EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

ALGORITHMIC LITERACY: Generative Artificial Intelligence Technologies for Data Librarians

Alexandre Semeler[1], Adilson Luiz Pinto[2], Tibor Koltay[3], Thiago Magela Rodrigues Dias[4], Arthur Longoni Oliveira[1], José Antonio Moreiro González[5], Helen Beatriz Frota Rozados[1]

[1] Universidade Federal do Rio Grande do Sul, Brazil.
[2]Federal University of Santa Catarina, Brazil.
[3]Eszterházy Károly Catholic University, Hungary.
[4]CEFET-MG, Brazil.
[5]Carlos III University of Madrid, Spain.

Abstract

INTRODUCTION: Artificial intelligence (AI) is a novel type of library technology. AI technologies and the needs of data librarians are hybrid and symbiotic, because academic libraries must insert AI technologies into their information and data services. Library services need AI to interpret the context of big data.
OBJECTIVES: In this context, we explore the use of the OpenAI Codex, a deep learning model trained on Python code from repositories, to generate code scripts for data librarians. This investigation examines the practices, models, and methodologies for obtaining code script insights from complex code environments linked to AI GPT technologies.
METHODS: The proposed AI-powered method aims to assist data librarians in creating code scripts using Python libraries and plugins such as the integrated development environment PyCharm, with additional support from the Machinet AI and Bito AI plugins. The process involves collaboration between the data librarian and the AI agent, with the librarian providing a natural language description of the programming problem and the OpenAI Codex generating the solution code in Python.
RESULTS: Five specific web-scraping problems are presented. The scripts demonstrate how to extract data, calculate metrics, and write the results to files.
CONCLUSION: Overall, this study highlights the application of AI in assisting data librarians with code script creation for web scraping tasks. AI may be a valuable resource for data librarians dealing with big data challenges on the Web. The possibility of creating Python code with AI is of great value, as AI technologies can help data librarians work with various types of data sources. The Python code in Data Science web scraping projects uses a machine-learning model that can generate human-like code to help create and improve the library service for extracting data from a web collection. The ability of nonprogramming data librarians to use AI technologies facilitates their interactions with all types and data sources. The Python programming language has artificial intelligence modules, packages, and plugins such as the OpenAI Codex, which serialises automation and navigation in web browsers to simulate human behaviour on pages by entering passwords, selecting captcha options, collecting data, and creating different collections of datasets to be viewed.

Keywords: Generative Pre-trained Transformer, Algorithmic Literacy, Python, Open AI, Data Librarian

Received on 12 September 2023, accepted on 11 January 2024, published on 11 January 2024

doi: 10.4108/eetsis.4067

1. Introduction

Artificial intelligence (AI) is a novel type of library technology. AI technologies and the needs of data librarians are hybrid and symbiotic, because academic libraries must insert AI technologies into their information

and data services. Library services need AI to interpret the context of big data. An answer to the data deluge is the OpenAI Generative Pre-Trained Transformer (GPT) core technology [1] software agents, such as Perplexity AI [2] and ChatGPT [3], which are public tools developed using OpenAI Codex core technologies, and natural language processing-based software platforms, which make data and machine learning models accessible to the public.

According to [4], these AI software agents allow users to enter textual natural language commands and receive responses extracted from the knowledge acquired by the AI tool by machine learning done interactions with web sources. OpenAI (GPT) software agents can be used to create review and documentation code, images, text, simulations, and videos. These AI technologies allow researchers to explore the programming skills of data librarians and other Library and Information Science (LIS) professionals.

In the era of AI software, data librarians are responsible for managing and organising digital data. Data librarians are mainly involved in data management, modelling, and information retrieval. This ensures that data are stored, catalogued, and accessible to users. With the advent of AI technology and software tools, data librarians in the AI software era can utilise software tools, database systems, and information management platforms to handle and provide access to big data assets effectively.

Data librarians must apply OpenAI throughout each library. Data literacy is the always-on condition to the libraries, the problem caused by data deluge; the vast amount of data may be processed quickly, in volume, and various formats. A big data librarian can use AI to create new library services. OpenAI GPT technologies encompass more than software gadgets; OpenAI GPT tools are invaluable resources for researchers in numerous disciplines. AI allows for the comprehension of intricate processes such as memory, learning, and language. The use of AI has elicited discussions on human intelligence and exceptionalism. It simplifies and streamlines various facets of everyday life from the comfort of homes, schools, science labs, offices, movie theatres, and libraries [5].

Some key AI GPT technologies related to library work can replace or amplify the job of data librarians. One issue caused by these technologies is the obsolescence of professions, which has transformed the context in which academic libraries work. The autonomy of technology leaves no doubt that it may eliminate specific tasks, practices, and techniques. The question regarding the obsolescence of professions and their disappearance is not new. Work instruments have evolved with tools, machines, and devices replacing tasks previously performed by one or more people.

OpenAI tools proficient in programming can help create new data collection services on web sources such as reference databases, repositories, journal websites, digital libraries, and research data repositories. OpenAI code-proficient tools use the machine learning model used in data science projects to generate human code, which is applied to help create, improve, explain, and review codes and create unit tests [6,7].

A data librarian must develop hacking skills to adapt to the new needs of work routines with AI technologies and be able to create data collection techniques on diverse data sources in the context of large volumes of data and on a variety of dispersed data sources on the web. Algorithms are often necessary to develop complex processes. Automation of routine tasks by creating scripts is an effective method. OpenAI code provides a new form reference service, such as chatbots, automating functions in the design of translation services, and creating computer scripts or codes.

