refers to the number of publications or occurrences related to a specific subject area for the author. It represents how frequently the author has published works in that particular field. The higher the frequency, the more prolific the author is in that subject area.

In the example provided, the author has published 1368 papers related to PHYSICS- Physics and Astronomy (All) 329 papers related to Engineering (ENGI), and 47 papers related to Mathematics (MATH).

5.The next part of the output is the JSON response from the API call to retrieve author metrics. It includes information such as the co-author count, citation count, cited-by count, document count, and h-index.

A computer screen with white text

Description automatically generated

In the output you provided, the section is related to the "co-author" information and the research impact of the author with the given ID (57080392300). Let's break down the key components:

1. "@\_fa": "true": This is an internal flag or indicator used by the system and is not directly relevant to the user.
2. "@status": "found": This indicates that the information for the author with the given ID was found in the system.
3. "coauthor-count": "669": This shows that the author with ID 57080392300 has collaborated with 669 other authors in various publications. It represents the total number of co-authors the author has worked with.
4. "coredata": This section contains some essential core data about the author, including:
   * "citation-count": " 88071": This indicates the total number of times the author's publications have been cited by other researchers.
   * "cited-by-count": " 28896": This represents the number of times the author's works have been cited by other papers or articles.
   * "dc:identifier": "AUTHOR\_ID: 57080392300 " This shows the ID of the Author that we are extracting his informations
   * "document-count": "1338": This shows the total number of documents. authored by the given author.
   * "prism:url": This is the URL that leads to more detailed information about the author
5. "h-index": "132": It is used to quantify the impact and productivity of a researcher's work.

5.The following part of the output is a list of document identifiers associated with the author. Each identifier is preceded by the "SCOPUS\_ID" prefix.

Finally, the code prompts the user to input a document ID and retrieves detailed information about that specific publication.

A black and white screen with white text

Description automatically generated

6.Finally, the code prompts the user to input a document ID and retrieves detailed information about that specific publication. The output displays the authors, title, journal name, volume, issue identifier, page range, cover date, article number, DOI, and the number of times the publication has been cited.

A screen shot of a computer screen

Description automatically generated

References

1. <https://api.elsevier.com/>
2. PYTHON DOCUMENTATION
3. [Support and assistance | PyCharm Documentation (jetbrains.com)](https://www.jetbrains.com/help/pycharm/getting-help.html)
4. [The Kitchin Research Group](https://kitchingroup.cheme.cmu.edu/blog/2015/04/03/Getting-data-from-the-Scopus-API/)