The Popular Content Filenames Dataset: Deriving Most Likely Filenames from the Software Heritage Archive

Valentin Lorentz, Roberto Di Cosmo, Stefano Zacchiroli

The Popular Content Filenames Dataset: Deriving Most Likely Filenames from the Software Heritage Archive*

Valentin Lorentz, Roberto Di Cosmo, Stefano Zacchiroli†

July 26, 2023

Abstract

The Popular Content Filenames Dataset provides for each unique file content present in the Software Heritage Graph dataset its most popular filename. For the 2022-04-25 version, it contains over 12 billion entries and weights 413 gigabytes. This dataset allows to easily select subsets of the file contents from the Software Heritage archive based on file name patterns, facilitating reseach tasks in areas like data compression and machine learning.

1 Introduction

Software Heritage is a non profit multi-stakeholder initiative launched in 2016 by Inria, in partnership with UNESCO and a growing number of supporters, with the stated goal to collect, preserve forever, and make publicly available the entire body of software, in the preferred form for making modifications to it [1]. Currently the Software Heritage archive[1] is the largest collection of software source code and its accompanying development history, containing more than 15 billion unique source code files and 3 billion unique commits, collected from over 230 million software projects. The source of these artifacts are major collaborative development platforms such as GitHub, GitLab, and Bitbucket, as well as package repositories like PyPI, Debian, and NPM.

In the archive, these software artifacts are stored in a uniform representation that links together source code files, directories, commits, and snapshots of entire version control systems (VCS) repositories, as observed during the periodic crawls by Software Heritage.

On a regular basis, Software Heritage releases an anonymized dump of the archive content, known as the *Software Heritage Graph Dataset*, which is described in detail in [11]. That dataset is openly available on Amazon's AWS S3 platform [6] and full documentation for using it is available on the Software Heritage website [7]. Its accessibility and scale make it a unique resource that facilitates exploration into a wide range of questions related to public software development—see [10, 14, 13, 12, 9] for a few examples.

1.1 Deduplication of file contents in the Merkle DAG

The data model of the Software Heritage archive, faithfully reproduce in the Software Heritage Graph Dataset, is based on a Merkle [8] structure (specifically a Merkle direct acyclic graph, or *Merkle DAG*), which links together all software artifacts archive by Software Heritage: individual file contents, entire directories, commits, releases, VCS snapshots (see [5] for details).

A particular feature of the Merkle construction is that if a file content is present in multiple projects, with possibly different filenames, it is stored only once, as a leaf of the graph that contains

*This work was made possible by Software Heritage, the great library of source code: `https://www.softwareheritage.org`

†Valentin Lorentz, Inria, France. Email: `valentin.lorentz@inria.fr`; Roberto Di Cosmo, Inria and Université Paris Cité, France. Email: `roberto@dicosmo.org`; Stefano Zacchiroli, LTCI, Télécom Paris, Institut Polytechnique de Paris, France. Email: `stefano.zacchiroli@telecom-paris.fr`

[1] `https://archive.softwareheritage.org`

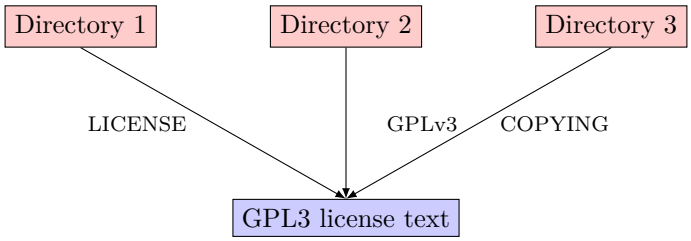

Figure 1: Fragment of the Software Heritage Merkle DAG illustrating how the same content may have multiple filenames

the rest of the information. This deduplication of contents is particularly useful to reduce storage size, and for all applications where one does not want to see the same content more than once, like in machine learning tasks on source code.

On the down side, with this data model, given a specific content, it is not possible to identify in constant time a good *representative* among all possible filenames used in the graph to refer to that particular content. Indeed files with the same content can be present in many different directories, with different filenames, but still correspond to a single file content node in the Merkle DAG. See Fig 1 for an example of this common situation.