In this study, we propose the application of AI code-proficient software to data librarianship to enhance the possibility of creating repeated tasks, such as collecting information from web sources, by applying generative AI tool technologies. We propose an experiment where we write code in Python using the data librarian's natural language input instructions in OpenAI GPT tools that are proficient in programming to write the code.

In this context, the objective of AI technologies could be to interact with data librarians to expand their hacking skills, that is, the capabilities and speed at which they can create or revise code in programming languages, even if they have no experience as programmers.

Data librarians do not need to generate code; instead, they can use a code-proficient tool plugin in OpenAI GPT to do so. Data librarians do not need to become programmers but should understand the technical jargon to write code. Data librarians require data and algorithmic literacy concerning what is essential for writing code, such as programming logic, programming paradigms, knowledge of language compilers, and AI code-proficient applications. Software agents use generative AI deep learning and machine learning to quickly generate code and identify potential errors. In these horizons, data and algorithmic literacy have emerged as new aspects of digital literacy.

Thus, we propose an OpenAI-powered code creation method for nonprogramming data librarians. We utilise AI to create an OpenAI-based Python-proficient tool to generate scripts such as Machinet AI [8], aiXcoder [9], GPT-Mentor [10] and Bito AI [11] which are used to write, comment, check, and explain syntax code. These plugins require an application programming interface (API) in the OpenAI code; the integrated development environment (IDE) used for compilation is PyCharm Community [12] and the dataset deposit repository is Zenodo [13] In short, the data librarian applies algorithmic literacy techniques in the AI Python-proficient tool; the AI writes the code in the Python language [14], then the human data librarian analyses the code and compiles it using PyCharm. If there are errors, the librarian returns the code to the AI for tweaking, recompiles it, collects the data, and shares it in a data repository, such as Zenodo.

In the applied plan, this study elaborates on web scraping scripts using requests from a web data extraction

model based on OpenAI. Based on this experiment, it is possible to describe the data scraping codes in ORCID [15] Google Scholar Google Scholar [16], ScopusID [17], and Web of Science Researcher ID. We present five data collection problems. First, the ORCID is used to collect the IDs and names of authors. Based on these IDs, we create search lists to collect the citation and h-index data of such authors in Google Scholar, Scopus, and Web of Science Researcher ID [18]. The citation and h-index [19] data were chosen because they are easy-to-use and frontline research indicators for a given area of knowledge. The research front is composed of a set of the most-cited authors in a given field. It indicates the elite authors and is an easy-to-use indicator that is generally available in open sources. The case study was organised into the following topics:

a) data librarianship core competencies: hacking skills and algorithmic literacy.
b) algorithmic literacy: the essentials.
c) AI code-proficient tools.
d) web scraping logic for AI data extraction.
e) OpenAI-powered code script creation in generative artificial intelligence.
f) OpenAI generative artificial intelligence applies to web scraping.
g) natural language input describing a programming problem.
h) results and problems.

These AI tools allow non-programming data librarians to enter natural language commands and receive computer code to perform complex database analyses. This model is significant because it offers valuable possibilities for data librarians to develop Algorithmic Literacy, create with AI technology to automate tasks such as bibliometrics, and create innovative library services such as chatbots.

2. Data Librarianship Core Competencies: Hacker Skills and Algorithmic Literacy

Data-driven librarianship builds on a range of skills already incorporated and known by librarians [19]. Interest in research data generates new methods and practices that may contribute to and modify the investigations of these professionals. In a study on competencies in data librarianship, [20] concluded that a data librarian reflects some of the uncertainty of working in an emerging role in academic librarianship, and that digital, computational, and data literacy are fundamental for data librarian work.

Generally, data technologies occupy the work practices of data librarians; in their praxis, data are directly related to the digital technologies of manipulation, collection, and visualisation of datasets, which transform how they perceives their work activities. [21, 22] argued that data librarians and scientists should be knowledgeable about data literacy. This literacy is connected to information literacy, especially in its data-intensive paradigm, by including the possible roles of academic library work concerning data quality, management, curation, and citation. [19] also shares this view, defining a set of skills and abilities related to techniques and uses of research data as necessary for data librarians while emphasising data literacy with research data.

Indeed, we must be aware that "libraries have already been collecting and managing data, providing a higher level of support to researchers in this field had not been a regular part of their activities" [23]. Today, research and citizen data have acquired tremendous importance, motivating librarians to become data literate. However, awareness of data literacy has appeared only when information, media, and other forms of literacy are already known. On the other hand, data literacy was implicitly recognised in the documents addressing it. For instance, the 2018 definition of information literacy states that it relates to "information in all its forms: not just print, but also digital content, data, images, and the spoken word" [24].

Somewhat parallel to this development, librarians have begun to understand that they should not lose sight of "the multiplicity and interconnection of data literacy practices with other literacies" [25]. Data literacy involves numeracy, quantitative literacy, mathematical and statistical calculations, problem-solving, communication, and decision-making [26]. Fulfilling these functions is required to successfully "transform data into information and actionable knowledge by enabling one to access, interpret, critically assess, manage, and ethically use it" [21, 22]. It is also considered a way of knowing and a complex practice involving critical thinking [27]. The unmistakable presence of big data facilitated the appearance of further "derived" literacy constructs, such as personal data literacies.

