Advanced search features for the SWH archive

Abstract

Software Heritage is an ambitious research project whose goal is to collect, preserve, and share all the publicly available software with its source code and development history. The Software Heritage archive has a search feature for the metadata and the repository URLs. This metadata includes the name of the package, version, description, list of authors, et cetera. The archive search is accessible via the Web UI as well as the REST API.

The goal of my project is to make the archive search more expressive and powerful which primarily involves changes in the archive web app(swh-web), the storage service(swh-storage), and the Elasticsearch service(swh-search).

What do I want to achieve?

1. **Search Box Autocomplete**
    a. I propose to replace the 'search in metadata (instead of URL)' checkbox with a search as you type autocomplete feature for url only search.
    b. **If the user doesn't select any of the suggested URLs then we will do a metadata search.**
    c. In case the user is interested in only url based search, even in the metadata search, we can boost a result (a feature provided by elasticsearch) if the search keywords are found in the url.

2. Search Results and Interface for Advance queries
    a. **Sorting** based on:
        i. Recently updated
        ii. Recent release
        iii. Newest
        iv. Best match(score generated by Elasticsearch)
    b. **Filters** based on **(Combination allowed)**:
        i. License or Language or Mimetype (Mined by swh-indexer, but needs to be integrated)
        ii. Date range (last updated range)
        iii. Repository owner (based on URL)
        iv. Visit type (Already implemented in swh-search)

3. Simple Query Language (To be done only if time permits)
    a. Use Elasticsearch string query to build a very simple query language that can be used for now
    b. Later it can be integrated into the Archive search query language project.

Why is it important and useful for Software Heritage?

Modern-day forges and registries like Github have advanced search features that support complex but easy-to-write queries. We are building the largest library of source code which is bigger than any other forge or registry. That's why we need powerful but easy-to-use advanced search features for the archive.

Work Plan

Before May 17 (Before Announcement of Accepted students):

- Familiarize me completely with the code related to search and the Elasticsearch schema by reading the code and asking relevant questions on chat.
- Study about best practices related to Elasticsearch.
- Study the search features provided by github.com and libraries.io.

May 18 – Jun 6 (Community bonding period): 20 Days

- I will remain in constant touch with my mentors and all the community members to discuss and finalize the planned features and designs.
- With the help of my mentors, I will become absolutely clear about my future goals, the final implementations that need to be done, and the approach that I will follow.

Phase 1 (Jun 7 - Jul 16): 40 days

Origin autocomplete for the search box (Jun 7 – Jun 22): 15 days

- Make frontend changes for search box autocomplete(only for origin search) feature.
- Integrate it with Origin search endpoint which is ready to use and already uses elasticsearch if available (otherwise swh-storage)

UI for sorting and filters in archive search (Jun 22 – Jul 16): 25 days

- Add ‘sort by’ select box and the UI to apply filters in swh-web
  (I’m doing the frontend first as it won’t require much work on adding new filters later)
- Come up with a few mockups and implement the best one. (If possible, I will try to complete this during the community bonding period).
- Integrate API for visit type(already implemented) filter in the new interface

Phase 2 (Jul 17 - Aug 23): 38 days

Backend changes for filters and sorting in archive search (Jul 17 – Aug 16): 31 days

- Integrate “sort by” options in the following order :
  - Newest (to be calculated from swh-storage)
  - Recent release (to be calculated from swh-storage)

    - Recently updated (to be calculated from swh-storage)
- Integrate filters in the following order :
    - Visit type (already implemented in swh-search)
    - Repository owner (only based on URL path)
    - Date range (ex: last updated range which can be based on previously calculated value for “recently updated” based sorting)
    - License or Language or mimetype (mined by swh-indexer, but needs to be integrated)

Final Week (Aug 17 – Aug 23): 7 days

- Buffer week for unexpected delays otherwise for search query language or writing documentation.

Future aspects :

- Contributors and dependencies, if mined, can be used to empower the search.
    - Sorting results by maximum dependent repos or number of contributors
    - Every contributor’s email address is his/her unique identifier so a filter can be made to search his/her repositories across swh.
- We can have a metric to calculate the community health (maybe the one made by CHAOSS or our own metric like libraries.io’s Sourcerank) and use it in sorting.
- Integrating extrinsic metadata (ex: Stars and forks count) for sorting and filtering

Accepted revisions:

1. swh-web: Add frontend test for origin intrinsic metadata search
2. swh-indexer: Fix SingleFileMapping case sensitivity
3. swh-web: Add navbar authentication link
4. swh-scanner: Clean path in scan output
5. swh-indexer: Sync with official codemeta repo
6. swh-indexer: Add mapping for CITATION.cff
7. swh-storage: Deduplicate lists passed to *_add endpoints

In progress:

1. D5420: swh-model: cli/identify: Add support for --recursive
2. T2823: swh-journal: Write tests for swh/journal/writer/inmemory.py

About Me

I’m Kumar Shivendu from India. I am a pre-final year student pursuing a Bachelor’s degree in Computer Science and Engineering at the Indian Institute of Technology, Bhilai. I have acquired a decent knowledge of Web Development and Machine Learning through my previous internships and personal projects.

I have been contributing to open-source projects for some time and I am excited by the idea of preserving open source projects. That’s why I love contributing to Software Heritage. I am grateful for the learnings from the community’s feedback. That's why I want to keep contributing to Software Heritage even after GSoC.

Github: https://github.com/KShivendu
LinkedIn: https://www.linkedin.com/in/kshivendu
Portfolio: https://krshivendu.ml