For example, the following query can be run on Amazon Athena using the Software Heritage graph dataset 2022-12-07. Its result show that the specific version of the text of the GPL-3 license, whose SWHID is swh:1:cnt:94a9ed024d3859793618152ea559a168bbcbb5e2,[2] appeared in the Software Heritage archive at the time under 2871 distinct filenames.

```
SELECT from_utf8(name) AS name, COUNT(*) AS occurrences
FROM directory_entry
WHERE target = '94a9ed024d3859793618152ea559a168bbcbb5e2'
GROUP BY from_utf8(name)
ORDER BY occurrences DESC;
```

This is why we created the *Popular Content Filenames Dataset*. The dataset provides for each file content stored in the Software Heritage archive the filename that occurs most frequently in a directory containing it, allowing constant time access to this information for other analysis.

For the 2022-04-25 version, it contains 12033434903 entries and weights 413 gigabytes when stored in the ORC columnar format.

2 Methodology

A natural approach for defining the "most popular" filename for a given file content (blob) is to choose the filename that appears most often among the directories containing that blob. Two main methods for achieving that using the Software Heritage Graph Dataset are proposed: executing a large SQL query or leveraging the `swh-graph` [2] compressed in-memory representation of the Merkle DAG.

2.1 Method 1: SQL query

The first approach is to execute an SQL query on the dataset that counts the frequency of each filename associated with each blob and selects the most frequent filename for each blob. The following listing shows an SQL query implementing this:

[2]SWHIDs, for *Software Heritage Identifiers* are intrinsic, persistent identifiers for software source code artifacts, based on strong cryptographic checksums. See [4] for a detailed discussion of SWHIDs and `https://docs.softwareheritage.org/devel/swh-model/persistent-identifiers.html` for their technical specification.

Completed | Time in queue: 261 ms | Run time: 7 min 2.808 sec | Data scanned: 9.32 TB

Results (2,871)

Copy | Download results

Search rows

1 2 3 4 5 ... >

#	name	occurrences
1	LICENSE	11009822
2	COPYING	9581340
3	LICENSE.txt	2023944
4	COPYING3	1334976
5	COPYING.txt	1096360
6	license.txt	592145
7	GPLv3.txt	574983
8	gpl-3.0.txt	506600
9	gpl.txt	383760
10	LICENSE-GPL3.txt	338716
11	COPYING.GPLv3	336060
12	LICENSE.md	217227

Figure 2: The 2871 different filenames for the GPL-3 license, ordered by number of occurrences in the directory-entry table

```sql
WITH name_counts AS (
  SELECT target, name, COUNT(*) AS count,
    row_number() OVER (PARTITION BY target ORDER BY COUNT(*) DESC) AS rn
  FROM directory_entry
  GROUP BY target, name
)
SELECT target, name, count
FROM name_counts
WHERE rn = 1;
```

Listing 1: SQL query to select the most popular filename for each blob

2.2 Method 2: `swh-graph` compressed representation

The second approach leverages the `swh-graph` compressed in-memory representation of the graph, built on top of the WebGraph graph compression framework [3]. It allows efficient traversal of all leaf nodes and their ancestors. The algorithm can be described with the following pseudocode, that assumes to have access to the following functions:

`labeled_ancestors: integer -> list[arc]`: a function returning the adjacency list of a given node identifier (an integer), as supported by the `swh-graph` API

`filename: integer -> string`: a function resolving filename ids to strings

`swhid: integer -> string`: a function resolving node ids to SWHIDs

`length: integer -> integer`: a partial function returning the length of a content node given its id

```
create map:
    count: integer -> integer
for i from 0 to num_nodes:
    if node i is a blob:  # i.e. a lead of the graph
        reset count
        for each arc in labeled_ancestors(i):
            # arc.source == i
            if arc.target is a directory:
                increase count[arc.filename_id]
        end for
        j = integer such that for which count[j] is the maximum
        emit(swhid(i), length(i), filename(j), count[j])
end for
```

Listing 2: Pseudocode to select the most popular filename for each blob using 'swh-graph'

These methods, although differing in their implementation, follow the common principle of identifying the most frequently occurring filename associated with each blob in the dataset.

