

CS410 Project Documentation

Topic: Chrome extension for Google Scholar

Software overview

This software is a Chrome extension for Google Scholar. When you are on your Google Scholar author page and turn on the extension, it adds the following three functionalities:

1. Provide the count of independent citations for each of your publications listed. the original Google Scholar author page only shows the total citations, regardless of whether it's a self-citation (cited by yourself and your coauthors) or an independent citation.
2. Make a list of the authors who cited your publications, with their Google Scholar links.
3. For each of your publications, find the most relevant articles to the publication from its citations, make it a recommended reading list to you.

Software implementation

The software is a package following the “client-server” paradigm where the client side includes a popup-style chrome extension (of manifest version 3), and the server is based on Node.js's Express web-application framework (<https://expressjs.com/>). The client side is responsible for scraping the Google Scholar author page to collect publication and author information. The information is then sent to the Express server via Http post request in JSON format. The server listens for the requests and for each request, the JSON post request is parsed and fed to a Python program which uses the search engine technique and ranker function BM25 for text retrieval. The program generates a ranking file whose content is sent back by the Express server to the client for parsing and display.

Originally, I planned to use local host as the server, but the Google Chrome Extension new safety protocol discourages the intrusive extension that communicates directly to the local file system. So, I hosted a dedicated server on AWS (Amazon Web Services) so that user does not need to handle server locally. However, in case that the AWS server is down (I use a free version of AWS so it is likely), or user wants to use his/her own server (e.g., local host), user simply needs to change the server IP address in the code. I will give detailed instructions in the appendix.

Project Structure

- |—— client-codes → This folder should be downloaded and unpacked in Google Chrome
 - |—— background.js → Send post request to server wait for response
 - |—— button.css → CSS File for the popup extension's "Go" button
 - |—— content_script.js → Setup event listener of the "Go" button in Chrome extension
 - |—— images → Folder contains icon images of the Chrome extension
 - |—— manifest.json → Main configuration file of the Chrome extension
 - |—— popup.html → Front-end interface of the Chrome extension
 - |—— popup.js → Setup event message sender of the "Go" button in Chrome extension
- |—— server-codes → This folder is copied to server and ran by "node index.js" command
 - |—— index.js → Entry point of the Express server, listens for incoming requests
 - |—— install.sh → Script for installing Express framework in Node.js
 - |—— node_modules → Folder created by Node.js after installing
 - |—— package-lock.json → File created by Node.js after installing
 - |—— package.json → File created by Node.js after installing
 - |—— search_ranker_test → Folder for search engine ranking program
 - |—— config_A → Meta file 1 for making a dynamic configuration file
 - |—— config_B → Meta file 2 for making a dynamic configuration file
 - |—— env.yml → Anaconda Python environment file
 - |—— line.toml → Meta file 3 for configuration
 - |—— run.sh → Main script for preparing running environment for ranking program
 - |—— search_test.py → BM25 ranking program that's called by "run.sh" script
 - |—— stopwords.txt → Stop words used for BM25 ranking program

Algorithms

Functionality 1 & 2: Count Independent Citation and Author Information Summary

To collect citation information, the Chrome extension loops through all publications displayed on the Google Scholar author page and follows the hyperlinks of author's publication citations. It counts the number of independent citations, i.e., the one that's not cited by yourself or your coauthors. Web scraping is needed to collect such information. Luckily, in Google Scholar, all citation links have the same unique class ID (Figure 1). However, since Google Scholar only shows limited number of publications in the citation page, to collect all citations the program needs to recursively follow the "next page" link (Figure 2 and Figure 3).