Data infrastructure literacy encourages "critical inquiry, imagination, and intervention around the infrastructures for creating, using, and sharing data," which demands "going beyond data as an informational resource, and in this way equipping people with data skills in order to cultivate 'sensibilities for data sociology, data culture, and data politics'" [28]. The convergence among information literacy, data literacy, media literacy, and other literacy changed thinking about the impact of data. For example, [29] state that data literacy might potentially stop "the worst upshots of the information crisis". To enable this, librarians should foster their and their users' critical and active agency at a time "when society's datafication and algorithmically driven decision-making have become normalized".

Owing to the growing interest in methods and tools for visualising different phenomena, there literacy is also related to data literacy [30]. This type of literacy is "suitable to extract the correct information to complete a task and make an informed decision while minimizing the impact of biases" [31]. As to the educational use of data literacy, [32] enumerated the following eight common conceptions:

- Life skills for everyday problem-solving, enabling community engagement, citizen empowerment, activity tracking, and personal health management.
- Data protection, security, and privacy in social networks and personal data management.
- Data-based/data-driven decision making, supporting business strategy, classroom practice, library assessment, and implementing applied analytics.
- Essential business and learning analytics education: explaining data doubles and practising business ethics and professional conduct.
- Data-driven storytelling in media and business: using data visualization/infographics.
- Research skills for students and professionals to access existing datasets to produce and communicate new knowledge and make scientific experiments robust and reproducible.
- Building blocks and critical success factors for implementing data science in business, government, and research
- A new lingua franca or second language for organisations.
- Another view is that data science exploration in libraries is a new study area for data librarians as it involves issues related to the programming practice skills of a computer scientist. This view circulates the development of algorithmic literacy for collecting, manipulating, analysing, and visualising research data, which requires specific knowledge of programming languages and computational logic in addition to data literacy.

According to [33, 34, 35] data librarians must learn the general skills and competencies of data science. However, they do not need to become programmers, although they should be interested in learning computer languages and programming logic, which are skills linked to algorithmic literacy.

Algorithmic literacy relates to computer and digital literacy. Ridley and [36] explained that these terms are more specific to cyber literacy linked to computational thinking. The relationship between algorithmic thinking and authors is related to AI literacy. The authors reported that algorithmic thinking implies seeing algorithms as a part of everyday life and suggests a more profound and broader interpretation of algorithm literacy.

The hacking skills of a data scientist are fundamental to data and algorithmic literacy is a new aspect that needs to be investigated. These skills are directed at extracting data from remote locations where information is located and transforming it for analysis, with algorithmic literacy being a more fundamental part of the broader concept of AI.

3. Algorithmic Literacy: the Essentials for Non-Programming Data Librarian

As described by [37] the European Commission (2020) defines AI as a collection of technologies that combines data, algorithms, and computing power. The expertise of a data scientist in creating algorithms, flows, and models to be applied in any programming language allows an understanding of an algorithmic situation in the real world; therefore, creating algorithms is essential for data librarians interested in AI literacy. [38] defined a group of competencies in AI literacy, including data literacy and programmability, within their AI framework. They also stated that programmability is the property of being programmable, that is, the capability of being programmed.

Algorithmic literacy has emerged as a problematic aspect of AI technologies, algorithms as a mechanism for solving a well-specified computational problem and is the basis of programs. According to [36], algorithmic literacy refers to the comprehension of a new set of rules and promotion of abilities such that people can use algorithms without using them, which is a critical, novel and imperative challenge for libraries of all types.

An algorithm may be specified in a natural language such as English; alternatively, it may also be designed as a pseudocode or fuzzy logic. These pseudocodes, structured in English or the visual norms used to describe the algorithms, are not programming languages. [39] explained that an algorithm is a well-defined computational process that takes a value or set of values as input and produces a value or location of values as output, an algorithm specifies the relationship between the defined input and desired output and is said to be correct if it stops at the correct output for each input instance. The correct algorithm answers a given computational problem with sufficient computational literacy.

In this context, [36], explained that algorithmic literacy is recent. The general terms computer literacy and digital literacy have spawned more specific terms, such as cyber literacy, computational thinking, and algorithmic thinking. Generally, they are used for learning programming logic, and it is possible to think about computational procedures that focus only on the logic of an algorithm, without being distracted by the syntactic details of programming languages [40].

Data are stored in an always-on condition. Therefore, a data librarian must be able to apply algorithmic literacy to the most diverse AI technologies to manipulate digital data. Computer programming is essential for understanding digital data. Knowledge of programming languages and algorithms is the foundation of algorithmic literacy.

Algorithm literacy is a human ability related to computer programming skills to resolve data-deluge problems [28] and is fundamental for those about to start studying programming. According to [36], algorithmic literacy refers to the ability, experience, and awareness to understand and reason about algorithms and their processes, recognising and interpreting their use in systems (embedded or open), and creating and applying algorithmic techniques and tools to problems in various

domains. This involves gaining and understanding a new set of rules and nurturing skills and abilities so that people may use algorithms and not be used by them. Libraries typically advocate for accessible technology and practical use.

The simplest way to define algorithm literacy is by using a set of instructions executed in a computer language in a particular order to achieve a programming goal. Algorithm literacy involves the ability to construct a logical proposition, a true or false sentence, with conditions, recursion, looping, and all types of data that a computer can process. Understanding this will allow the creator of an algorithm to be proficient in any computer programming language because they are linked to how objects, flows, processes, and all the relations and events between them are represented.

