Goodreads Scraper Methodology Write-Up
Olivia Pivovar | BACSS Lab Research Fellow Co-op Trial Submission

---

1. Book Matching

To identify the correct Goodreads page for each book, the script reads the provided goodreads_list.csv, which includes the title, author, and book ID. For each entry, the script constructs a Goodreads search URL by URL-encoding the book title. It then parses the resulting HTML using BeautifulSoup to extract book links with the bookTitle class.

To avoid irrelevant matches, it filters out URLs containing keywords like "summary," "study-guide," or "analysis." To further improve accuracy, we implemented fuzzy string matching using the rapidfuzz library. For each search result, the script compares the concatenated title and author string against the search result's display text using partial_ratio, accepting matches above a configurable threshold (currently set to 75). This approach improves robustness to inconsistencies in title formatting and minor variations in search results.

Early in development, I used generative AI (ChatGPT) to help identify HTML tags and structure. For example, I would inspect a Goodreads review card and ask for help turning the observed class names into robust scraping selectors. Over time, I grew more confident navigating the DOM manually and identifying appropriate tags on my own, which reduced my reliance on AI for low-level scraping tasks.

---

2. Review Navigation

Once the correct book page is found, the script clicks the "More reviews and ratings" button to navigate to the full reviews section. It then iterates through paginated review cards dynamically loaded on the page.

Each review is scraped from elements with the ReviewCard class and includes:

- Reviewer ID
- Review rating
- Review date
- Review text
- Upvotes
- Comments
- User shelf tags

To adhere to performance expectations and ethical scraping practices, the script limits itself to collecting 100 reviews per book (or any number specified via the REVIEW_LIMIT_PER_BOOK constant). This strikes a balance between dataset richness and runtime feasibility.

Pagination is handled by repeatedly clicking the "Show more reviews" button until either the desired number of reviews is collected or no more reviews are available. The script avoids duplicates by checking if a reviewer_ID has already been stored for the book.

---

3. Challenges and Solutions

Several real-world issues were encountered during development:

- Login Persistence: Used session cookies to avoid repeated manual logins. After the first login, cookies are saved and reloaded in future runs to maintain authentication automatically.

- Session Timeouts: Occasionally, the browser session would disconnect mid-run. This was solved by catching exceptions like InvalidSessionIdException and reinitializing the driver when necessary.
- Missing Elements: Some books lacked the "More reviews and ratings" button or had zero reviews. The script logs these and moves on.
- Dynamic Content Loading: Goodreads dynamically loads reviews, so explicit waits and JS scroll commands were used to ensure elements were loaded before parsing.
- Resumability: To support long runs across hundreds of books, the script:
    - Saves progress after every book to reviews_output.csv
    - Checks how many reviews each book already has before scraping
    - Only adds reviews that are new and not already in the file
    - Can be stopped and resumed via Ctrl+C without losing progress
- Fuzzy Book Matching: I discovered that over 40 books were not being matched correctly even though they existed on Goodreads. This was due to small variations in title formatting or additional bracketed metadata (e.g., "[Paperback]" or edition notes). To fix this, I:
    - Implemented fuzzy string matching using rapidfuzz.partial_ratio.
    - Pre-cleaned titles by removing bracketed/parenthetical segments.
    - Ensured that the match is similar enough in title and, importantly, by the same author.

These changes significantly improved quality and efficiency, especially for a dataset with 470+ books and tens of thousands of reviews.

---

4. GenAI Usage

In the early stages of development, I used ChatGPT to help interpret Goodreads' dynamic HTML structure. For instance, I would identify specific elements from the page source (like review text containers or buttons), and ChatGPT assisted in writing the appropriate Python code to extract or interact with them using BeautifulSoup and Selenium.

Over time, I became more confident in navigating and understanding the page structure myself. As I gained experience, I began identifying the relevant HTML patterns independently, relying on AI less for direct code and more for second opinions and refinements.

Beyond HTML parsing, I used ChatGPT for advice on debugging complex issues and, more importantly, for improving the scraper's robustness and efficiency. Key architectural strategies — such as saving progress incrementally, checking how many reviews already exist per book, enforcing deduplication, resuming from interruptions, and integrating a pause/resume workflow — were ideated and refined through iterative discussions with ChatGPT.

AI played a supportive role throughout this project as it helped me expand my understanding of HTML parsing and develop ideas to improve the program's efficiency and accuracy.