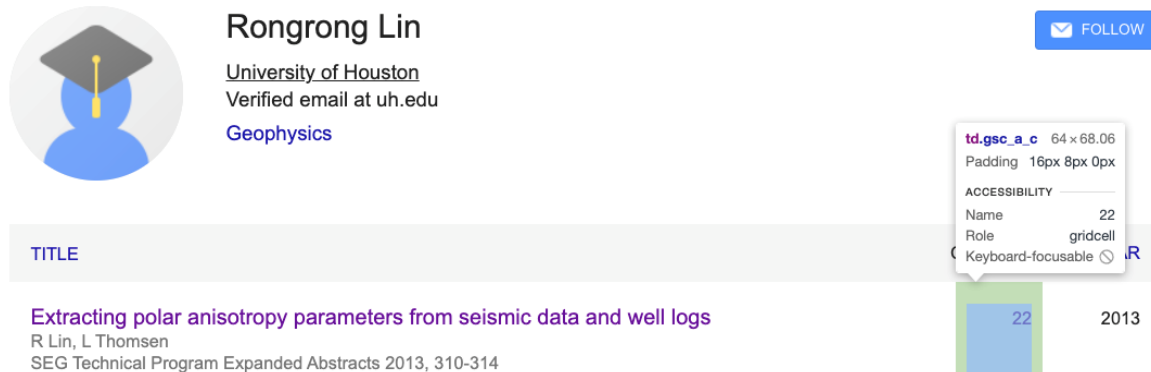


Figure 1. Citation information can be scraped following its unique ID.



Figure 2. Due to the limited results shown in each Google Scholar page, I need to follow the “next” page link. A DOM “querySelectorAll” with filtering the “innerText” is sufficient to find the link.

```

115     ... // find url to next page
116     ... let nextUrl = null;
117     ... doc.querySelectorAll("#gs_n_a").forEach((node) => {
118     ...     if (node.innerText === "Next") {
119     ...         nextUrl = node.href;
120     ...     }
121     ... });

```

Figure 3. Code snippet for finding the “next” page URL.

Using the above strategies, I can collect all the publication and author information to build a relational database. To avoid complication and keep things simple, I just used a JavaScript “Object” data structure to store the information.

During the testing, I was frequently blocked by Google for repeated and short burst http requests sent. This is due to the recursively following the “next” page for the publication that got cited by a large numbers of papers. To restrict the maximum recursion depth, an input “Max Depth” is provided. The default “Max Depth” is 3. The extension GUI is shown in Figure 4. Users click the “Go” button to trigger all 3 functionalities of the extension and the extension will modify the original author page to add additional information. The independent citation count will be displayed within the parentheses after the total citation count (Figure 5). Also, a summary table with author information scraped from looping the citations will be displayed in the popup GUI (Figure 6).

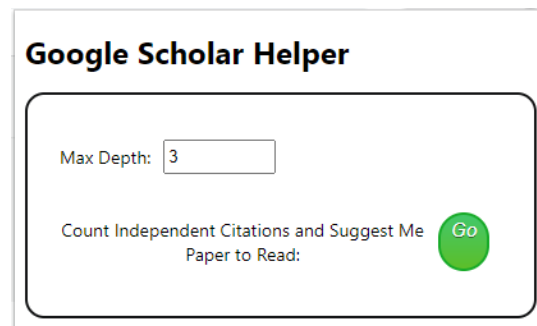


Figure 4. Google Chrome Extension GUI. The input field “Max Depth” controls the maximum number of “next” pages to follow during information retrieval stage. Users click “Go” button to trigger all 3 functionalities of the extension and the extension will modify the original author page to add additional information. Finally, a summary table with author information will be displayed in the popup GUI.

Google Scholar Author Page: Rongrong Lin

University of Houston
Verified email at uh.edu
Geophysics

Publication: Extracting polar anisotropy parameters from seismic data and well logs
R Lin, L Thomsen
SEG Technical Program Expanded Abstracts 2013, 310-314

Citation Data: 22(19) 2013

Recommended reading list from citations:

- Rank 1: University of Houston, 2Delta Geophysics
- Rank 2: Bayesian lithology analysis with seismic-driven likelihood to anisotropic medium: a case study from Northwest Australia
- Rank 3: A two-step anisotropic seismic AVO inversion strategy for the VTI media
- Rank 4: A two-step anisotropic inversion for the VTI media based on a three-term AVO equation
- Rank 5: Converted-wave contributions to anisotropic shale behavior
- Rank 6: Rock physics model of shale: Predictive aspect
- Rank 7: Seismic amplitude variation with offset inversion in a vertical transverse isotropy medium
- Rank 8: Interpreting azimuthal Fourier coefficients for anisotropic and fracture parameters
- Rank 9: P-and PS-wave vector wavefields for anisotropic petrophysics
- Rank 10: Estimation of Brittleness and Anisotropy Parameters in Transversely Isotropic Media With Vertical Axis of Symmetry

Figure 5. Google Scholar author page gets updated after user clicking “Go” in the popup extension. In this example, “22(19)” means that there are 22 papers citing current paper while 19 of them are independent citations. Among all the cited papers, the relevance to the current paper is ranked and the ranking results are displayed by the prefix “Rank #”. The hyperlinks are extracted from Google Scholar and user can click to follow to the paper webpage. Currently only the top 10 ranked paper are displayed.

