Lobster Notes Web Scraper

Abstract

The Lobster Notes Web Scraper is a modular system of Python-based programs developed to collect, organize, and structure educational materials from publicly accessible online learning platforms. These scrapers support sources such as Khan Academy and MIT OpenCourseWare, extracting metadata, videos, documents, and other learning resources. The system produces standardized JSON outputs compatible with the Lobster Notes database, enabling automated integration of open-access educational content into research and learning tools.

Introduction

The Lobster Notes Web Scraper project was designed to facilitate the automated collection of structured data from online educational platforms. Each scraper is optimized for a specific platform but follows a consistent architecture, ensuring that output data across multiple sites can be processed uniformly. The system adheres to ethical scraping practices, limiting requests and complying with publicly available content policies.

System Overview

The Lobster Notes Web Scraper consists of multiple standalone Python scripts, each corresponding to a supported educational platform (for example, khan_scraper.py for Khan Academy and mit_scraper.py for MIT OpenCourseWare). All scrapers share a common framework that includes HTML parsing, headless browsing, content filtering, and standardizing data for output. The system's outputs conform to a unified schema that categorizes extracted materials as resources, notes, videos, documents, images, or website entries.

Features

- Multi-Platform Support: Modular scrapers tailored for various educational sites.
- Automated Metadata Extraction: Titles, descriptions, and content types are extracted directly from HTML structures and meta tags.
- Video and Document Identification: Embedded media, YouTube links, and PDFs are detected and cataloged.
- Content Filtering: Uses an external configuration file (ignore_links.json) to skip irrelevant or duplicate content.
- Standardized Output: All collected data conform to the Lobster Notes JSON schema.
- Polite Operation: The scraper adheres to respectful crawling practices, including limited request rates and headless browsing.

Data Cleaning and Normalization

Data cleaning and normalization are integral components of the Lobster Notes Web Scraper. Because online educational platforms often contain repetitive, redirected, or inconsistent links and metadata, each scraper applies several post-processing steps to ensure that the collected data is accurate, consistent, and ready for integration.

The cleaning process included the following key operations:

- Duplicate Removal:
  - The scraper maintains internal sets to track previously seen URLs, image sources, and resource identifiers. Any duplicate links or assets encountered during scraping are automatically excluded from the final dataset.
- Ignored Link Filtering:
  - Each scraper references the ignore_links.json configuration file, which defines specific URLs and keywords to be excluded. Links containing these patterns, or matching predefined ignored words, are discarded early in the extraction process. This prevents the inclusion of advertisements, login prompts, or irrelevant navigation links.
- URL Normalization:
  - Relative URLs are combined with their base page URLs to form complete, absolute links. YouTube and PDF links are normalized to standardized formats to ensure consistent referencing across different runs and sites.
- Metadata Validation:
  - The scraper performs validation of critical fields such as titles, descriptions, and durations. Empty or malformed values are removed, truncated, or replaced with defaults to maintain schema integrity.
- Text Truncation:
  - Long text fields, including titles, descriptions, and file names, are truncated to safe lengths suitable for database storage. This ensures compatibility with downstream systems and avoids data overflow.

Program Usage

The Lobster Notes Web Scraper is designed to run locally using Python, requiring no elevated system permissions. Each scraper, cleaning script, and validator operates as a standalone module, enabling users to scrape, clean, and validate educational content from supported platforms. The following instructions outline how to install the environment, run the scrapers, and generate standardized JSON outputs suitable for loading into the Lobster Notes database.

1. Clone the Repository

Begin by downloading the project from the Git repository:

- git clone --branch scraping_branch https://github.com/nikkigorski/cos457_course_proj.git lobsternotes
- Navigate to the scraping module:
- cd 'lobsternotes/Scraping scripts'

2. Create and Activate a Virtual Environment

The scrapers are written in Python and require isolated dependencies. Create a virtual environment:

- python3 -m venv testenv
- source testenv/bin/activate

Install all required packages:

- pip install -r requirements.txt

3. Running the MIT Scraper

To extract content from an MIT OCW page, pass the target URL and desired output filename:

- python3 mit_scraper.py https://ocw.mit.edu/example mit_output.json

After scraping, clean and validate the data:

- python3 data_cleaner.py mit_output.json && python3 validate.py mit_output_cleaned.json

4. Running the Khan Academy Scraper

To scrape a Khan Academy lesson page:

- python3 khan_scraper.py https://www.khanacademy.org/example khan_output.json

5.Cleaning and validation:

- python3 data_cleaner.py khan_output.json && python3 validate.py khan_output_cleaned.json

Conclusion

The Lobster Notes Web Scraper provides a standardized, extensible, and ethically compliant framework for collecting and organizing open educational resources. By combining browser automation, structured data extraction, and consistent output formatting, the system facilitates the integration of educational materials from diverse sources into unified research and study environments.

References

1. BeautifulSoup Documentation. (n.d.). Beautiful Soup Documentation. Retrieved from https://www.crummy.com/software/BeautifulSoup/
2. Selenium Documentation. (n.d.). Selenium with Python. Retrieved from https://www.selenium.dev/documentation/
3. yt-dlp Documentation. (n.d.). yt-dlp: A youtube-dl fork with additional features and fixes. Retrieved from https://github.com/yt-dlp/yt-dlp
4. Khan Academy. (n.d.). Khan Academy. Retrieved from https://www.khanacademy.org
5. Massachusetts Institute of Technology. (n.d.). MIT OpenCourseWare. Retrieved from https://ocw.mit.edu