Intelligent Agents: Development Team Project Report

Group 1: Maria Ingold, Camille Grimal, Abdullah Alaskar

INTRODUCTION

The academic research publication landscape is expanding, posing challenges for comprehensive evidence synthesis (Gusenbauer & Haddaway, 2020). While this report does not aim to encompass all research sources, it explores crucial considerations for the development of online academic research using intelligent agents.

The Academic Research Online System (AROS) leverages agents, small autonomous programs, to sift through online sources and extract research data based on predefined search terms. The gleaned data undergoes systematic processing, storage, and presentation tailored for academic researchers.

This proposal outlines the key system requirements, design decisions, approaches, rationale, and challenges of developing a basic academic research system.

SYSTEM REQUIREMENTS

Overview

Ascertaining the AROS system requirements requires a basic understanding of web scraping.

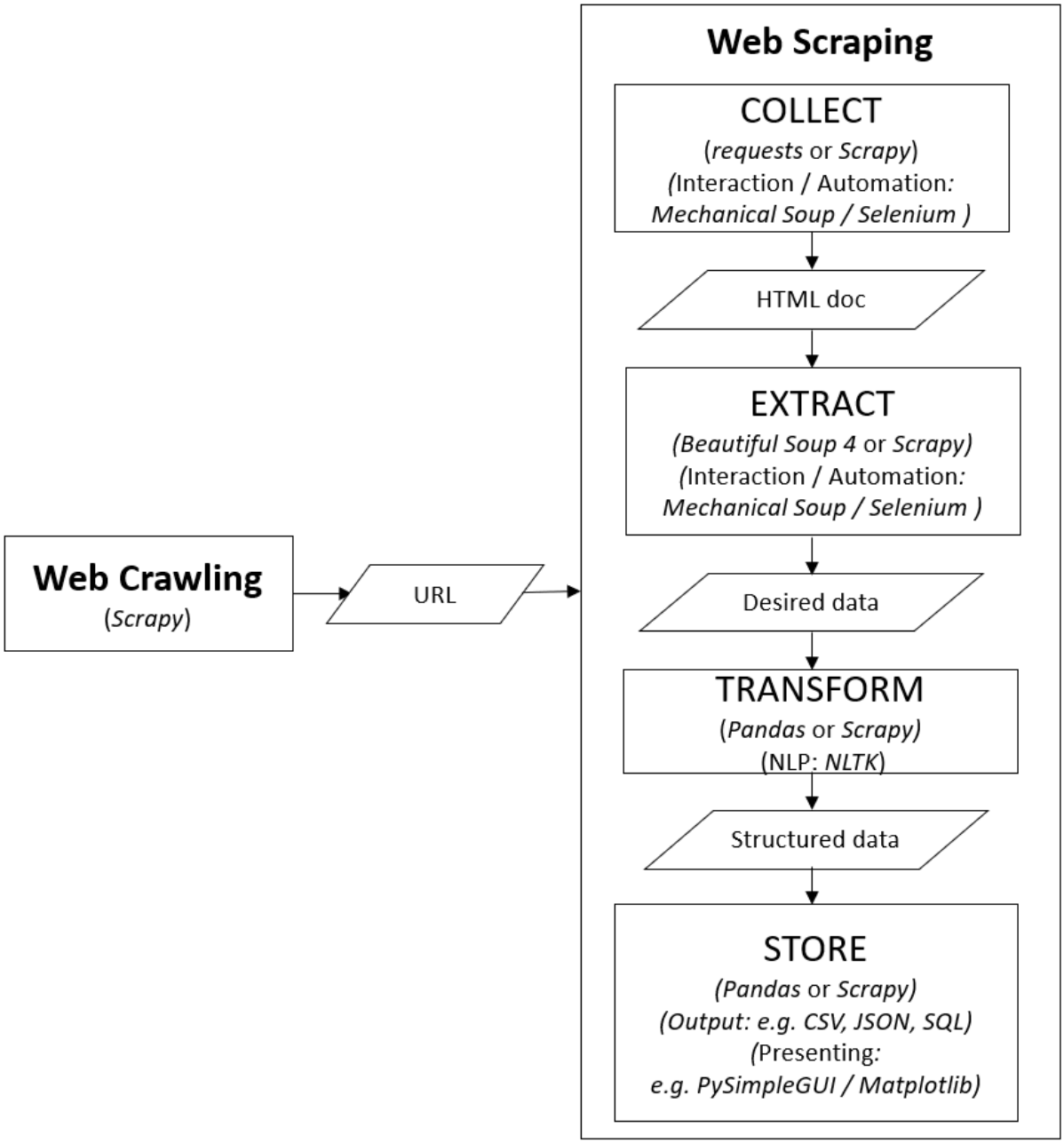

FIGURE 1 | Web scraping components

Figure 1 shows web scraping is composed of crawling and scraping (Khder, 2021). Web crawling automatically and methodically fetches, understands, and indexes webpages, while web scraping automatically extracts data from those pages and structures it (vanden Broucke & Baesens, 2018; Asikri, Krit & Chaib, 2020). Web scraping is composed of collecting the HyperText Markup Language (HTML) content

from a web page, parsing and extracting the data, transforming, and storing or presenting it (Hillier, 2021; Khder, 2021; Amos, 2022).