3 Dataset format and potential use cases

The dataset comes as a compressed comma separated value (CSV) file, containing the following fields:

SWHID the Software Hash identifier of the file content

length the length of the file content

filename the most popular filename, computed as described in the previous section

occurrences the number of times this filename has been seen for this content

Here are the first few lines of the dataset:

```
SWHID,length,filename,occurrences
swh:1:cnt:ba44664678a2f4b04ee41ba9624ed73793e47416,2467,drm_gem_names.h,2
swh:1:cnt:95252ee04ea25a860e7786d71f25358a7fedaf0a,6421,board_ea4357.c,3
swh:1:cnt:59d65388faf53d890d7ecd9603ac0d772a7a0e48,23899,als300.c,496
swh:1:cnt:3dfdd22b4cea2208c85a90ba1ff1bd5aa10ea521,54343,ng_ubt.c,1
swh:1:cnt:b55d6c9d816f60ab05d39211a104134d736f60f5,3156,TaskServiceDummy.java
    ,47
swh:1:cnt:e1509f701055803fcc98aa9d0f441891c30e2294,27729,ixgbe_fdir.c,3
swh:1:cnt:2512503b1ff5dbcac4a9702b5efb93b5cb76e431,30168,
    toshiba_rbtx4927_setup.c,1
swh:1:cnt:4865b0342f3f005f2cbf3615116feb121e7362da,117214,msm_otg.c,1
swh:1:cnt:b5a9b7aed1360b1532156f620ab14d59d4b8018f,1024,lkHeartbeat,1
```

3.1 Bias

Because of the deduplication within the Merkle structure, this definition of likeliness gives more weight to projects which made many changes to the directory containing a given file; and only count any given directory only once, even if it appears verbatim in multiple projects.

3.2 Potential Use Cases

With the information contained in this dataset, it is easy to obtain, for example, a list of file contents written in a given programming language, based on the file extension, and/or a set of such file contents totalling a given size, e.g. 200 GiB. This can be very useful for specific experiments, like large scale data compression experiments or machine learning tasks.

4 Data Accessibility

The dataset is available as part of the Software Heritage Graph Dataset hosted on Amazon AWS, in the framework of the Amazon Open Dataset program [6]. It is provided in the columnar storage ORC format, that is ready to use, for example, with Amazon Athena.

Here is how you can access the dataset associated to the 2022-04-25 version of the graph dataset:

```
aws s3 ls --no-sign-request s3://softwareheritage/derived_datasets/2022-04-25/
    popular_contents/
2023-04-03 11:20:11 41323165231 3d78e6c2-c295-434b-a176-90190d17b47a.orc
2023-04-03 11:20:13 41293499356 59d4bb9d-f282-4c8c-b34c-282044a4ebbc.orc
2023-04-03 11:20:13 41287775036 6a2b423e-9fef-47ec-9c1a-017e13a76ec3.orc
2023-04-03 11:20:12 41329652285 73f5494e-79af-4d99-b99e-decdfdb7e516.orc
2023-04-03 11:20:12 41291466465 ab4f5064-5f19-4877-b383-c5e3639c6f18.orc
2023-04-03 11:20:13 41319605899 b07959f3-0329-4c0c-9c6d-706012843957.orc
2023-04-03 11:20:13 41340910158 be3c5f73-c60a-4d4a-b693-647415b77e39.orc
2023-04-03 11:20:12 41293437394 daeca5cd-797f-4455-9607-891859641448.orc
2023-04-03 11:20:12 41322316453 f05df65a-0688-4729-a5f0-2d7c0b792905.orc
2023-04-03 11:20:12 41302796505 fb7b4ad3-35c0-4649-a417-34a00b2cecb7.orc
```

The total size is 413 gigabytes.

5 Conclusion

The "Popular Content Filenames" dataset provides a valuable enhancement to the accessibility and utility of the Software Heritage Graph, offering a precomputed index of all the available unique file contents, with their most popular file name.

6 Acknowledgment

We are grateful to the AWS Open Dataset program for supporting the distribution of this dataset.

References

Articles

[1] Jean-François Abramatic, Roberto Di Cosmo, and Stefano Zacchiroli. "Building the Universal Archive of Source Code". In: *Communications of the ACM* 61.10 (Sept. 2018), pp. 29–31. ISSN: 0001-0782. DOI: 10.1145/3183558. URL: http://doi.acm.org/10.1145/3183558.