4. AI Code-Proficient Tools

The choice of programming language depends on the type of scientific software being developed. Each type of programming language allows one or more algorithms to be used. For example, web scraping is coded in Python, whereas statistical analyses can be performed using R [20]. In the case of data collection, analysis, and visualisation from research data repositories, it is possible to combine two or more languages.

The Data Science Central website presented an article by [41] titled “Python Overtakes R for Data Science and Machine Learning” that summarised the trend of programming language usage by data scientists based on a series of proxy metrics using Google Trends [42]. The article concludes that the USA’s most commonly used languages in data science are Python and R. The Rice University website commends twelve computer languages for data science: Python, SQL, R, Visual Basic for Applications (VBA), Julia, JavaScript, Java, Scala, SAS, MATLAB, C/C++, and Swift. Thus, it is essential to present brief definitions of the framework and some packages and plugins, particularly for selenium [43].

Selenium is a portable framework for testing web applications. It provides a replay tool for creating functional tests without learning a test scripting language; for example, a test a surfed-in web page for data collection and for manually copying website information.

The main Python packages employed in this study, which are generally used in data science projects, are as follows: Beautiful Soup (BS), a class used to extract data from hypertext markup language (HTML) and XML files; comma-separated values, a package that implements classes to read and write tabular data in the CSV format, which is the most common import and export format for spreadsheets and databases; [44] a toolkit for compiling XML that uses C language classes, such as libxml2 and labels, and combines the functionality and completeness of these classes with the clarity of a native Python API, mostly XML standards compliant; Selenium, used for automating web applications for testing.

These Python programming language packages are considered the ideal tools for elaborating data services that include data collection, analysis, and visualisation, which are fundamental to developing a project based on the data science process in libraries.

The Python-proficient tool plugin is the OpenAI GPT, which is used in data science projects. It is a machine-learning model that can generate human-like code and is applied to help create, improve, explain, and review code as well as create unit tests. Examples include Machinet AI, a software agent that uses Chat with GPT-4; aiXcoder, an intelligent programming tool to recommend code snippets to improve coding efficiency and code quality; GPT-Mentor, an expert AI programmer agent that uses unit tests to verify the behaviour of a code; and Bito AI, used to write, comment, check security, and explain the syntax of a code. These plugins require an API in the Open AI code [8, 9, 10, 11].

Thus, we tested whether our application is functional and verified whether the script models created by AI are helpful in serving as applicable techniques in data science. Knowing whether a script can be translated into Python directly through these AI-proficient tools or indirectly using a graphical interface is essential for using software with executables or artificial intelligence modules.

5. Web Scraping Logic for AI Data Extraction

The fundamental procedure for elaborating the logic of an algorithm for web scraping is the standard approach for parsing a webpage into a representation to read HTML. The HTML markup of a website is dynamic because its appearance is updated; therefore, a web scraping algorithm must tolerate changes to the HTML markup as much as possible and be fast enough to be usable. For example, suppose a web service modifies a web page. In that case, the scraping algorithm must be modified to conform to new standards and adapter restrictions on the pattern, such as any HTML errors.

A web scraping algorithm needs to balance several aims. A web scraping looks for current copies of pages and must revisit them to detect changes. At the same time, it must discover new pages. Thus, a web scraping algorithm must have three primary policies: selecting the best quality pages first, revisiting them, which means updating the index when the pages change, and good education, the aim of which is to avoid overloading the pages.

Web information extraction from research data repositories through web scraping combines web content with systematic and automated metadata extraction methods. In this process, the scraping software AI agent simulates human browsing behavior on web servers by copying and rearranging disorganised data into organised data.

According to [45], web scraping operates as follows: (a) access to the website via a web communication

protocol (HTTP) during the request and response process between a client, typically a web browser, and a web server; (b) the scraper program parses the HTML and extracts its content; and (c) it stores the output as text to transform the extracted content into a structured representation for later analysis and storage. There are different search techniques for web scraping, including HTTP manipulation, data mining, scraping tools, manual copying, and microformats.

HTTP manipulation allows static and dynamic data collection from a website through an HTTP request. Data mining is an automatic process that recognises information on a website according to predefined scripts containing embedded data. Scraping tools are software used to extract information related to websites, functionalities, and data structures. They also extract data from social networks and are helpful in all web marketing activities. Although scraping tools are available, manual copying is sometimes necessary, for example, in cases where website information is blocked against any form of web scraping. Finally, microformats refer to semantic web technologies, such as open data files linked via metadata. These sets of information are generally designed for exchange through vocabulary and ontologies [45, 46, 47, 48].

The following section presents the operationalization methodologic experimentation with web scraping.

6. Openai-Powered Code Script for Non-Programming Data Librarian

This investigative current raises questions about practices, models, and methodologies for obtaining helpful code script insights from a heterogeneous and complex code environment linked to AI technologies elaborating human algorithm solicitation, that is, an AI Python-proficient tool.

OpenAI Codex is a deep-learning model trained on Python code from repositories. Provided with an NLP description of a programming problem as the input, Codex generates a solution code as the output. This is carried out by OpenAI Codex, a core AI in ChatGPT, Perplexity AI, and other popular AI tool agents. It is a descendant of GPT-4 and the base for Open AI plugins in the IDE PyCharm Community; as Machinet AI, this plugin can be used to create or modify files, fix errors, answer questions, and perform many other programming tasks. OpenAI Codex is a Python-proficient tool that includes different languages such as JavaScript, Go, Perl, PHP, Ruby, Swift, TypeScript, and Shell [1, 49, 50 51].

