ChatGPT

Report on Google Scholar Scraper Input URL Issue and Solutions

Background

The `google_scholar_scraper.py` script is designed to collect author information and publication lists from Google Scholar by using the third-party **scholarly** library. The script takes a Google Scholar author profile URL, extracts the `user` identifier and then uses the library's `search_author_id` method to fetch data. The method accepts a Google Scholar ID and returns basic author details [1]. In our testing, we discovered that users encountered issues when passing certain URLs, especially those containing many query parameters and the `&` character. These problems prevented the script from running correctly.

Original Issues

1. **Shell interpretation of `&` in URLs.** In Unix-like shells, the ampersand (`&`) is interpreted as a command separator that runs the preceding command in the background. When users typed a command such as:

```
python3 google_scholar_scraper.py https://scholar.google.co.kr/citations?hl=en&user=hdV_QJMAAAAJ&view_op=list_works
```

the shell executed `python3 … https://scholar.google.co.kr/citations?hl=en` in the background, and attempted to run `user=hdV_QJMAAAAJ…` as a separate command. The blog post Curl, unquoted URLs, and LANGSEC explains that an unquoted URL containing an ampersand is interpreted as two commands [2]. This behaviour applies to any command, including our script. Consequently, the script did not receive the full URL.

1. **Limited robustness in extracting the author ID.** The original `extract_author_id` function assumed that the query parameter `user` would always appear in lowercase and did not provide a fallback for other cases or for missing parameters. If the URL used uppercase letters or different domains (e.g., `.co.kr`), the function might fail to find the ID. Additionally, it offered no alternative means to supply the author ID directly.

2. **Mandatory positional URL argument.** The script required a positional URL argument even when users might have preferred to supply the `scholar_id` directly. This created friction for users and offered no solution when quoting was forgotten.

Solutions Implemented

To resolve the problems described above, several modifications were made to the script.

1. Robust URL parsing

The `extract_author_id` function now:

- Normalizes all query-parameter names to lowercase before searching for `user`, ensuring that variations such as `User` or `USER` are supported.
- Ignores the domain and focuses on the query parameters, so `.co.kr`, `.com` and other country domains are treated the same.
- Returns `None` gracefully if the URL cannot be parsed, rather than crashing.

These changes make author ID extraction more tolerant of varied inputs while still returning the correct identifier (e.g., `hdV_QJMAAAAJ`) [1].

2. New `--author-id` option

A new optional argument `--author-id` was added. When this option is supplied, the script bypasses URL parsing entirely and uses the provided ID directly. For example:

```
python3 google_scholar_scraper.py --author-id hdV_QJMAAAAJ
```

This approach completely avoids shell-quoting problems. It also aligns with the `scholarly` library's `search_author_id` method, which fetches author information by ID [1].

3. Optional positional URL and improved help messages

The positional `url` argument is now optional. The script checks that either a URL or an author ID is provided; if neither is present, it shows a descriptive error. The help text explicitly warns users to quote URLs containing `&` characters so the shell does not treat them as command separators. The **Brain on Fire** article emphasises that the correct solution to unquoted ampersands is to wrap the URL in single or double quotes [2], and this advice is now integrated into the script's usage instructions.

4. Enhanced error messages and documentation

If the script cannot extract an author ID from the URL, it now suggests using the `--author-id` option. This provides clear guidance instead of a generic parsing error. Internal documentation and docstrings were updated to describe these behaviours and to note the importance of quoting URLs.

Demonstration Example

Below is a side-by-side example showing how to run the updated script using a quoted URL and using the new `--author-id` option.

Method	Command Example	Notes
Quoted URL	`python3 google_scholar_scraper.py "https://scholar.google.co.kr/citations?hl=en&user=hdV_QJMAAAAJ"`	Double quotes ensure the `&` characters are treated as part of the URL rather than command separators [2].

Method	Command Example	Notes
Direct ID	`python3 google_scholar_scraper.py --author-id hdV_QJMAAAAJ`	No URL quoting is required; the ID is passed directly and processed via `search_author_id` [1].

These examples demonstrate how both approaches achieve the same result while addressing the original quoting issue.

Considerations and Contrarian Viewpoint

While the modifications significantly improve the user experience, it is important to note some caveats:

- **Legal and ethical considerations.** Google Scholar does not provide an official API, and scraping can violate terms of service. Overuse may trigger rate limits or IP blocks. As an alternative, official services like Semantic Scholar offer robust APIs for research data and should be considered for critical workflows.
- **Maintenance burden.** The `scholarly` library depends on the current structure of Google Scholar's pages; changes on Google's side can break the scraper. Thus, relying on scraping may require frequent code updates.

A small percentage of users may prefer to avoid quoting altogether by escaping ampersands (e.g., `\&` in bash). However, quoting remains the more reliable and portable solution [2].

Conclusion

The original input URL issues stemmed from shell interpretation of unquoted ampersands and limited parameter parsing. By improving the URL parser, adding a `--author-id` option, and refining argument handling, the `google_scholar_scraper.py` script can now handle complex Google Scholar URLs and provides clear guidance to users. The updated script encourages safe quoting practices and offers a direct ID input path, making data collection more robust while acknowledging the limitations and risks associated with scraping.

---

[1] Quickstart — scholarlyORG 1.0b1 documentation
https://scholarly.readthedocs.io/en/stable/quickstart.html

[2] Curl, unquoted URLs, and LANGSEC | Brain on Fire
https://www.brainonfire.net/blog/2017/04/01/curl-unquoted-urls-langsec/