Deliverable D5.2
Use cases evaluation report

Making Software FAIR: A machine-assisted workflow for the research software lifecycle

Project acronym: SoFAIR

Grant Agreement Number: CHIST-ERA-22-ORD-08

Deliverable information	
Deliverable number and name	D5.2 Use cases evaluation report
Due date	M24
Actual delivery date	
Work Package	WP5
Lead Partner for deliverable	Brno University of Technology
Authors	Martin Docekal, Luca Foppiano, Patryk Hubar, Alain Monteil, Tomasz Umerle
Reviewers	David Pride, Laurent Romary
Approved by	
Dissemination level	Public
Version	0.9

Document revision history			
Issue Date	Version	Author	Comments
22/04/2025	0.1	Martin Docekal	Structure draft
25/11/2025	0.9	Martin Docekal, Luca Foppiano, Patryk Hubar, Alain Monteil, Tomasz Umerle	Review ready
	Final		

Table of Contents

Executive summary

This report provides a description of use cases developed within Work Package 5 (WP5). Two demonstrators and one case study is presented.

The first demonstrator (Linking research studies to software in life sciences for Europe PMC) aims to create a pipeline for integrating TEI-annotated content, generated by SoFAIR machine learning models, into the existing Europe PMC annotations infrastructure.

The second demonstrator (Validating extracted software mentions within an institutional repository) focuses on creating reliable and scalable end-to-end system for identifying, processing, and validating software mentions in publications deposited in the HAL institutional repository.

Finally this document describes the case study in the digital humanities investigating the potential of automated software-mention detection for analyzing digital transformation processes in the humanities by examining software-usage.

1. Introduction

This document focuses on the **evaluation and deployment** of the tools and workflows developed by the SoFAIR project. The primary **objectives** were to assess the effectiveness and scalability of the developed pipelines for extracting and processing software mentions from scholarly literature in real-world contexts, and to provide feedback for improving both the tools and the overall **SoFAIR workflow**.

Two demonstrators and one case study are presented. The demonstrators test the outputs of the core machine learning and annotation tools (like **Softcite**, **Grobid**, and **candidate filter**) developed in earlier stages. One demonstrator additionally serves as a proof of concept for the end-to-end workflow, involving authors as validators.

The case study examines digital transformation processes in the humanities, utilizing tools developed during the SoFAIR project.

All case studies provide a deep **description**, followed by a **quantitative evaluation** of the effort, and conclude with **recommendations** and a description of potential **limitations**.

2. Use Cases

2.1. Demonstrator 1: Linking research studies to software in life sciences for Europe PMC

Europe PMC is a trusted world leading biomedical literature resource with a unique value offer; combining the functionality of a search engine with full text repository, and enabling context-aware information retrieval. Users can traverse abstracts and full text from PubMed, PMC, preprint servers, and other sources on one comprehensive platform. It connects publications and data, extracting evidence to uncover meaningful relationships between concepts. Metadata is enriched with persistent identifiers (PIDs), uses community standards, and is available in open formats. Data can be accessed through open APIs and data dumps for integrations, application development, artificial intelligence (AI) model training, and meta analysis studies.

Europe PMC contains a corpus of approximately 70,000 full text preprints, which have been converted from PDF to XML as a result of funding from the Europe PMC funders. This corpus is composed of two subsets - the COVID-19 corpus of approximately 50,000 preprints and the remainder are identified as those funded by Europe PMC funders, which is an ongoing funded project.

The Europe PMC platform provides an annotations submission platform which enables outputs from other projects to deposit grounded links of associated outputs to the literature.

As a demonstrator use case the full text preprint corpus is an ideal use case to enhance the utility and discoverability of preprints alongside associated open science outputs in the form of research software.

Problem statement

Europe PMC seeks to evaluate a workflow for integrating TEI-annotated content, obtained from machine learning models developed as part of the SoFAIR project, into its existing annotations infrastructure.
To support this evaluation, a one-off proof of concept is required to transform the TEI annotations into the Europe PMC format and deliver them to the Europe PMC annotations platform. Once delivered, Europe PMC must be able to ingest these annotations and expose them through its public API.

The original project proposal also included the use of the Scilite web application. However, this component was eventually abandoned due to technical constraints and because Europe PMC no longer plans to support Scilite in the future. Further details can be found in the *Limitations* section.

Methodology

Using existing tools developed during the SoFAIR project, we developed a pipeline for converting TEI files into a format compatible with Europe PMC. For our proof of concept, we used the corpus described in the introductory section of this use case.

Our work was not limited to the conversion process itself. We conducted a detailed analysis of the constraints encountered and formulated recommendations for future development. We concluded with a quantitative evaluation of the software mentions that were successfully transformed into the Europe PMC format.

Pipeline

This section outlines the complete pipeline developed for this use case. The figure below provides a visual summary of all processing stages, which are explained in detail in the subsequent text. The full implementation of the pipeline is available in the project's public GitHub repository (https://github.com/SoFairOA/UseCaseEuropePMC).

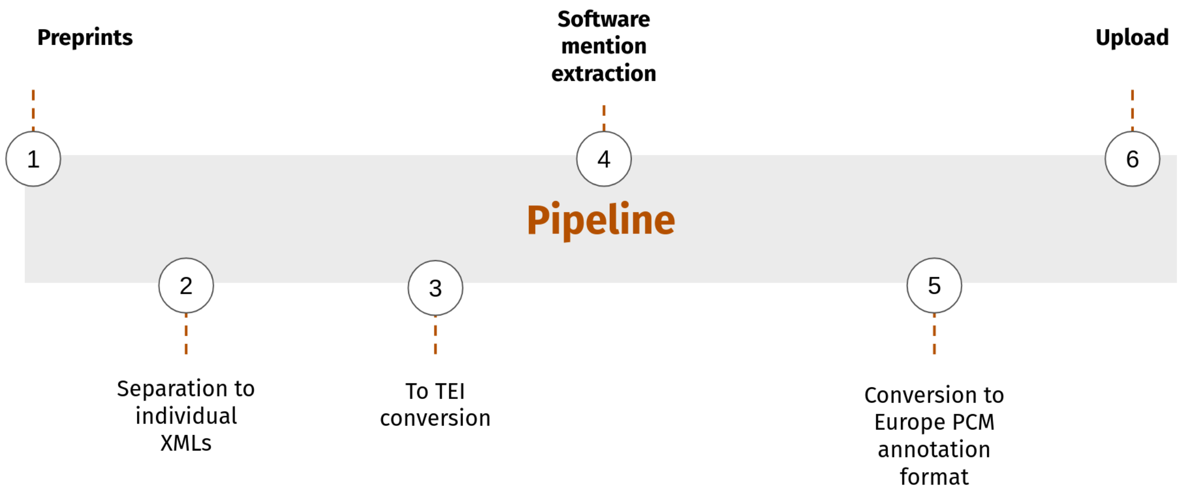

Preprints
Software mention extraction
Upload
1
4
6
Pipeline
2
3
5
Separation to individual XMLs
To TEI conversion
Conversion to Europe PCM annotation format

Figure 1: Overview of the pipeline for Europe PMC

Preprints

To demonstrate the pipeline, we used the Europe PMC preprint corpus introduced earlier in this use case. In our GitHub repository, we provide a download script with URLs that exactly identify our data sources. The script is available at https://github.com/SoFairOA/UseCaseEuropePMC/blob/main/download_and_convert/download.sh.

As noted previously, the corpus primarily consists of COVID-19–related publications. To provide more detailed information, we include the following pie chart, which offers an overview of the specific sources represented in the dataset.