The AI-powered method code script can be used to create scripts for nonprogramming data librarians as an applied tool in IDE PyCharm and Machinet AI, which are dedicated to elaborating web scraping scripts through the requests of an AI-based web data extraction model.

Code scripts perform web scraping, conversion, and data visualisation. In short, the human data librarian includes the logic of the search algorithms in the AI, which writes the code in Python language, then the human data librarian analyses the code and compiles it using the PyCharm software. If errors occur, they return the code to the AI for tweaking, recompiling, data collecting, and sharing it in a repository, such as GitHub, Zenodo, or Mendeley.

The following section describes the interaction process between the Al agent and data librarian. These integrations require algorithm literacy on the part of the data librarian, so that they may understand the answers of AI agents.

7. Natural Language Description of a Programming Problem

The basic parameters and requirements for creating code for web scraping are communicating a description of a programming problem as input in a natural language, with the AI code generating a solution as its output, specifying the Python 3.11 version using the PyCharm compiler, and installing the following Python libraries: LXML, request, bs4, Selenium, and OS. Machinet AI and Bito AI, which are used for writing, commenting, checking security, and explaining code syntax, are also required. These plugins require APY in the Open AI code available at (https://platform.openai.com/account/api-keys).

Based on this experiment, it is possible to describe data scraping codes in ORCID, Google Scholar, Scopus, and Web of Science Researcher ID. First, ORCID collects the ID and names of the authors. Based on these IDs, we create search lists to collect citation and h-index data for such authors from Google Scholar, Scopus, and Web of Science Researcher ID. Citation and h-index data have been selected because they are easy-to-use indicators of front research in a given area.

Natural language is present at some level of difficulty. First, we introduce only information about the web source and needs; for example, collecting an H index on Google and saving a file. Then, we introduce the variables and libraries required in the code and algorithmic logic after introducing loops, conditions, and functions.

The following natural language instructions were given to generative artificial intelligence interaction.

8. Input: Natural Language Problem

In this section, we describe the natural language problem inputs and the resulting code scripts used to perform web scraping of the data extraction of the H-index (reference databases: ORCID, Google Scholar, Web of Science Researcher ID, and Scopus). The main problem is to collect the H-index from various sources and compare them to create an indicator. We use ten researchers in earth sciences for the experiment, and the lists may be amplified to conform to the algorithmic logic required by a library. We present five data collection problems.

8.1. Problem 1: Web Scraping ORCID Numbers

We ask the plugin to write the web scraping script in Python 3.11 and name the file ORCIDscraping_AuthorsIDS.py. To extract data from the ORCID, the script must read a list of names from a tab-separated value (TSV) file and save a file called Authors_names.tsv. The script extracts each profile and ORCIDs for all profiles; subsequently, it imports the following modules: codecs, requests, Beautiful Soup, and Selenium. Each name opens a webpage using Selenium, and Beautiful Soup is used to parse the HTML content. It then searches for the name and ORCID of the author in the parsed HTML and stores them in variables. If the values are not "N/A", the accumulator variables are updated with the extracted values. The script then writes the extracted information to the ORCIDresults.tsv file as “name, ORCID”. After processing all names and ORCIDs, the script closes the ORCIDresults.tsv file and exits the Selenium driver.

8.2. Problem 2: Web Scraping Google H Index

The plugin writes the web scraping script in Python 3.11, and we name the file GSCHOLARscraping.py. To extract data from Google Scholar, the script reads a list of links from a TSV file and saves a file called GSCHOLARresults.tsv. For each profile, the name, total citations, citations since 2018, h-index, and h-index since 2018 are used. It also calculates the total number of citations and the average h-index for all profiles; it must import modules such as codecs, requests, and Beautiful Soup. The script must create a counter variable that is initialised to track the progress of the extraction, and accumulator variables must be set up to store the total h-index, documents, and citations. The script then iterates through each link in the link_list. For each link, it opens the webpage using Selenium and Beautiful Soup is used to parse the HTML content. It then searches for the h-index, documents by author, and total citations in the parsed HTML and stores them in variables. If the values are not "N/A", the accumulator variables are updated with the extracted values. The script then writes the extracted information to the GSCHOLARresults.tsv file: name, total citations, citations since 2018, and h-index. After processing all links, the script closes the GSCHOLARresults.tsv file and quits the Selenium driver.

8.3. Problem 3: Web Scraping WOS Publons H Index

A web scraping script is written in Python 3.11 and the file is named WOSPublonscrapingAnon.py. To extract data from the WOS website, the script reads a list of links from a TSV file called WOSPublonsList.tsv, and writes the extracted information to another TSV file called WOSPublonsResults.tsv. The code must extract a list with the number of citations, H-index, articles, and reviews from the WOS database and save it in a TSV file. The code is a web scraper that extracts citation metrics of researchers from a list of links to their profiles on Publons. It writes the extracted information into a tab-separated file and calculates the average h-index, total citations, total articles, and total reviews for the entire list. Selenium and Beautiful Soup are used to navigate and parse webpages. The extracted information is written to a TSV file called PublonsResults.tsv. The script then writes the extracted information to the WOSPublonsResults.tsv file in the following formats: citations, H-index, articles, and reviews. After processing all the links, the script closes the WOSPublonsResults.tsv file and exits the Selenium driver.