[2] Paolo Boldi, Antoine Pietri, Sebastiano Vigna, and Stefano Zacchiroli. "Ultra-Large-Scale Repository Analysis via Graph Compression". In: *SANER 2020: The 27th IEEE International Conference on Software Analysis, Evolution and Reengineering*. IEEE, 2020.

[3] Paolo Boldi and Sebastiano Vigna. "The webgraph framework I: compression techniques". In: *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*. Ed. by Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills. ACM, 2004, pp. 595–602. ISBN: 1-58113-844-X. DOI: 10.1145/988672.988752. URL: https://doi.org/10.1145/988672.988752.

[4] Roberto Di Cosmo, Morane Gruenpeter, and Stefano Zacchiroli. "Identifiers for Digital Objects: the Case of Software Source Code Preservation". In: *Proceedings of the 15th International Conference on Digital Preservation, iPRES 2018, Boston, USA*. Sept. 2018. DOI: 10.17605/OSF.IO/KDE56. URL: https://hal.archives-ouvertes.fr/hal-01865790.

[5] Roberto Di Cosmo and Stefano Zacchiroli. "Software Heritage: Why and How to Preserve Software Source Code". In: *Proceedings of the 14th International Conference on Digital Preservation, iPRES 2017*. Sept. 2017. URL: https://hal.archives-ouvertes.fr/hal-01590958/.

[6] Software Heritage. *Software Heritage Graph Dataset*. Ed. by Registry of Open Data on AWS. 2023. URL: https://registry.opendata.aws/software-heritage/ (visited on 06/03/2023).

[7] Software Heritage. *Software Heritage Graph Dataset Documentation*. 2023. URL: https://docs.softwareheritage.org/devel/swh-dataset/graph/dataset.html (visited on 06/03/2023).

[8] Ralph C. Merkle. "A Digital Signature Based on a Conventional Encryption Function". In: *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*. Ed. by Carl Pomerance. Vol. 293. Lecture Notes in Computer Science. Springer, 1987, pp. 369–378. ISBN: 3-540-18796-0. DOI: 10.1007/3-540-48184-2_32. URL: https://doi.org/10.1007/3-540-48184-2_32.

[9] Antoine Pietri, Guillaume Rousseau, and Stefano Zacchiroli. "Forking Without Clicking: on How to Identify Software Repository Forks". In: *MSR 2020: The 17th International Conference on Mining Software Repositories*. IEEE, 2020, pp. 277–287. DOI: 10.1145/3379597.3387450.

[10] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. "The Software Heritage Graph Dataset: Large-scale Analysis of Public Software Development History". In: *MSR 2020: The 17th International Conference on Mining Software Repositories*. to appear. IEEE, 2020.

[11] Antoine Pietri, Diomidis Spinellis, and Stefano Zacchiroli. "The Software Heritage graph dataset: public software development under one roof". In: *Proceedings of the 16th International Conference on Mining Software Repositories, MSR 2019, 26-27 May 2019, Montreal, Canada*. Ed. by Margaret-Anne D. Storey, Bram Adams, and Sonia Haiduc. IEEE, 2019, pp. 138–142. ISBN: 978-1-7281-3412-3. URL: https://dl.acm.org/citation.cfm?id=3341907.

[12] Davide Rossi and Stefano Zacchiroli. "Geographic Diversity in Public Code Contributions: An Exploratory Large-Scale Study Over 50 Years". In: *The 2022 Mining Software Repositories Conference (MSR 2022)*. ACM, 2022, pp. 80–85. DOI: 10.1145/3524842.3528471.

[13] Stefano Zacchiroli. "A Large-scale Dataset of (Open Source) License Text Variants". In: *The 2022 Mining Software Repositories Conference (MSR 2022)*. ACM, 2022, pp. 757–761. DOI: 10.1145/3524842.3528491.

[14] Stefano Zacchiroli. "Gender Differences in Public Code Contributions: a 50-year Perspective". In: *IEEE Software* 38.2 (2021), pp. 45–50. ISSN: 0740-7459. DOI: 10.1109/MS.2020.3038765.