Google Scholar Helper

Max Depth:

Count Independent Citations and Suggest Me Paper to Read: Go

Summary

Name	Full Name	Affiliation	Citation Number	Cited Paper(s)
JE Downton	Jonathan Downton	CGG	1	Extracting polar anisotropy parameters from seismic data and well logs(1)
B Roure	Unknown	Unknown	1	Extracting polar anisotropy parameters from seismic data and well logs(1)
F Zhang	Feng Zhang	Professor of Geophysics, China University of Petroleum	7	Extracting polar anisotropy parameters from seismic data and well logs(7)
T Zhang	TUO ZHANG	GFZ German Research Centre for Geosciences	1	Extracting polar anisotropy parameters from seismic data and well logs(1)
XY Li	Unknown	Unknown	5	Extracting polar anisotropy parameters from seismic data and well logs(5)
FC Loh	Unknown	Unknown	1	Extracting polar anisotropy parameters from seismic data and well logs(1)
BL Chuah	Unknown	Unknown	1	Extracting polar anisotropy parameters from seismic data and well logs(1)

Figure 6. Summary table generated after user clicking “Go” button. Author’s full name and his/her affiliation are extracted if he/she has a Google Scholar account. The column “Citation Number” represents the total number of papers he/she published that cite the current author’s paper(s) in the author page.

Functionality 3: For each of your publications, find the most relevant articles to the publication from its citations, make it a recommended reading list to you.

For this functionality, the publication title will serve as the “query” and the abstracts of the articles that cite this publication will serve as the “documents” to be searched from. The searching program will return a list of the top 10 documents that are most relevant to the query, i.e., the 10 most relevant articles to your publication. The list will be in the order of most relevant to the least relevant.

The relevance level is determined by the BM25 ranking function which is one of the most popular ranking functions used in text retrieval. (https://en.wikipedia.org/wiki/Okapi_BM25). The searching program uses the BM25 function from the python package metapy (<https://github.com/meta-toolkit/metapy>). Before applying the BM25 ranker, some preprocessing such as tokenization with unigram representation, stopwords removal are also applied from the metapy library.

In the case that a publication has less than 10 citations, or less than 10 relevant citations, the returned list will certainly be of length less than 10.

Each returned list will be displayed right beneath each publication on the Google Scholar author page (Figure 5).

How to use the software

How to install

To install the chrome extension (client side), user needs to turn on the developer mode in Chrome and unpack the project folder (for detailed instruction, please refer to Google's documentation at

<https://developer.chrome.com/docs/extensions/mv3/getstarted/#manifest>).

Client-side Installation

1. Go to <https://github.com/R2Lin/CourseProject> and download the ZIP file which contains all the source code (Figure 7).
2. Unzip the source code on your machine.
3. Go to `chrome://extensions/` and turn on "Developer mode" (Figure 8 A). Please do not use "Incognito" mode since all extensions are disabled in the mode.
4. Click "Load unpacked" button (Figure 8 B) and choose the unzipped folder "client-codes" from step 2 (Figure 9). You will see it is successfully loaded (Figure 10).
5. Click the extension button and pin it to the address bar (Figure 11). You will see it successfully pinned on the address bar (Figure 12).