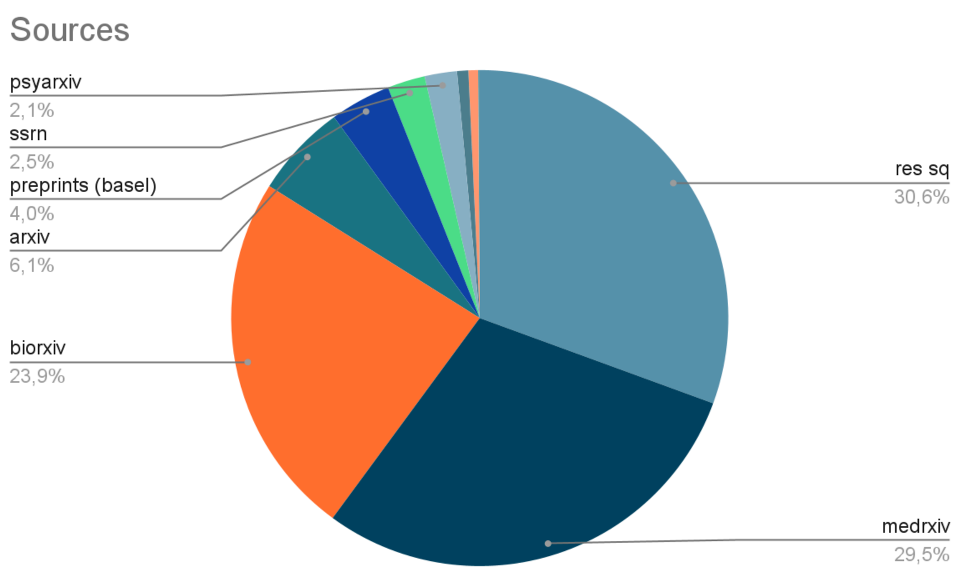


Figure 2: Europe PMC corpus sources

Our pipeline processes 67,809 papers as input. All of these papers are provided in XML format, which we convert into TEI— the format required by the Softcite tool used for extracting software mentions.

The first step is to split the source XML files into individual documents, as the original data contains multiple papers within a single XML file. This task is handled by a Python script available at:

https://github.com/SoFairOA/UseCaseEuropePMC/blob/main/download_and_convert/separate.py

After separating the documents, we run the Pub2TEI converter (https://github.com/kermitt2/Pub2TEI) to generate the corresponding TEI files.

Software Mention Extraction

In this step, we process the TEI documents produced previously using Softcite with the model trained during the SoFAIR project. To do this, we use the Softcite Python client (https://github.com/softcite/software_mentions_client), which sends requests to the Softcite server packaged in the Docker image: **lfoppiano/software-mentions:0.8.2-sofair_2**.

An example shell script demonstrating how to run the extraction workflow is provided in our GitHub repository:
https://github.com/SoFairOA/UseCaseEuropePMC/blob/main/extract_mentions.sh

The extraction process resulted in **1,099,332 software mentions** (counting all mentions, not unique software names). Our analysis shows that **52,684 publications (~78%)** contain at least one software mention. Additional details are provided in the *Results* section.

Conversion to Europe PMC format

We need to convert the extracted mentions because Softcite does not provide results in a format compatible with the Europe PMC annotation specification, as described in the official documentation (https://europepmc.org/AnnotationsSubmission#data-format).
For this use case, we utilize the following fields from the Europe PMC annotation format:

- **src:** Source of the article (for our case PPR as we use preprints corpora)
- **id:** Identifier of the article in the context of the src field provided
- **provider:** Name of the provider (SoFAIR)
- **anns:** List of annotations
    - **type:** Annotation type (we use software_mentions)
    - **exact:** Text of the tagged entity (we use raw form of software mention)
    - **prefix:** Portion of the sentence that appears before the software mention
    - **postfix:** Portion of the sentence that appears after the software mention
    - **tags:** List of the entities tagged by this annotation (in our case it is list with one element)
        - **name:** Name of the tagged entity (we use again the raw form of software mention)
        - **uri:** URI to the ID or the Accession number the entity is linked to (we use the extracted URL of the associated software mention)

The **uri** field, which is mandatory, poses the greatest limitation for our use case, as not every software mention has an associated URL or other identifier. Consequently, we only include mentions that have a corresponding URL. Further details and quantitative evaluation are provided in the *Results* section.

Results

In this section, we present a quantitative evaluation of our use case.

From the Europe PMC corpus, we extracted **1,099,332 software mentions** (counting all mentions, not unique software names). These mentions originate from **52,684 publications (~78%)**, while the remaining publications do not contain any software mentions.

During the conversion process, we identified a key limitation: the mandatory uri field. This requirement is restrictive because Softcite cannot provide a URI/URL for every software mention.

We also assessed the quality of the extracted URLs by verifying whether each one is a valid HTTP/HTTPS link.

Our findings are summarized in the following table:

missing url ok	invalid url ok	number of mentions	%	number of documents with at least one mention	%
TRUE	TRUE	1099332	100.00%	52684	77.69%
TRUE	FALSE	1094978	99.60%	52648	77.64%
FALSE	TRUE	16626	1.51%	2384	3.52%
FALSE	**FALSE**	**12272**	**1.12%**	**1694**	**2.50%**

Table 1: Statistics of extracted software mentions

The table shows that there are 12,272 suitable software mentions (with associated valid URLs) in the Europe PMC format. These mentions appear in 1,694 documents, which represents only 2.5% of the total corpus.

To address this limitation, we conducted an experiment and developed a tool capable of extracting URLs from the immediate context surrounding a software mention. The tool links a URL to a software mention only when the URL contains a surface form of that mention as a case-insensitive substring.

We manually evaluated the precision of the URL extraction tool using a random sample of 50 software mentions that the tool had enriched with URLs. We found that 49 of the URLs were correct. Based on this high precision, we decided to incorporate the URL extraction tool in the post-processing step, as it substantially increases the number of software mentions suitable for Europe PMC, as shown in the following table.

missing url ok	invalid url ok	number of mentions	%	number of documents with at least one mention	%
TRUE	TRUE	1099332	100.00%	52684	77.69%
TRUE	FALSE	1094944	99.60%	52648	77.64%
FALSE	TRUE	45521	4.14%	10247	15.11%
FALSE	**FALSE**	**37014**	**3.37%**	**8616**	**12.71%**

Table 2: Statistics of extracted software mentions

Filter Experiment

Table 1 demonstrates that a high proportion of publications in the corpus contains at least one software mention.

We conducted an experiment using our fast filtering tool (available at: https://github.com/SoFairOA/filter). This filter is designed to quickly identify and retain candidate documents that are highly probable to contain software mentions, thereby filtering out those that likely do not.

The filter is highly suitable for this experiment, having achieved a high **recall (91%)** and **precision (86%)** on testing data.

Filter Performance Verification

To verify that the number of publications passing our filter is consistent with the number of mentions extracted by Softcite on this same corpus, we first applied our filter to the existing corpus. It selected **51,236 candidate documents (75.55%)**. This proportion is close to the number of documents (**77.69%**) identified by Softcite, confirming the filter's utility as an effective preprocessing step.

Comparison to Case Study in the Digital Humanities

Comparing results to results from the **case study in the digital humanities**, we observe that the preprint COVID corpus contains a significantly greater proportion of documents with software mentions (75.55%) than in range for DH (30–60%) and even greater than TLL (10-15%).

Limitations and Future Work

The original intention was to add the mined software citations directly to the SciLite. However, several constraints made this challenging.

First, the SciLite panel only supports links resolved through Identifiers.org, while our extracted mentions currently provide only general URLs. Future work should focus on disambiguation, enabling the use of supported identifiers.

Second, SciLite requires the exact in-text location of each annotation within the article. Because the article undergoes multiple conversion steps, obtaining these precise positions is cumbersome. Although implementing this would be technically possible, an alternative approach, highlighting the mined software directly on the main article page, was proposed. This solution is expected to have greater impact and is technically less complex.