Programming Language

Python is recommended for its readability, maintainability, extensive support, and its ability to complete tasks in a unified language (Hillier, 2021; Khder, 2021; Abodayeh et al., 2023). As an interpreted language, Python mitigates the need for re-compilation and deployment when modifying agent functionality (Asikri et al., 2020).

Libraries and Frameworks

Python offers several options for HTML scraping, with libraries like *requests*, *Beautiful Soup*, and *Pandas* combined for access, parsing and extracting, and transformation and output respectively (Nigam & Biswas, 2021; Abodayeh et al., 2023). While this blend suits beginners and smaller projects, *Scrapy* is ideal for medium projects with more extensive tasks, offering features like web-crawling, faster speed, scheduling, and parallel computing (Asikri, Krit & Chaib, 2020).

Both Scrapy and Pandas have built-in support for comma-separated values (CSV) and JavaScript Object Notation (JSON), while Pandas also supports Structured Query Language (SQL) (Asikri, Krit & Chaib, 2020).

For more advanced systems, interaction and automated browsing can be provided by *MechanicalSoup* or *Selenium* and natural language processing (NLP) by the *Natural Language Toolkit* (NLTK) (Mehta & Pandi, 2019; Van Dijk, Staut & Wolfs, 2019; Amos, 2022). For small static sites using HTML and XML, Beautiful Soup outperforms Selenium for runtime, central processing unit (CPU) and memory usage, performance, and simplicity for beginners, but Selenium is better for automation,

JavaScript, interaction, and dynamic web pages, with both on par for reliability (Morina & Sejdiu, 2022).

DESIGN DECISIONS AND APPROACH

Design Decisions

TABLE 1 | Domain Specifics

What	Description
URL	https://arxiv.org/abs/2109.00656
Element	meta name
Data	citation_title citation_author citation_date

Given the team's early-stage expertise, AROS starts as a simple web-scraper, searching an arxiv.org Uniform Resource Locator (URL), identifying and extracting key citation data, transforming it into a usable format, and storing it in a CSV file (Table 1). Simple, static, and small-scale benefits from Python's requests, Beautiful Soup 4 and Pandas libraries. Beautiful Soup's *html.parser* is used because, while *lxml* parser is faster, it requires C (Abodayeh et al., 2023). Modular design enables migration for scale and extension for functionality.

For simplicity, this starts as local client-based, not cloud. For the future, client-server—with client handling user interaction and server managing data extraction and processing—could help scale and modularity. Similarly, instead of a relational database (SQL), CSV is used, and any user interface is text-based not graphical. Regardless, modularity and error handling provide a foundation for automation.

Methodology

While AROS does not yet handle all goals or learning in Figure 2, the diagram shows how a modular approach maps web scraping components to goal-based agent design (Wooldridge, 2009; Russell & Norvig, 2021).