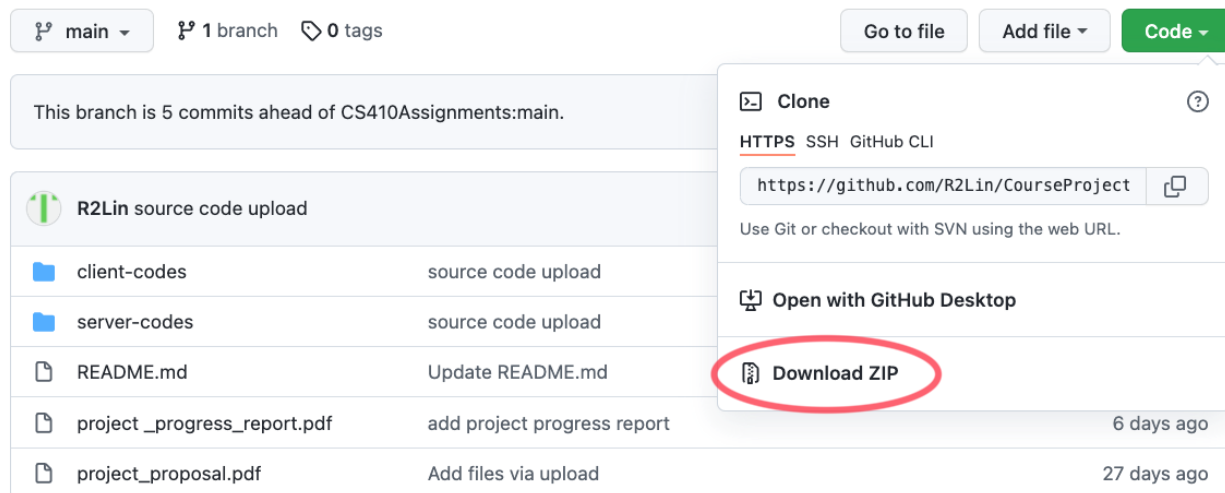


Figure 7. Download the source code.



Figure 8. Turn on "Developer mode" then click "Load unpacked" to load the Chrome extension.

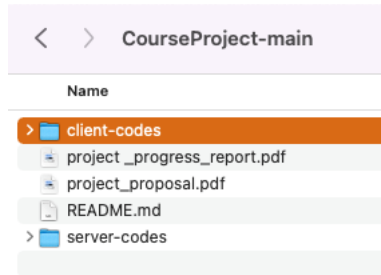


Figure 9. Choose “client-codes” subfolder in “Load unpacked” folder selection dialogue.

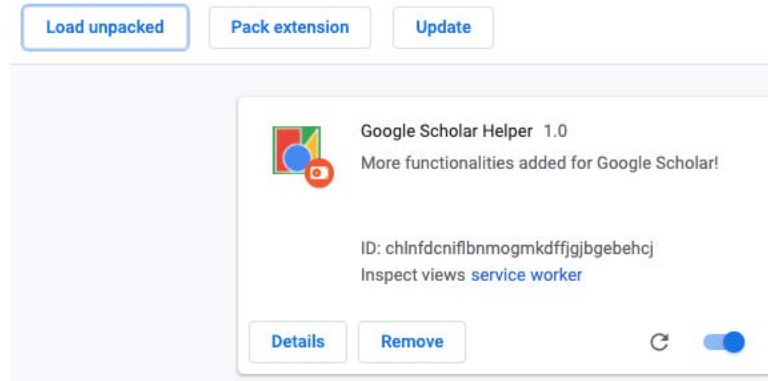


Figure 10. Successfully loaded the extension.

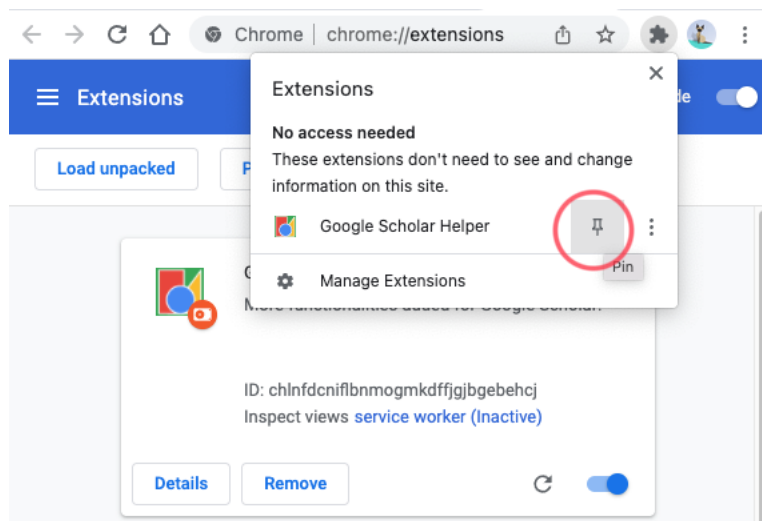


Figure 11. Pin the extension to address bar.