Conclusion

We developed a pipeline to convert Softcite annotations derived from Europe PMC data into the Europe PMC format. We quantitatively evaluated the approach and documented its limitations. In addition, we analysed documents containing software mentions (focusing primarily on the COVID-19 publication corpus) and compared the results with those from **case study in the digital humanities**. Additionally we verified the usability of the candidate documents filter.

We recommend that infrastructure providers remove restrictions that make the disambiguation of software mentions mandatory through the use of specific identifiers. We believe that users can still benefit from these annotations even when such disambiguation is not enforced.

2.2. Demonstrator 2: Validating extracted software mentions within an institutional repository (INRIA)

Abbreviations

HAL: Hyper Articles en Ligne
CCSD: Centre pour la Communication Scientifique Directe, the organisation that develop and maintain HAL
ACABA:
SWH: Software Heritage
SWID: Software Heritage Identifier

SoftCite vs Software mentions: SoftCite is the project built by James Howison and Patrice Lopez to extract software mentions from scientific articles, while Software mentions is the piece of software that extracts such information from a scientific paper. Those two terms are often used interchangeably. In this document we always refer to the software used for extracting mentions from a document.

Introduction

HAL is both a software platform and a unique multidisciplinary open repository. It hosts more than 150 institutional archives from universities and research organizations, and includes dedicated spaces such as **HAL-SHS** for the Human and Social Sciences, **HAL Theses** for doctoral dissertations, and **MediHAL** for visual and audio materials.

Designed for researchers and their institutions, HAL covers all scientific disciplines and accepts a wide range of research outputs — including peer-reviewed articles, conference papers, theses, preprints, and software.

Initiated in 2001 by physicist **Franck Laloë**, HAL is developed by the **Center for Direct Scientific Communication (CCSD)**, a support and research unit operating under the supervision of three public research organizations: the **National Center for Scientific Research (CNRS)**, **Inria**, and **INRAE**.

HAL is supported by France's **national Plans for Open Science** and forms part of **HAL+**, a certified research infrastructure recognized by the Ministry of Higher Education and Research. It plays a key role in the country's **open access institutional policies** (see also HAL's position in the French Open Science Monitor).

HAL is **CoreTrustSeal certified** and listed in major international registries, including:

- the *Directory of Open Access Repositories (OpenDOAR)*,
- the *Directory of Open Access Preprint Repositories (DOAPR)*, and
- the *Registry of Research Data Repositories (re3data)*.

For more information, see the political declaration *“HAL, an open archive built in common to share and disseminate scientific knowledge”* (June 2022).

The repository also serves researchers affiliated with foreign academic institutions, whether public or private.

Workflow

Introduction

The workflow is implemented using the shared, large-scale infrastructure Grid5000[1]. Grid5000 is a large-scale testbed for research in computer science, focusing on parallel, distributed, cloud, HPC, Big Data, and AI computing.

It offers 15,000 cores and 800 compute nodes with a variety of hardware options, including GPUs, SSDs, NVMe storage, and high-speed networks. Researchers can deploy custom software environments and isolate experiments at the network level using bare-metal deployment. The platform provides monitoring of network activity and power consumption to support experiment analysis. Designed to encourage open and reproducible research, Grid5000 tracks all software and hardware changes and is supported by a community of more than 500 users and a dedicated technical team.

Grid5000 is supported by Inria and other organisations including CNRS, RENATER and several Universities.

Pipeline

The pipeline is composed by several stages (Figure 3):

- **HAL synchronization**: nightly scheduled job that look up all the new publications in HAL and download the related PDF documents
- **Document processing**: new PDF documents are processed by Grobid and SoftCite using two asynchronous pipelines
- **Load and notify**: the software mentions extracted are loaded in the software mention API (or COAR Notify inbox, in Figure 3) application which triggers COAR notification toward HAL and SWH.
- **Validation from HAL**: The software mentions are visualised in the HAL interface, and the authors are asked to validate them.
- **Post-validation from HAL**: After the data is validated, the software is registered into the HAL database and a COAR notification is sent back to the software mention API application to be registered.

All these stages are running asynchronously, their dependency is only data-related, meaning that are scheduled at a certain time on the Grid5000 infrastructure but each of them may run when the requested resources are available.

[1] https://www.grid5000.fr

For example the Grobid processing uses 10 CPU nodes, while Softcite requires GPU nodes, so the waiting time for GPU may be larger.

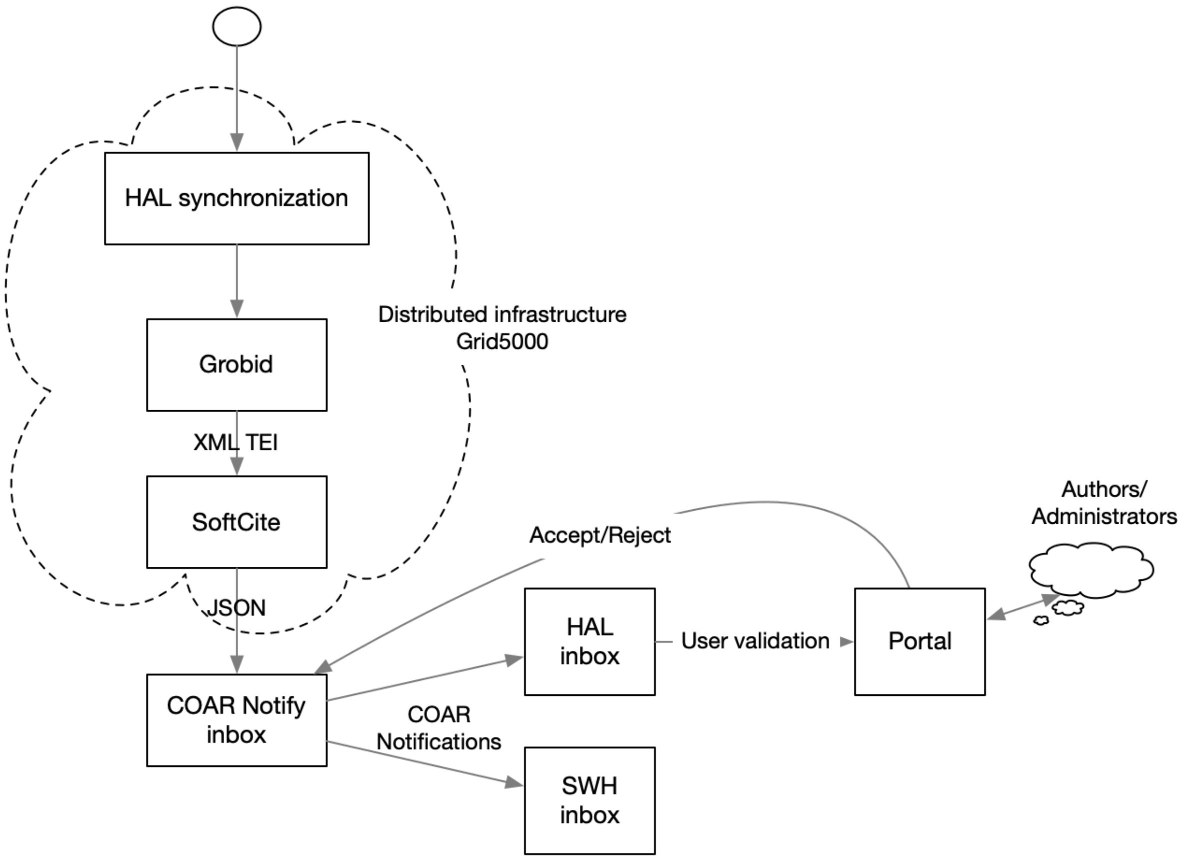


Figure 3: Overview of the pipeline

HAL Synchronization

The synchronization of the HAL copy is automated on the Grid5000 platform to ensure a daily update of a local mirror of HAL records and associated files.
Each day, a script queries the HAL API to retrieve the list of identifiers (HAL identifier) indexed during the past 24 hours. These data are compared with those from the previous day to identify deleted records (halIDs that have disappeared), which are then removed along with their corresponding files from local directories.
At the same time, new and modified records are downloaded in XML-TEI format, together with their associated PDF files, except when the files are under “embargo” (not yet publicly available). Files under embargo are tracked in a dedicated list until their release date, at which point they are automatically downloaded. This two-part process ensures that the local HAL copy remains consistently up to date, reliable, and complete.