8.4. Problem 3: Web Scraping Scopus H Index

The web scraping script is written in Python 3.11, and the file SCOPUSscrapingAnon.py is created to extract data from the SCOPUS website. The script reads a list of links from a TSV file called SCOPUSlist.tsv and writes the extracted information to another TSV file called SCOPUSresults.tsv. The script opens the SCOPUSlist.tsv file, reads the lines, and stores them in the variable link_list. It then opens the SCOPUSresults.tsv file for writing and adds the headers for ScopusID, H-Index, documents by author, and documents cited by. A counter variable is initialised to track the progress of extraction, and accumulator variables are set up to store the total h-index, documents, and citations. The script then iterates through each link in link_list. For each link, it opens the webpage using Selenium, and Beautiful Soup is used to parse the HTML content. It then searches for the H-Index, documents by author, and total citations in the parsed HTML and stores them in variables. If the values are not "N/A", the accumulator variables are updated with the extracted values. The script then writes the extracted information to the SCOPUSresults.tsv file in the following format: ScopusID, H-Index, documents by the author and cited by. After processing all the links, the script closes the SCOPUSresults.tsv file and exits the Selenium driver.

8.5. Problem 5: Web Scraping Compare and Compile The H Index

Finally, to analyse the data from GSCHOLARresults.tsv, WOSPublonsResults.tsv, and SCOPUSresults.tsv, we create a Python 3.11 script named CompareHindex.py. This script reads a list of H-indices and writes the extracted data to a TSV file called ComperHindexlistResults.tsv. The script opens and reads the lines of the H-Index list, storing them in the "hindex_list" variable. It then opens the ComperHindexlistResults.tsv file for writing and adds headers for GoogleID, ScopusID, WOSID, and mediaHindex. The script then iterates through each link in the H-Index list, writing the extracted information to the ComperHindexlistResults.tsv file in the following formats: GoogleID, ScopusID, WOSID, and mediaHindex. A counter variable is initialised to track the extraction progress, and accumulator variables are set up to store the total GoogleID, ScopusID, WOSID, and media H-Index.

9. RESULTS PROBLEMS AND SOLUTIONS

The results represent a public project for developing scripts and datasets produced by AI technologies for nonprogramming data librarians. The collected data, scripts, and packages are published following the Research Data Management Plan (PGD) and are available under the CC BY 4.0, licence in the Mendeley repository.

Table 1. Output ai code file and data files

File and Data Names	Links
ORCIDscraping_AuthorsIDS.py	
Authors_names.tsv	
ORCIDresults.tsv	
GSCHOLARscraping.py	
GSCHOLARlist.tsv	
GSCHOLARresults.tsv	
WOSPublonscrapingAnon.py	
Publonslist.tsv	
PublonsResults.tsv	
SCOPUSscrapingAnon.py	
SCOPUSlist.tsv	
SCOPUSresults.tsv	
CompareHindex.py	
readme.txt	
Chromedriver	
Geckodriver	
Venv	

The dataset, tab.1, contains all the files in the root directory with instructions for use, version data, contact information of the data authors, and technical descriptions for performing the analysis. The root folder contains scripts with the .py Python codes for the parser, data collection, and conversion; the data folder contains textual files (.tsv) containing the list of data and the env directory containing the Python libraries required to run the scripts.

The code created by AI must be corrected. The error was found. Therefore, when using an AI to generate code that will be used in web scraping, a possible recurring flaw is the lack of script testing by the AI regarding the expected output and the non-guarantee of operation, owing to different sites having different behaviours when it comes to HTTP requests made automatically. Another problem was that the script attempts to use one of the entire lines of the read file (in the list of links, GSCHOLARlist.tsv) instead of only the second item in each column, for example, in GSCHOLARscraping.py.

Before:

with codecs.open ('GSCHOLARlist.tsv ', 'r', 'utf-8-sig').readlines()[1:] as scholar:".

After:

with codecs.open('GSCHOLARlist.tsv', 'r', 'utf-8-sig') as scholar: for the next line to use ".readlines()[1:]".

Before:

for i, link in enumerate(link_list): link = link.strip() # remove leading/trailing whitespaces print(f'{i}: Extracting: {link}')

After:

for i, link in enumerate(link_list): link = link.split('\t')[1].strip() print(f'{i}: Extracting: {link}') # remove leading/trailing whitespaces

In summary, compared with the AI script, we eliminate some lines that could have their content executed in a single line to generate the "data_list" variable.

Finally, to reuse the dataset described in this paper, it is necessary to download the dataset from doi:10.17632/bzpytwcy88.1. The dataset can be reused for metrics studies and postgraduate methodologies.

10. Final Considerations

AI technologies such as OpenAI GPT have great potential for non-programming data librarians. These AI tools allow users to enter natural language commands and receive responses generated by the AI based on machine learning and interactions with web sources. Algorithmic literacy is a crucial aspect of nonprogramming data librarians. This involves understanding programming logic, languages, and algorithms to solve digital data-related challenges. Algorithmic literacy allows data

librarians to apply AI technologies and effectively manipulate digital data.