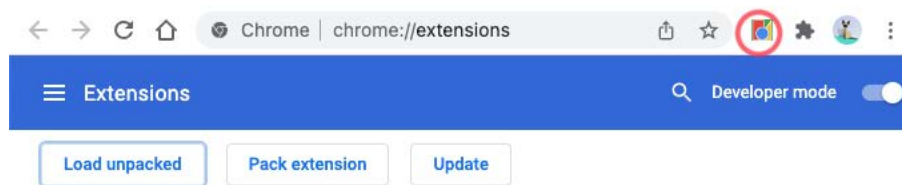


Figure 12. Pinned extension on address bar.

Server-side Installation

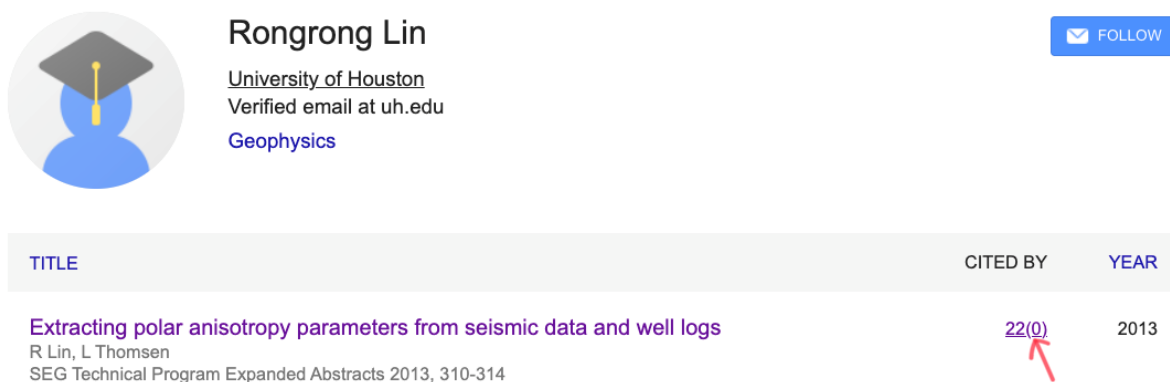
For the current setting, server-side code is preloaded on my AWS server and is listening for requests from client side across internet. So, nothing needs to be done from user side. However, if you want to install your own server, or the AWS server is down, you can follow the instructions in the Appendix for server installation.

How to run

1. Go to your Google Scholar author page, for example, my Google Scholar page at <https://scholar.google.com/citations?user=MtUhDT0AAAAJ&hl=en>
2. Click the extension icon and a popup window will show up (Figure 4).
3. Click “Go” button and the Google scholar page will be updated with the independent citation number and ranked paper lists (Figure 5). A summary table for the authors will be shown in the popup extension window (Figure 6).

Limitations of the current version

1. Google will block the IP address if user keeps scraping the webpage frequently (<https://stackoverflow.com/questions/22657548/is-it-ok-to-scrape-data-from-google-results>). You will know you are blocked if you see the independent citation count is always 0 (Figure 13) and nothing show up for the pop-up summary Gui and the recommended reading list. If you encounter such occasion, you need to manually click the citation link to unlock the block (Figure 14). Clear browsing cookie also solves the problem. To prevent Google blocking, you can try to reduce the recursion depth in the popup extension window (Figure 4 “Max Depth” field).



TITLE	CITED BY	YEAR
Extracting polar anisotropy parameters from seismic data and well logs R Lin, L Thomsen SEG Technical Program Expanded Abstracts 2013, 310-314	22(0)	2013

Figure 13. An indicator that your IP is blocked from scraping webpages.

Please show you're not a robot

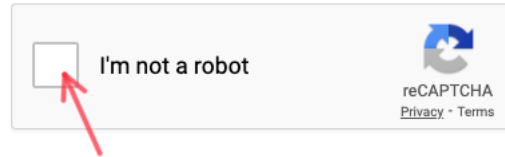


Figure 14. Manually click the citation link to unlock the block.

2. Every time you click the “Go” button, please have a refreshed Google Scholar author page, otherwise the added information from the extension will accumulate each time.
3. It could take a long time for the extracted information to load when the Google Scholar author has a lot of publications listed.
4. The extension only processes the author publications listed on the current page, it will not exhaust all the publications of the author (by click “show more” on the author page).