Document processing

The document processing pipeline is described in detail in Figure 4. The pipeline can be divided into three separate parts:

- Grobid processing
- Software mentions processing
- Load and notification

Both Grobid processing and SoftCite processing are structured in the same way, the differences are the type of the input and output files, and the resources requested. Everything else is implemented in the same way.
Using the same structure it is possible to add easily any further processing, e.g. Dataset extraction, quantities extractions, etc..

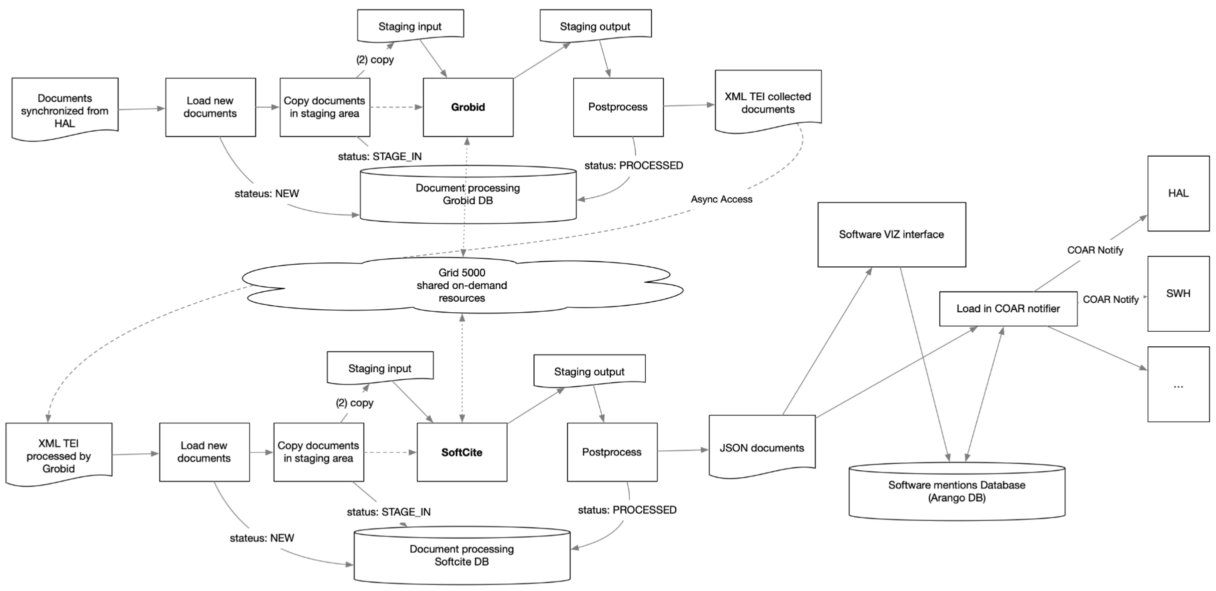


Figure 4: Document processing and notification

The Grobid and Softcite processing make use of a “Staging area”, a storage where the files are copied (to input) and retrieved (from output) after processing to avoid working directly on the original data. Any error or disruption to the staging area is reset at each run.

The **first step is collecting the list of files** that are provided in the entry directory (for Grobid will be the output of the HAL Synchronization, for SoftCite will be the output of the Grobid processing).

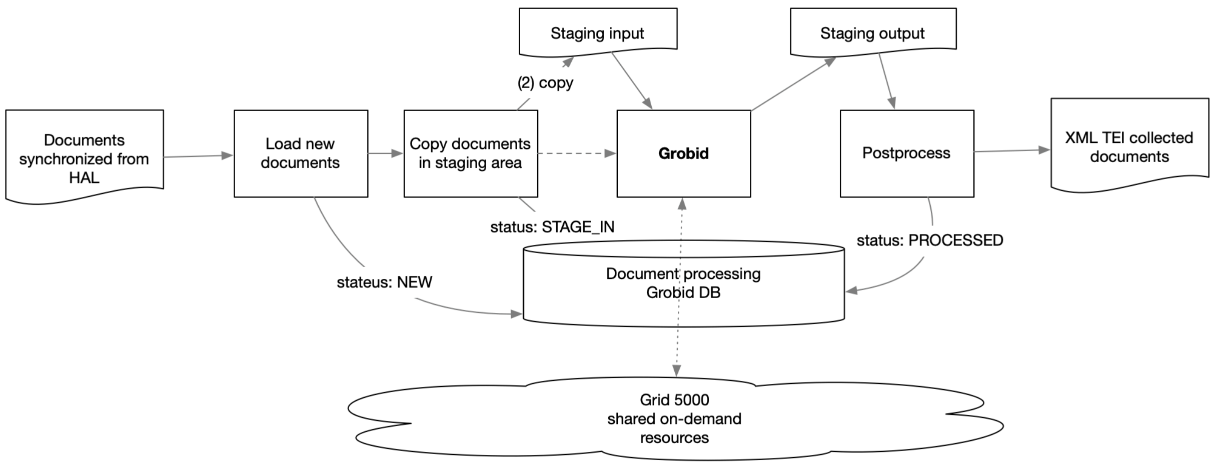

Staging input
Staging output
(2) copy
Documents synchronized from HAL
Load new documents
Copy documents in staging area
Grobid
Postprocess
XML TEI collected documents
status: STAGE_IN
status: PROCESSED
stateus: NEW
Document processing Grobid DB
Grid 5000 shared on-demand resources

Figure 5: Document processing workflow for Grobid and SoftCite (with minimal variations)

The list is then loaded into a local database which collects only the paths that are not already present (new files). The database is used to keep track of the location and status of each document.

The statuses explain the different step the document is within the processing:

- NEW: new documents
- STAGE_IN: the document was added to the input staging area
- PROCESSED: the document output was collected from the output staging area
- ERROR: the document was not processed correctly (the error cause will also be collected, depending on the processor, see below)
- NOTIFIED_OK: the COAR notify was successful (only valid for the software mentions database)
- NOTIFIED_FAIL: the COAR notify was unsuccessful (only valid for the software mentions database)
- CLEANUP: the PDF was removed (only valid for the Grobid database)

The **second step is to select, chunk and allocate a certain number of documents** to the staging area, the number of chunks depends on the number of nodes that are used for processing. This number is currently fixed and we plan to make it dynamic based on the number of documents that needs to be processed. Once the documents are allocated their status changes from “NEW” to “STAGE_IN”.

Furthermore the **processing** is launched, both Softcite and Grobid schedule N nodes and allocate each node to a chunk of the staging area, for example we managed to run 10000 documents to 10 nodes processing 100000 PDF documents in about 8 hours. This modulation of resources reduces dramatically the carbon footprint and the costs using dedicated infrastructures.

Finally, after processing, the data is postprocessed, in particular three control are made:

- we verify that the corresponding output files are created
- we collect errors messages (e.g. Grobid produces text files containing the errors) and store them in the database
- we copy the files outside of the staging area

If all these three steps are successful, we update the database changing the status from STAGE_IN to PROCESSED, otherwise to ERROR.