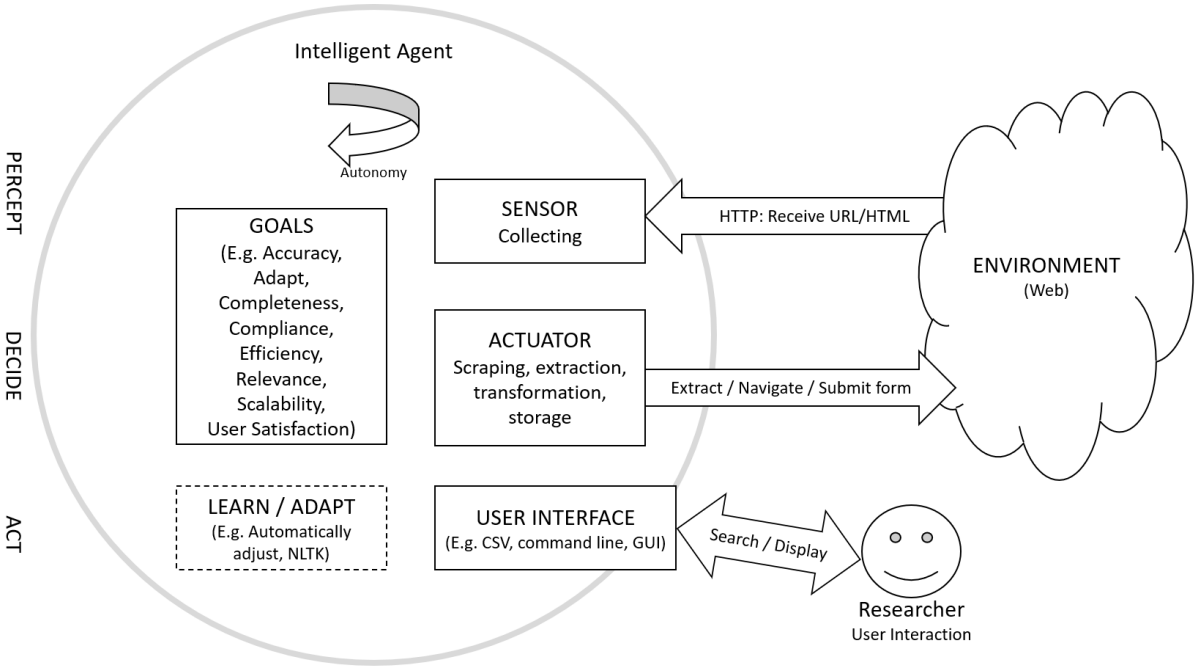

FIGURE 2 | Intelligent Agent AROS (Simple Reflex Agent)

This report follows a “PRINCE2 Agile” methodology. PRINCE2 to understand the business objective, product description and shared goal, and Agile to encourage modular iteration of small, prioritised chunks (Cooke, 2016).

Development Techniques

Approaches to web scraping include the following rough order of complexity, scalability, and automation: 1) copy-paste 2) string-based 3) regular expressions 4) DOM tree-based 5) string-based 6) semantic framework and 7) machine learning-based (e.g. NLP or computer vision) (Khder, 2021; Nigam & Biswas, 2021).

At its simplest, AROS balances moderate complexity with high scalability and automation using the client-side Document Object Model (DOM) tree (Khder, 2021; Nigam & Biswas, 2021). Beautiful Soup uses HTML parsing to convert the page content into a DOM parse tree (Abodayeh et al., 2023).

Code is managed under version control (Table 2).

TABLE 2 | Tools and packages

Tool / Package	Description
GitHub	Open-source version control system
Visual Studio Code	Free source code editor (works with GitHub)
python	Programming language
pip	Python package installer
requests	HTTP requests
beautifulsoup4	Web scraping and parse tree
pandas	Data structures and analysis

The structure—collect, extract, transform and store—from Figure 1, becomes code modules like Figure 3.

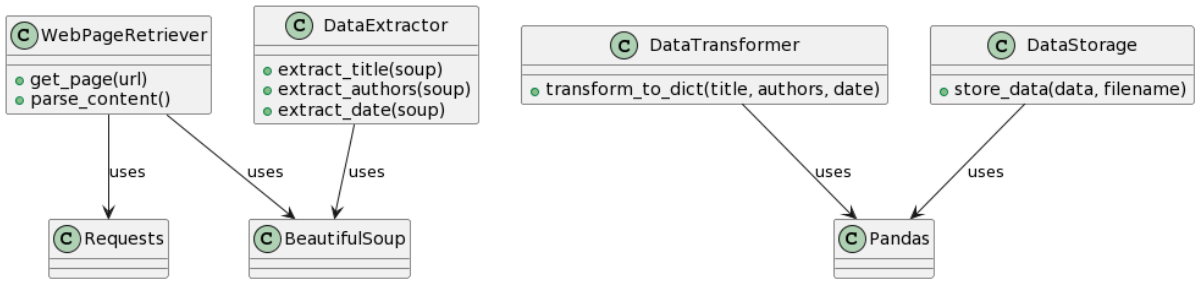

FIGURE 3 | Class diagram (PlantText)

CHALLENGES

Data

Web scraping brings challenges, notably the ever-changing nature of data. A code sample for finance.yahoo.com, functional in 2021, now delivers a 404 error (Nigam & Biswas, 2021). AROS must grapple with data variety, volume, velocity, and veracity, which requires specifying collected data and ensuring regular updates (Krotov & Johnson, 2023).

Technology

Web scraping, exemplified by AROS, demands fault tolerance and adept error handling (Huber et al., 2022). Navigating representational errors and networking issues necessitates extending code functionality or adapting.

The further challenge will be moving AROS from a simple reflex agent to a model-based reflex agent (internal state of visited pages) and extending to goal-based (planning links to follow) or beyond to utility-based agents (Russell & Norvig, 2021).

Legislation and Ethics

Legislation surrounding unauthorized access, usage policies, copyright infringement, and ethical considerations like privacy and bias must be taken into consideration (Khder, 2021; Krotov & Johnson, 2023). AROS should adhere to ethical practices, avoiding private data, respecting robots.txt, and implementing rate limiting.