APPENDIX: Set up your Own Server

1. Copy subfolder “server-codes” to the server you want to use.
2. Login to your server and go to the “server-codes” directory.
3. Install Node.js from <https://nodejs.org/en/>
4. Install Miniconda Python from <https://docs.conda.io/en/latest/miniconda.html>
5. Run command “npm init -y && npm i express” from terminal in “server-codes” directory.
6. Go to “server-codes/search_ranker_test” directory.
7. Run command “conda env create -f env.yml” to install a python environment with metapy installed.
8. Run command “conda activate scholar_ranker” to activate the installed python environment
9. Go to “server-codes” directory.
10. Run command “node index.js” to start the server. Then you need to modify IP addresses in client-side codes following next few steps.
11. Find the IP address for the server you want to use. If you want to use your own computer, you can find your IP address by following instruction in this link:

<https://www.pcmag.com/how-to/how-to-find-your-ip-address>

12. Go to “client-codes/manifest.json” line 31 and change the “host_permissions” field to your server IP address (Figure 15).
13. Go to “client-codes/background.js” line 6 and change the IP address for the first argument of “fetch” function (Figure 16).
14. Reload the Google Scholar Helper extension in your Chrome (Figure 17).



Figure 15. Change IP permission for the Chrome extension in “client-codes/manifest.json”.

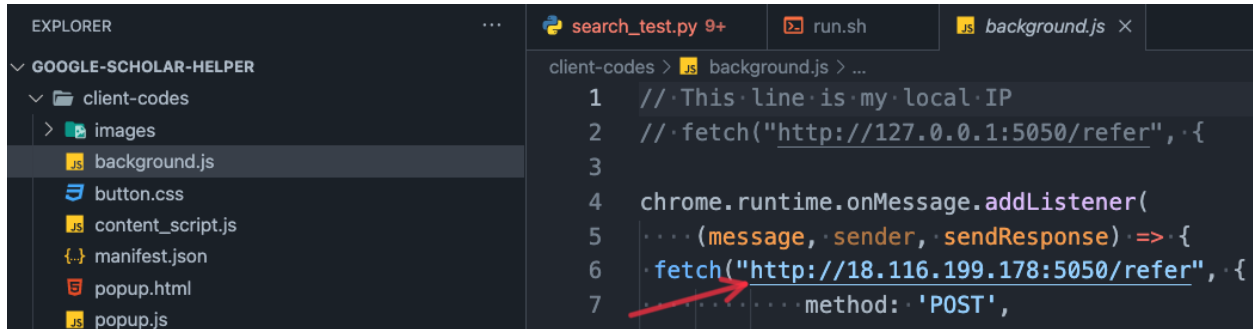


Figure 16. Change the fetching address.

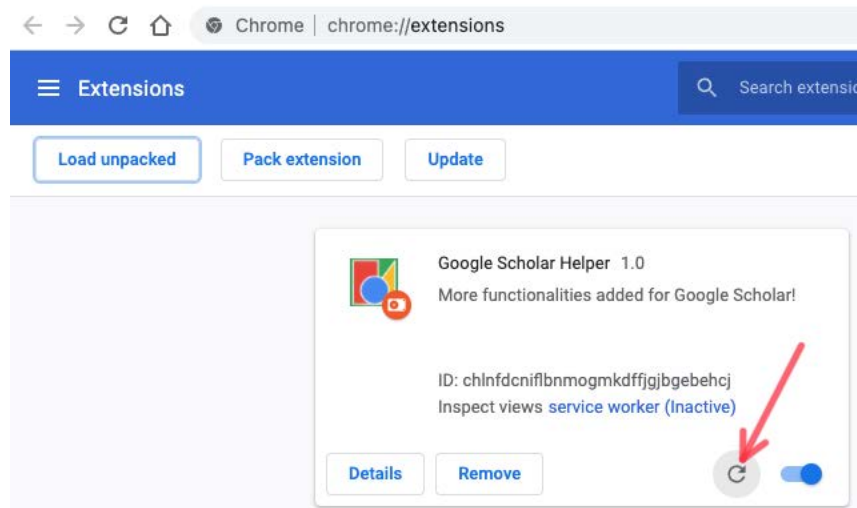


Figure 17. Click refresh button to reload the extension.