NOTE: documents that lead to errors are not reprocessed automatically, however we have in place a manual step that “recycles” errored documents. This is currently manual because we want to be in control when to trigger, for example after an update of Grobid or Softcite.

We produce regular statistics to monitor the processes, note that Grobid and Softcite provide different error information, so the statistics are slightly different. For example Grobid provide ore information about the type of error which is often a characteristic of the PDF (e.g. when the pdf is too big “TOO_MANY_BLOCKS”, or does not contains any text “NO_BLOCK”[2]):

```
{
  "total": 260680,
  "status_counts": {
    "cleanup": 248857,
    "error": 3542,
    "new": 88,
    "processed": 548,
    "staged_in": 7645
  },
  "missing_files": 0,
  "invalid_entries": 0,
  "error_analysis": {
    "total_errors": 0,
    "error_code_counts": {
      "ANY": 1834,
      "BAD_INPUT_DATA": 349,
      "NO_BLOCKS": 1110,
      "TIMEOUT": 28,
      "TOO_MANY_TOKENS": 185,
      "UNKNOWN": 36
    }
  }
}
```

Grobid processor statistics

```
{
  "total": 249405,
  "status_counts": {
    "error": 5,
    "notified_FAIL": 63,
    "notified_OK": 174,
    "processed": 249163
  },
  "missing_files": 0,
  "invalid_entries": 0,
  "error_analysis": {
    "total_errors": 5,
    "error_code_counts": {
      "ANY": 5
    }
  }
}
```

Software mentions statistics

Figure 6: Example of collected statistics in the local databases for Software mentions and Grobid processing

[2] See https://grobid.readthedocs.io/en/latest/Grobid-service/#errors-handling

Load and Notification

In this stage, we collect all the documents with status PROCESSED and we send them to the COAR notify inbox. This is a new web application that collects the documents and software mentions in a graph database (ArangoDB) and sends the notification to HAL (Figure 7). Within the same work package, we have also integrated our pipeline to the COAR notification of Software Heritage (SWH) (Figure 8). Software Heritage represents a strategic partner to monitor software because of the vast variety of information that they have in storage.
No notifications are sent when the loaded document does not contain any software mentions. Beside the mandatory information described by the COAR standard (inbox, service, etc.) our notifications provide the software name, and the contexts, which represents all the snippets from the document that contains the mention.

```
{
    "@context": [
        "https://www.w3.org/ns/activitystreams",
        "https://purl.org/coar/notify"
    ],
    "id": "urn:uuid:c53603b5-1803-494d-965d-956ec87d41e2",
    "type": [
        "Offer",
        "coar-notify:ReviewAction"
    ],
    "actor": {
        "id": "https://datalake.inria.fr",
        "type": "Service",
        "name": "Inria DataLake"
    },
    "origin": {
        "id": "https://datalake.inria.fr",
        "type": "Service",
        "inbox": "https://prod-datadcis.inria.fr/coar/inbox"
    },
    "target": {
        "id": "https://inria.hal.science",
        "type": "Service",
        "inbox": "https://inbox-preprod.archives-ouvertes.fr/"
    },
    "object": {
        "id": "hal-04971161v1",
        "ietf:cite-as": null,
        "sorg:citation": {
            "@context": "https://doi.org/10.5063/schema/codemeta-2.0",
            "type": "SoftwareSourceCode",
            "name": "ClipCap",
            "codeRepository": null,
            "referencePublication": null
        },
        "mentionType": "software",
        "mentionContext": ["We used ClipCap to make the…", "The ClipCap software is
available on Github"]
    }
}
```

Figure 7: Example of the body of the notification sent to HAL

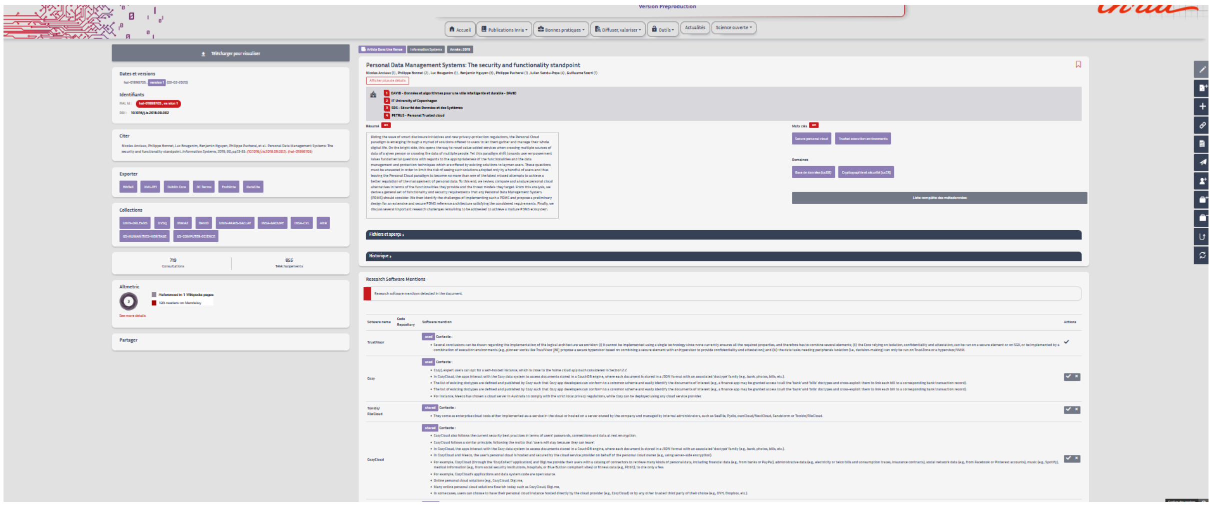
```
{
  "@context": [
    "https://www.w3.org/ns/activitystreams",
    "https://purl.org/coar/notify"
  ],
  "actor": {
    "id": "https://datalake.inria.fr",
    "type": "Service",
    "name": "Inria DataLake"
  },
  "context": {
    "id": "hal-01923108",
    "sorg:name": null,
    "sorg:author": {
      "@type": "Person",
      "givenName": null,
      "email": null
    },
    "ietf:cite-as": "https://doi.org/XXX/YYY",
    "ietf:item": {
      "id": "hal-01923108",
      "mediaType": "application/pdf",
      "type": [
        "Object",
        "sorg:ScholarlyArticle"
      ]
    },
    "type": [
      "Page",
      "sorg:AboutPage"
    ]
  },
  "id": "urn:uuid:c35982da-7021-4985-a31b-895980691ec6",
  "object": {
    "as:subject": "hal-01923108",
    "as:relationship": "https://w3id.org/codemeta/3.0#citation",
    "as:object": null,
    "as:name": "FASST",
    "id": "urn:uuid:a27d40bb-4e41-46b6-bbb3-98fd7dcb210b",
    "type": "Relationship"
  },
  "origin": {
    "id": "https://datalake.inria.fr",
    "type": "Service",
    "inbox": "https://prod-datadcis-api.inria.fr/coar/inbox"
  },
  "target": {
    "id": "https://archive.softwareheritage.org",
    "type": "Service",
    "inbox": "https://inbox.staging.swh.network/"
  },
  "type": [
    "Announce",
    "coar-notify:RelationshipAction"
  ]
}
```

Figure 7: Example of the body of the notification sent to Software Heritage

The notifications are sent and managed in a synchronous way, recording information about successful and failed notifications. In future, we can reprocess documents for which the notifications did not work.