The specific task AI-powered code generation can assist data librarians with the potential implications of AI technologies for their roles and responsibilities of data librarians in academic libraries. The ability to quickly process large volumes of data from various sources is essential, and AI technologies, such as GPT, can assist in creating new library services. Data librarians can leverage AI tools to develop data collection techniques from diverse web sources and automate routine tasks through script creation. Using AI-powered code generation, data librarians can expand their programming capabilities without becoming expert programmers. In summary, generative AI technologies, such as OpenAI GPT, offer valuable possibilities for non-programming data librarians. By developing algorithmic literacy and leveraging AI-powered code generation, data librarians can enhance their ability to work with data, automate tasks, and create innovative library services.

AI may be a valuable resource for data librarians dealing with big data challenges on the Web. The possibility of creating Python code with AI is of great value, as AI technologies can help data librarians work with various types of data sources. The Python code in Data Science web scraping projects uses a machine-learning model that can generate human-like code to help create and improve the library service for extracting data from a web collection. The ability of nonprogramming data librarians to use AI technologies facilitates their interactions with all types and data sources. The Python programming language has artificial intelligence modules, packages, and plugins such as the OpenAI Codex, which serialises automation and navigation in web browsers to simulate human behaviour on pages by entering passwords, selecting captcha options, collecting data, and creating different collections of datasets to be viewed.

In conclusion, how does the proposed method facilitate collaboration between a data librarian and an AI agent? This paper highlights the partnership between data librarians unfamiliar with programming and AI agents using the OpenAI Codex to generate code scripts for web scraping. This approach empowers librarians to harness AI technology and automate data extraction tasks, thereby facilitating information management and analysis. The interaction between the AI agent and data librarian was described, emphasising the need for algorithm literacy to understand the responses of the AI agent. Overall, this study highlights the application of AI in assisting nonprogramming data librarians by creating code scripts for web-scraping tasks.

A critical issue for data librarians is their ability to code. Python proficiency tool is powerful for data librarians. Recently, algorithmic literacy competencies have been taught at information and library science schools. Its symbiosis with AI technologies has inspired new generations of data librarians, non-programmers, and AI specialists. These software agents are the final frontier between non-programmers and code generation, and to ensure that code is not a limit, AI requires algorithmic literacy. Therefore, data literacy is required for effective data management. Algorithmic literacy is necessary to create a process for handling data in large datasets, which require automation scripts to collect and analyse the data. Generative AI technologies are new methodologies for data librarianship and LIS professionals.

References

[1] OpenAI 2023. Retrieved from https://openai.com/

[2] Perplexity. 2023. Retrieved from https://www.perplexity.ai/.

[3] ChatGPT (2023). Retrieved from https://chat.openai.com/.

[4] Pavlik, J. V. Collaborating with ChatGPT: Considering the implications of generative artificial intelligence for journalism and media education. *Journalism and Mass Communication Educato*r, 2023; 78(1), 84–93. doi:10.1177/10776958221149577.

[5] Boden, M. *Artificial intelligence*: A very short introduction, London: Oxford, 2016.

[6] Lund, B. D., & Wang, T. Chatting about ChatGPT: How may AI and GPT impact academia and libraries?. *Library Hi Tech News*. 2023; 40(3), 26–29. doi:10.1108/LHTN-01-2023-0009.

[7] Radford, A. Narasimhan, K., Salimans, T., Sutskever, L. *Improving language understanding by generative pretraining*. 2018.

[8] Machinet AI. 2023. Retrieved from https://www.machinet.net/.

[9] Aixcoder. 2013. Retrieved from https://www.aixcoder.com/en/#/.

[10] ChatGPT Mentor 2023. https://plugins.jetbrains.com/plugin/21316-gpt-mentor

[11] Bito AI. 2023. Retrieved from https://bito.ai/.

[12] PyCharm. 2023. Retrieved from https://www.jetbrains.com/pt-br/pycharm/.

[13] Zenodo. 2023. Retrieved from https://zenodo.org.

[14] Python. 2023. Retrieved from https://www.python.org/.

[15] ORCID. 2023. Retrieved from https://orcid.org/.

[16] Google Scholar. 2023. Retrieved from https://scholar.google.com.

[17] ScopusID. 2023. Retrieved from https://scopus.com/. ID.

[18] ResearcherID. 2023. Retrieved from https://www.webofscience.com/wos/.

[19] Gold, A.. Cyberinfrastructure, data, and libraries, part 1: A cyberinfrastructure primer for librarians. *D-Lib Magazine*. 2007; 13(9/10). Retrieved from http://www.dlib.org/dlib/september07/gold/09gold-pt1.html doi:10.1045/september2007-gold-pt1

[20] Federer, L. Defining data librarianship: A survey of competencies, skills, and training. *Journal of the Medical Library Association.* 2018; 106(3), 294–303. doi:10.5195/jmla.2018.306.

[21] Koltay, T. Data literacy for researchers and data librarians. *Journal of Librarianship and Information Science*. 2017; 49(1), 3–14. doi:10.1177/0961000615616450.

[22] Koltay, T. Accepted and emerging roles of academic libraries in supporting research 2.0. *Journal of Academic Librarianship*. 2019; 45(2), 75–80. doi:10.1016/j.acalib.2019.01.001.

[23] Perrier, L., Blondal, E., & MacDonald, H. The views, perspectives, and experiences of academic researchers with data sharing and reuse: A meta-synthesis. *PLOS ONE*. 2020 15(2), e0229182. doi:10.1371/journal.pone.0229182.