CONCLUSION

In conclusion, AROS is a simple reflex agent that performs basic web-scraping of an academic research site using Python, requests, Beautiful Soup and Pandas. Using a PRINCE2 Agile methodology to both plan and iterate a modular design, AROS can evolve to a more sophisticated model-based or goal-based agent. Future extensions could add libraries and frameworks, such as Scrapy (crawling), Selenium (automation) and NLTK (NLP), while considering error handling, fault tolerance and ethical design.

REFERENCES

Abodayeh, A., Hejazi, R., Najjar, W., Shihadeh, L., & Latif, R. (2023) Web Scraping for Data Analytics: A BeautifulSoup Implementation, *Proceedings - 2023 6th International Conference of Women in Data Science at Prince Sultan University, WiDS-PSU 2023* 65–69. DOI: https://doi.org/10.1109/WIDS-PSU57071.2023.00025.

Amos, D. (2022) *A Practical Introduction to Web Scraping in Python – Real Python*. Available from: https://realpython.com/python-web-scraping-practical-introduction/ [Accessed 10 December 2023].

Asikri, M.E., Krit, S. & Chaib, H. (2020) Using Web Scraping In A Knowledge Environment To Build Ontologies Using Python And Scrapy, *European Journal of Translational and Clinical Medicine* 7(3): 433–442. Available from: https://www.researchgate.net/publication/346215371_Using_Web_Scraping_In_A_Knowledge_Environment_To_Build_Ontologies_Using_Python_And_Scrapy [Accessed 10 December 2023].

Beautiful Soup (N.D.) *Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation*. Available from: https://beautiful-soup-4.readthedocs.io/en/latest/# [Accessed 12 December 2023].

vanden Broucke, S. & Baesens, B. (2018) From Web Scraping to Web Crawling, *Practical Web Scraping for Data Science* 155–172. Available from: https://doi.org/10.1007/978-1-4842-3582-9_6.

Cooke, J.L. (2016) PRINCE2 Agile: An Implementation Pocket Guide: Step-by-step Advice for Every Project Type, *IT Governance Publishing* 42–65.

Van Dijk, R.H.W., Staut, N. & Wolfs, C.J.A. (2019) Automated Sentiment Analysis of Text Data with NLTK, *Journal of Physics: Conference Series* 1187(5): 052020. DOI: https://doi.org/10.1088/1742-6596/1187/5/052020.

Gusenbauer, M. & Haddaway, N.R. (2020) Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources, *Research Synthesis Methods 11*(2): 181–217. DOI: https://doi.org/10.1002/JRSM.1378.

Hillier, W. (2021) *What Is Web Scraping? [A Complete Step-by-Step Guide]*. Available from: https://careerfoundry.com/en/blog/data-analytics/web-scraping-guide/ [Accessed 10 December 2023].

Huber, S., Knoll, F. & Döller, M. (2022) A Pipeline-oriented Processing Approach to Continuous and Long-term Web Scraping, *Proceedings of the 17th International Conference on Software Technologies (ICSOFT 2022)* 441–448. DOI: https://doi.org/10.5220/0011275100003266.

Mehta, S. & Pandi, G. (2019) An Improving Approach for Fast Web Scrapping Using Machine Learning and Selenium Automation, *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* [Preprint]. Available from: https://www.academia.edu/42899763/An_Improving_Approach_for_Fast_Web_Scrapping_Using_Machine_Learning_and_Selenium_Automation [Accessed 11 December 2023].

Khder, M.A. (2021) Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application, *Int J Advance Soft Compu Appl* 13(3). DOI: https://doi.org/10.15849/IJASCA.211128.11.

Krotov, V. & Johnson, L. (2023) Big web data: Challenges related to data, technology, legality, and ethics, *Business Horizons* 66(4): 481–491. DOI: https://doi.org/10.1016/J.BUSHOR.2022.10.001.

Morina, V. & Sejdiu, S. (2022) Evaluating and comparing web scraping tools and techniques for data collection., *UBT Knowledge Center* 53–57. Available from: https://www.researchgate.net/publication/369114323_Evaluating_and_comparing_web_scraping_tools_and_techniques_for_data_collection [Accessed 29 November 2023].

Nigam, H. & Biswas, P. (2021) Web Scraping: From Tools to Related Legislation and Implementation Using Python, *Innovative Data Communication Technologies and Application* 149–164. DOI: https://doi.org/10.1007/978-981-15-9651-3_13.

Russell, S. & Norvig, P. (2021) *Artificial Intelligence: A Modern Approach, Global Edition*. 4th ed. Pearson Education, Limited.

Wooldridge, M. (2009) *An Introduction to MultiAgent Systems*. Chichester, UK: John Wiley & Sons.