Validation from HAL

On the other end of our notification pipeline there is the inbox of HAL (and SWH). HAL notifications are processed through their inbox, and the detected software mentions are displayed on the article's page within the HAL portal (Figure 8).
However, due to HAL's internal regulations, software mentions cannot be displayed publicly without explicit authorisation from the authors. As a result, they are visible only to authors or deposit administrators once they are logged in.
Figure 9 provides a more detailed view of the software mentions, including the textual contexts in which the software appears and an interface element that allows users to accept or reject each mention. When an author performs one of these actions, a notification is sent back to the Datalake inbox to update the status of the software mentions, accordingly.
The validation process is not restricted to the primary authors. Any individual who has ownership of the deposit—such as co-authors, administrators, or librarians—can validate software mentions, ensuring flexibility and shared responsibility throughout the workflow.



Personal Data Management Systems: The security and functionality standpoint

Nicolas Anciaux (1), Philippe Bonnet (2), Luc Bouganim (1), Benjamin Nguyen (3), Philippe Pucheral (1), Iulian Sandu-Popa (4), Guillaume Scerri (1)

1 DAVID - Données et algorithmes pour une ville intelligente et durable - DAVID
2 IT University of Copenhagen
3 SDS - Sécurité des Données et des Systèmes
4 PETRUS - Personal Trusted cloud

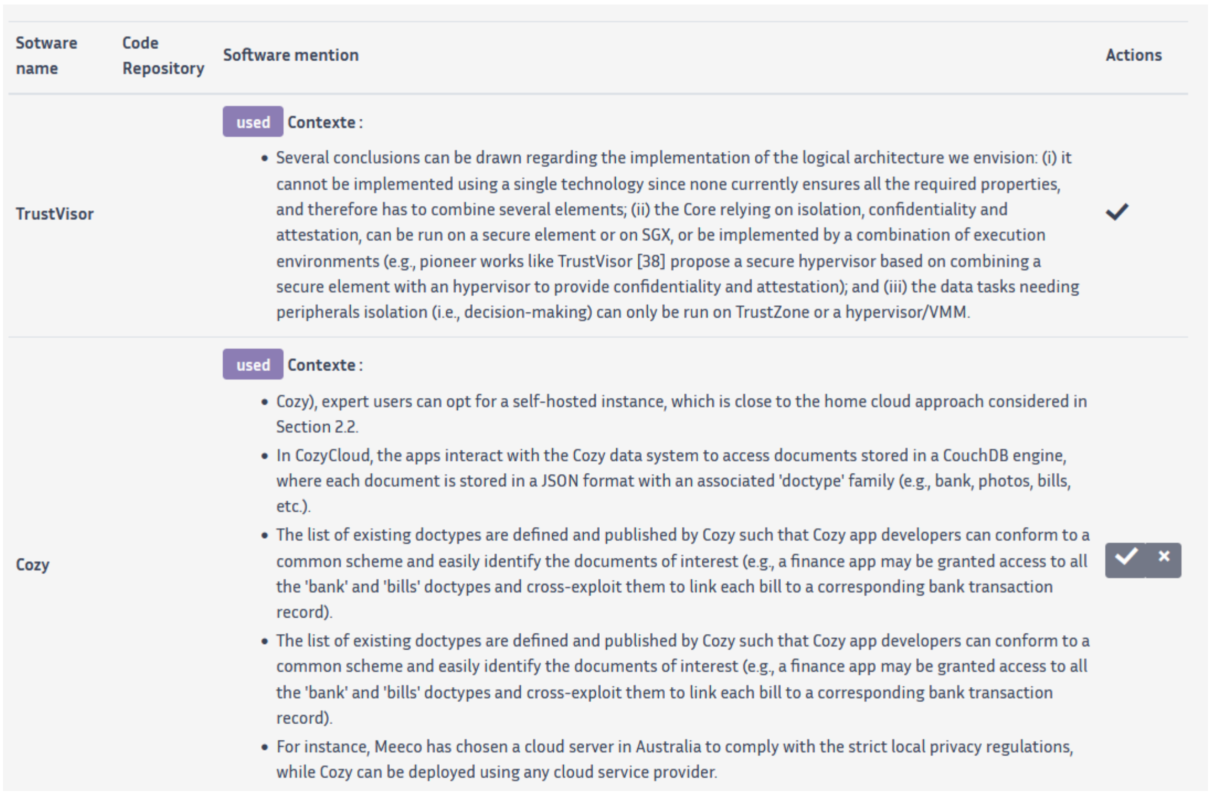
Riding the wave of smart disclosure initiatives and new privacy-protection regulations, the Personal Cloud paradigm is emerging through a myriad of solutions offered to users to let them gather and manage their whole digital life. On the bright side, this opens the way to novel value-added services when crossing multiple sources of data of a given person or crossing the data of multiple people. Yet this paradigm shift towards user empowerment raises fundamental questions with regards to the appropriateness of the functionalities and the data management and protection techniques which are offered by existing solutions to laymen users. These questions must be answered in order to limit the risk of seeing such solutions adopted only by a handful of users and thus leaving the Personal Cloud paradigm to become no more than one of the latest missed attempts to achieve a better regulation of the management of personal data. To this end, we review, compare and analyze personal cloud alternatives in terms of the functionalities they provide and the threat models they target. From this analysis, we derive a general set of functionality and security requirements that any Personal Data Management System (PDMS) should consider. We then identify the challenges of implementing such a PDMS and propose a preliminary design for an extensive and secure PDMS reference architecture satisfying the considered requirements. Finally, we discuss several important research challenges remaining to be addressed to achieve a mature PDMS ecosystem.

Research Software Mentions

Research software mentions detected in the document.

Sotware name	Code Repository	Software mention	Actions
TrustVisor		used Contexte : • Several conclusions can be drawn regarding the implementation of the logical architecture we envision: (i) it cannot be implemented using a single technology since none currently ensures all the required properties, and therefore has to combine several elements; (ii) the Core relying on isolation, confidentiality and attestation, can be run on a secure element or on SGX, or be implemented by a combination of execution environments (e.g., pioneer works like TrustVisor [58] propose a secure hypervisor based on combining a secure element with an hypervisor to provide confidentiality and attestation); and (iii) the data tasks needing peripherals isolation (i.e., decision-making) can only be run on TrustZone or a hypervisor/VMM.	✓
Cozy		used Contexte : • Cozy), expert users can opt for a self-hosted instance, which is close to the home cloud approach considered in Section 2.2. • In CozyCloud, the apps interact with the Cozy data system to access documents stored in a CouchDB engine, where each document is stored in a JSON format with an associated 'doctype' family (e.g., bank, photos, bills, etc.). • The list of existing doctypes are defined and published by Cozy such that Cozy app developers can conform to a common scheme and easily identify the documents of interest (e.g., a finance app may be granted access to all the 'bank' and 'bills' doctypes and cross-exploit them to link each bill to a corresponding bank transaction record). • The list of existing doctypes are defined and published by Cozy such that Cozy app developers can conform to a common scheme and easily identify the documents of interest (e.g., a finance app may be granted access to all the 'bank' and 'bills' doctypes and cross-exploit them to link each bill to a corresponding bank transaction record). • For instance, Meeco has chosen a cloud server in Australia to comply with the strict local privacy regulations, while Cozy can be deployed using any cloud service provider.	✓ ×
Tonido/ FileCloud		shared Contexte : • They come as enterprise cloud tools either implemented as-a-service in the cloud or hosted on a server owned by the company and managed by internal administrators, such as SeaFile, Pydio, ownCloud/NextCloud, Sandstorm or Tonido/FileCloud.	✓ ×
CozyCloud		shared Contexte : • CozyCloud also follows the current security best practices in terms of users' passwords, connections and data at rest encryption. • CozyCloud follows a similar principle, following the motto that 'users will stay because they can leave'. • In CozyCloud, the apps interact with the Cozy data system to access documents stored in a CouchDB engine, where each document is stored in a JSON format with an associated 'doctype' family (e.g., bank, photos, bills, etc.). • In CozyCloud and Meeco, the user's personal cloud is hosted and secured by the cloud service provider on behalf of the personal cloud owner (e.g., using server-side encryption). • For example, CozyCloud (through the 'CozyCollect' application) and Digi.me provide their users with a catalog of connectors to retrieve many kinds of personal data, including financial data (e.g., from banks or PayPal), administrative data (e.g., electricity or telco bills and consumption traces, insurance contracts), social network data (e.g., from Facebook or Pinterest accounts), music (e.g., Spotify), medical information (e.g., from social security institutions, hospitals, or Blue Button compliant sites) or fitness data (e.g., Fitbit), to cite only a few. • For example, CozyCloud's applications and data system code are open source. • Online personal cloud solutions (e.g., CozyCloud, Digi.me, • Many online personal cloud solutions flourish today such as CozyCloud, Digi.me, • In some cases, users can choose to have their personal cloud instance hosted directly by the cloud provider (e.g., CozyCloud) or by any other trusted third party of their choice (e.g., OVH, Dropbox, etc.).	✓ ×