[24] Chartered Institute of Library and Information Professionals. (2018). CILIP definition of information literacy 2018. Retrieved from https://www.cilip.org.uk/resource/resmgr/cilip/information_professional_and_news/press_releases/2018_03_information_lit_definition/cilip_definition_doc_final_f.pdf.

[25] Fotopoulou, A. Conceptualising critical data literacies for civil society organizations: Agency, care, and social responsibility. *Information, Communication and Society*. 2021; 24(11), 1640–1657. doi:10.1080/1369118X.2020.1716041.

[26] Fontichiaro, K., & Johnston, M. P. Rapid shifts in educators' perceptions of data literacy priorities. *Journal of Media Literacy Education*. 2020; 12(3), 75–87. doi:10.23860/JMLE-2020-12-3-7.

[27] Lloyd, A., & Hicks, A. Contextualising risk: The unfolding information work and practices of people during the COVID-19 pandemic. Journal of Documentation. 2021; 77(5), 1052–1072. doi:10.1108/JD-11-2020-0203.

[28] Gray, J. Jim Gray on eScience: A transformed scientific method. , 2009. In In: Hey, T.; Tansley, S.; Tolle, K. (Ed.). The fourth paradigm: data-intensive scientific discovery. Washington: Microsoft Research, 2009.

[29] Haider, J., & Sundin, O. Paradoxes of Media and information literacy: The crisis of information. London: Taylor & Francis; 2022.

[30] Carmi, E., Yates, S. J., Lockley, E., & Pawluczuk, A. Data citizenship: Rethinking data literacy in the age of disinformation, misinformation and malinformation. Internet *Policy Review*. 2020; 9(2), 1–22. doi:10.14763/2020.2.1481.

[31] Donohoe, D., & Costello, E. Data visualisation literacy in higher education: An exploratory study of understanding of a learning dashboard tool. *International Journal of Emerging Technologies in Learning*. 2020; 15(17), 115–126. doi:10.3991/ijet.v15i17.15041.

[32] Corrall, S. Repositioning data literacy as a mission-critical competence. 2019. Retrieved from http://d-scholarship.pitt.edu/id/eprint/36975.

[33] Burton, Matt and Lyon, Liz and Erdmann, Chris and Tijerina, Bonnie. The future of data science in libraries. Project Report. Pittsburgh, PA: University of Pittsburgh; 2018. Retrieved from http://scholarship.pitt.edu/33891/.

[34] Semeler, A. R., Pinto, A. L., & Rozados, H. B. F. Data science in data librarianship: Core competencies of a data librarian. *Journal of Librarianship and Information Science.* 2019; 51(3), 771–780. doi:10.1177/0961000617742465.

[35] Stuart, D. Practical data science for information professionals. London: Facet Publishing; 2020.

[36] Ridley, M., & Pawlick-Potts, D. Algorithmic literacy and the role for libraries. *Information Technology and Libraries*. 2021; 40(2). doi:10.6017/ITAL.V40I2.12963

[37] Cox, A. M., & Mazumdar, S. Defining artificial intelligence for librarians. *Journal of Librarianship and Information Science.* 2020 0(0). doi:10.1177/09610006221142029.

[38] Long, D., & Magerko, B. What is AI literacy? Competencies and Design considerations. In Conference on human factors in computing systems (CHI). 2020; doi:10.1145/3313831.3376727.

[39] Cormen,T.H., Leiserson, C.E., Rivest, R.L., Stein, C. Introduction to algorithms. Cambridge MA: MIT Press; 2009.

[40] Dalbey, J. Pseudocode standard. 2001. Retrieved from http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html.

[41] Granville, J. Data science central. 2017. Retrieved from https://www.datasciencecentral.com/python-overtakes-r-for-data-science-and-machine-learning/.

[42] Google trends. 2023. Retrieved from https://trends.google.com/trends.

[43] Selenium. 2023. Retrieved from https://www.selenium.dev/.

[44] LXML. XML and HTML with Python. 2023 Retrieved from http://lxml.de.

[45] Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., & Fdez-Riverola, F. Web scraping technologies in an API world. *Briefings in Bioinformatics*.

2014; 15(5), 788–797. Retrieved from http://bib.oxfordjournals.org/content/15/5/788. doi:10.1093/bib/bbt026.

[46] Carle, V. [KTH, Skolan för elektroteknik och datavetenskap (EECS)], Web scraping using machine learning. 2020. Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-281344 (Thesis).

[47] Diouf, R., Sarr, E. N., Sall, O., Birregah, B., Bousso, M., & Mbaye, S. N. Web scraping: State-of-the-art and areas of application. In: *IEEE International Conference on Big Data* (Big Data); 2019. doi:10.1109/BigData47090.2019.9005594.

[48] Webster, S. What is scraping? The basics for everyone. 2015. Retrieved from https://myhelpster.com/what-is-scraping-the-basics-for-everyone.

[49] Rice computer science. 2023. Retrieved from https://csweb.rice.edu/academics/graduate-programs/online-mds/blog/programming-languages-for-data-science.

[50] Brennan, R. W., & Lesage, J. Exploring the Implications of OpenAI codex on Education for Industry 4.0. 2023. doi:10.1007/978-3-031-24291-5_20.

[51] Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. The robots are coming: Exploring the implications of OpenAI codex on introductory programming. Paper presented at the ACM International Conference Proceeding Series. 2022. doi:10.1145/3511861.3511863.