1. **Understanding the Research Repository Structure -** By exploring the site, I identified the ability to use its search functionality and corresponding API to generate precise URLs for topics of interest. This approach enabled efficient scraping by allowing me to target only relevant articles. To include abstracts and ensure a more comprehensive text dataset, I also decided to scrape individual article pages. I avoid duplicates by checking for unique article titles.
2. **Examining Site HTML and Identifying Tags -** Using Google Chrome's developer tools, I inspected the site's relatively static HTML structure and identified key elements containing metadata. I also accounted for pagination to ensure complete data collection.
3. **Scraping the Repository -** I employed Python's ***requests*** and ***BeautifulSoup*** libraries to first scrape the targeted search URLs, extracting the individual article URLs. For each article, I scraped metadata fields such as titles, abstracts, and other relevant content, storing the data in dictionaries. These were subsequently converted into a ***pandas*** DataFrame and exported as a structured CSV file.
4. **Keyword Frequency Analysis -** To analyze the text, I concatenated all article titles and abstracts into a single corpus and lowercased everything. Using Python's ***re*** package, I removed punctuation and digits, tokenized the text with ***NLTK***, and filtered out stop words using ***NLTK's*** stopwords list. Finally, I performed word counts with ***pandas*** and visualized the results using the ***plotnine*** library, creating bar charts to showcase the most frequent keywords.