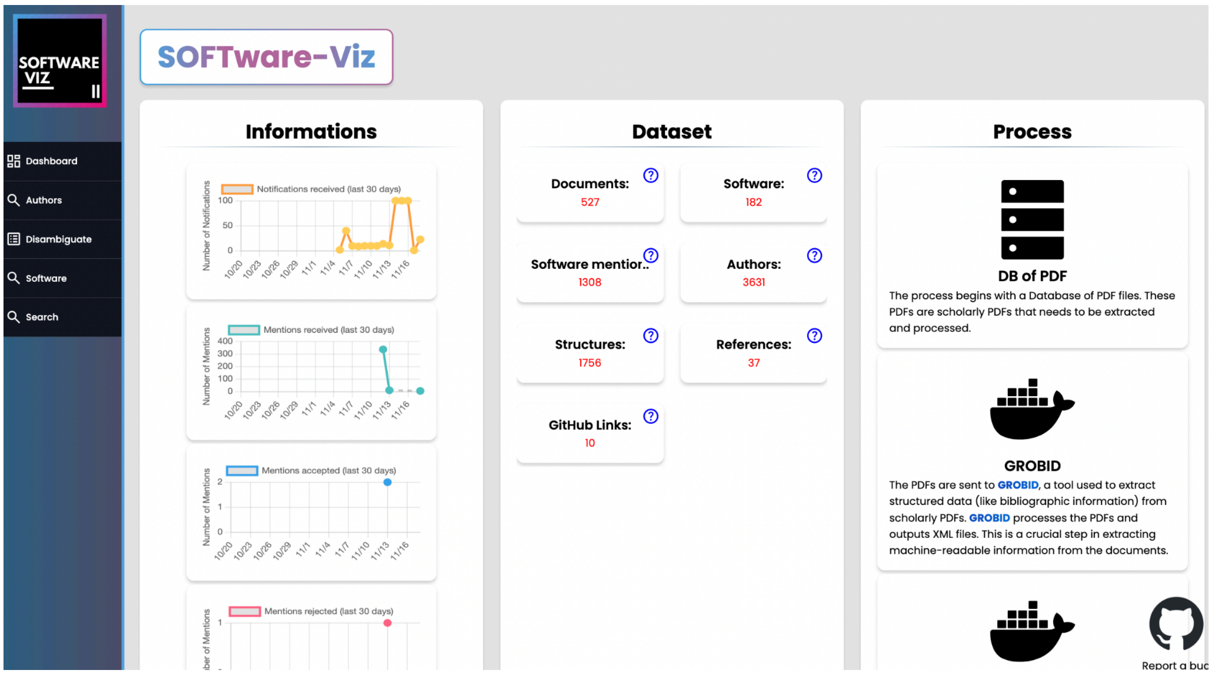
Figure 8: Example of the body of the notification sent to Software Heritage



Sotware name	Code Repository	Software mention	Actions
TrustVisor		used Contexte : • Several conclusions can be drawn regarding the implementation of the logical architecture we envision: (i) it cannot be implemented using a single technology since none currently ensures all the required properties, and therefore has to combine several elements; (ii) the Core relying on isolation, confidentiality and attestation, can be run on a secure element or on SGX, or be implemented by a combination of execution environments (e.g., pioneer works like TrustVisor [38] propose a secure hypervisor based on combining a secure element with an hypervisor to provide confidentiality and attestation); and (iii) the data tasks needing peripherals isolation (i.e., decision-making) can only be run on TrustZone or a hypervisor/VMM.	✓
Cozy		used Contexte : • Cozy), expert users can opt for a self-hosted instance, which is close to the home cloud approach considered in Section 2.2. • In CozyCloud, the apps interact with the Cozy data system to access documents stored in a CouchDB engine, where each document is stored in a JSON format with an associated 'doctype' family (e.g., bank, photos, bills, etc.). • The list of existing doctypes are defined and published by Cozy such that Cozy app developers can conform to a common scheme and easily identify the documents of interest (e.g., a finance app may be granted access to all the 'bank' and 'bills' doctypes and cross-exploit them to link each bill to a corresponding bank transaction record). • The list of existing doctypes are defined and published by Cozy such that Cozy app developers can conform to a common scheme and easily identify the documents of interest (e.g., a finance app may be granted access to all the 'bank' and 'bills' doctypes and cross-exploit them to link each bill to a corresponding bank transaction record). • For instance, Meeco has chosen a cloud server in Australia to comply with the strict local privacy regulations, while Cozy can be deployed using any cloud service provider.	✓ ×

Figure 9: Example of the body of the notification sent to Software Heritage

The loaded documents and their software mentions can be monitored thanks to the SOFTware-Viz application (Figure 10), which provides a dashboard overview of the software notifications: documents loaded (extracted by the pipeline), software loaded, notification accepted in the HAL portal, and notification rejected.


SOFTWARE VIZ
SOFTware-Viz
Dashboard
Authors
Disambiguate
Software
Search
Informations
Notifications received (last 30 days)
Mentions received (last 30 days)
Mentions accepted (last 30 days)
Mentions rejected (last 30 days)
Dataset
Process
Report a bug

Documents: 527

Software: 182

Software mentions: 1308

Authors: 3631

Structures: 1756

References: 37

GitHub Links: 10

DB of PDF

The process begins with a Database of PDF files. These PDFs are scholarly PDFs that needs to be extracted and processed.

GROBID

The PDFs are sent to **GROBID**, a tool used to extract structured data (like bibliographic information) from scholarly PDFs. **GROBID** processes the PDFs and outputs XML files. This is a crucial step in extracting machine-readable information from the documents.

Figure 10: Overview of the SOFTware-Viz interface

Results

The workflow for large scale data processing based on the Grid5000 shared infrastructure was validated in two phases: a) by processing a large quantity of PDF documents (1.6 million) documents from the HAL repository, and then by scheduling the pipeline every day to process only new documents.

The first phase was performed in over three weeks: the Grobid processing took around 1 week and the extraction of software took 2 weeks. This was in line with initial expectation considering that Grobid is extremely fast as compared to any specialised processor.
In this exercise the team was able to optimize the deployment and to tune each processor accordingly with the available resources (e.g. Memory, CPUs, GPUs, type of GPUs).
In the second phase, we tested scheduling the processing regularly every day, over a few months and processing around 250 thousand documents end to end.

Figure 11 illustrates the number of documents processed daily by Grobid and software-mentions. Important to notice that the Software-mention process started to work about one week later. Figure 12, shows that the process was designed to naturally catch up the delayed week.

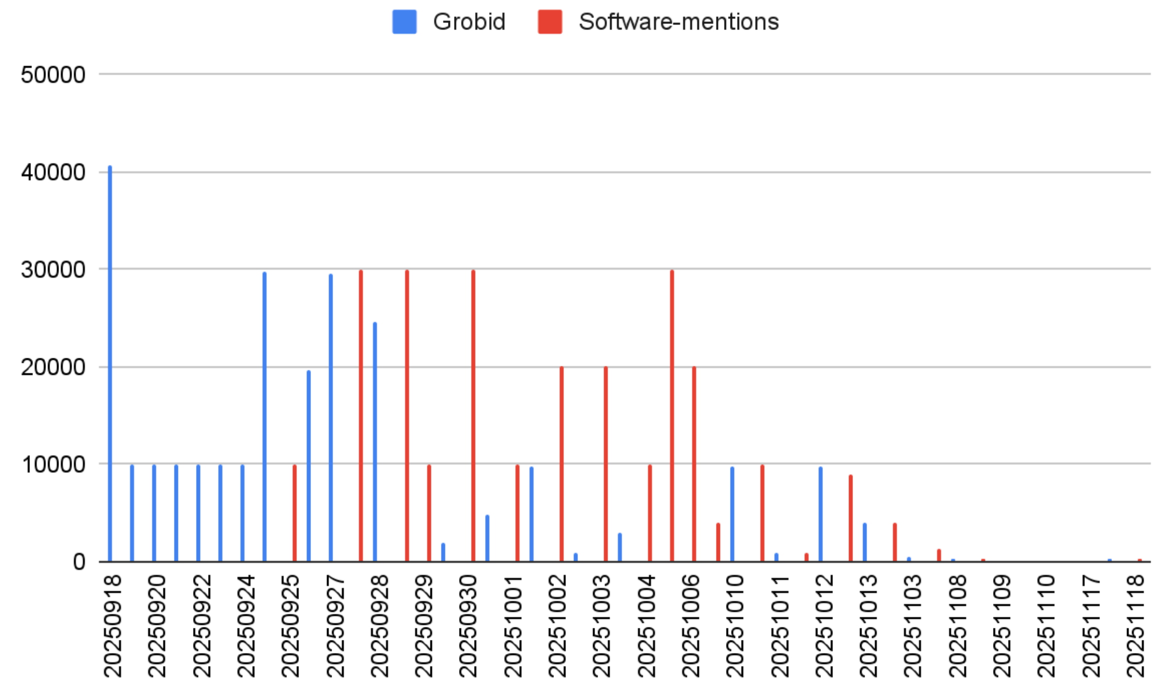


Figure 11: Daily process, we limit to 10000 documents per day. The software-mention process started about a week later. *The days with more than 10000 documents were the days the team was testing changes manually

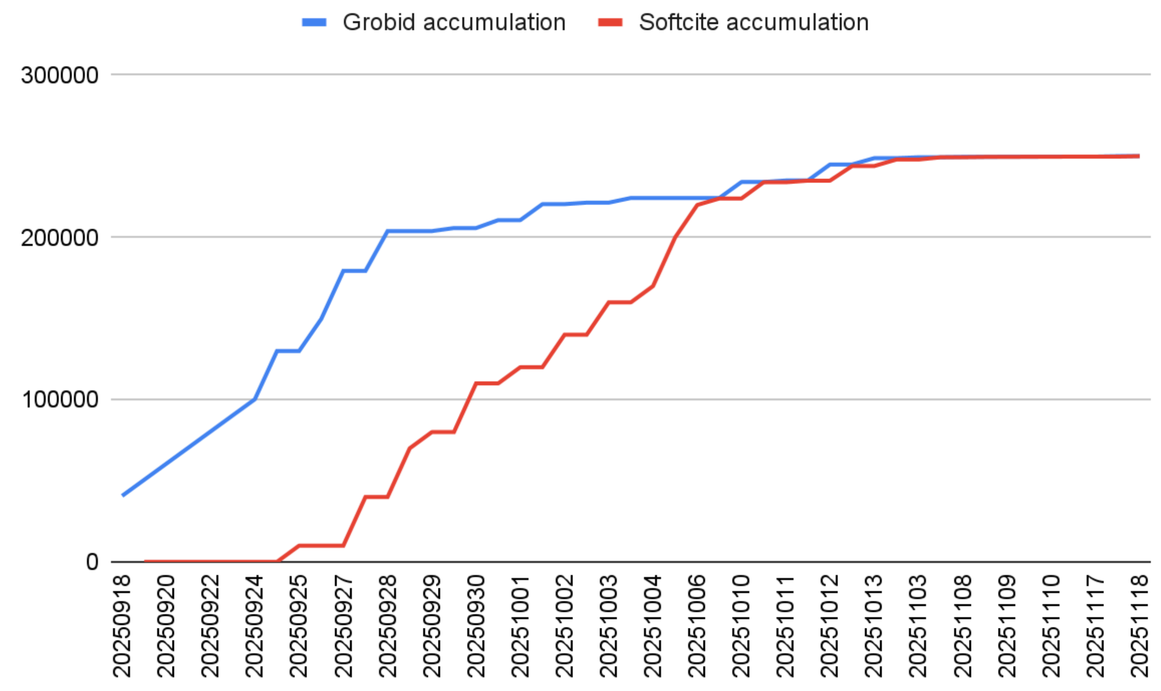


Figure 12: Accumulated documents during the test process. Notice that the software-mention process was designed with a capacity allowing it to catch up after being delayed.

Conclusions

The workflow presented here provides a reliable and scalable end-to-end system for identifying, processing, and validating software mentions in publications deposited in HAL. Leveraging Grid5000 for large-scale computation, Grobid and SoftCite for document analysis, and COAR Notify for interoperable communication, the pipeline ensures that software referenced in scholarly outputs is systematically captured and routed to the HAL portal. Author involvement through the HAL validation interface reinforces accuracy and supports a transparent, accountable process. As it evolves, the pipeline will continue to contribute to a more complete and sustainable ecosystem for recognising research software as a key component of open and reproducible science.

Limitations

The implementation described in this document still presents several limitations.
First, there is currently no direct way to correct extracted information—such as software names or context snippets—within the HAL portal, beyond simply accepting or rejecting the mentions. This limitation raises several questions about user engagement, as authors are generally not very active in maintaining metadata in HAL and may be reluctant to edit information manually. For this reason, an interface offering only Accept/Reject actions is likely to be the most effective option, at least during the first few months following the production release, while we gather feedback and monitor usage patterns.

A second limitation concerns notifications for older publications. At present, authors and administrators will not be alerted about software mentions extracted from older records. This issue requires careful consideration in order to avoid overwhelming users with excessive notifications. One possible compromise is to process a limited historical window—for example, articles published in 2024—to expand coverage while maintaining a reasonable notification load.

2.3. Case study in the digital humanities (IBL-PAN)

Overview

This use case investigates the potential of automated software-mention detection to analyse and interpret digital transformation processes in the humanities. Building on machine-learning tools integrated into GROBID and Softcite, the study introduces a long-term analysis of software-usage signals in humanities scholarship. The core assumption is that software mentions in scholarly publications constitute measurable traces of digital transformation. By examining journals representing traditional linguistics and literary studies (TLL) and digital humanities (DH), the study contributes to a broader understanding of methodological change in SSH fields. Unlike earlier research focused mainly on model development or dataset creation, this use case applies automated detection to a scientometric investigation of disciplinary evolution and assesses how the produced outputs may be reused by SSH infrastructures such as GoTriple and the SSH Open Marketplace.

In this report we present the key elements of the study which has been detailed in a scientific publication submitted to the peer review journal.

Description of workflow

The workflow began with the construction of a corpus. Journals published continuously for at least 20 years were selected through DOAJ. Two categories of journals were analysed: (1) traditional linguistics and literary studies, representing seven titles and more than three thousand full texts, and (2) digital humanities, represented by articles from DHQ and Code4lib, together with a large set of abstracts derived from a DH journal list. Publications were limited to English.

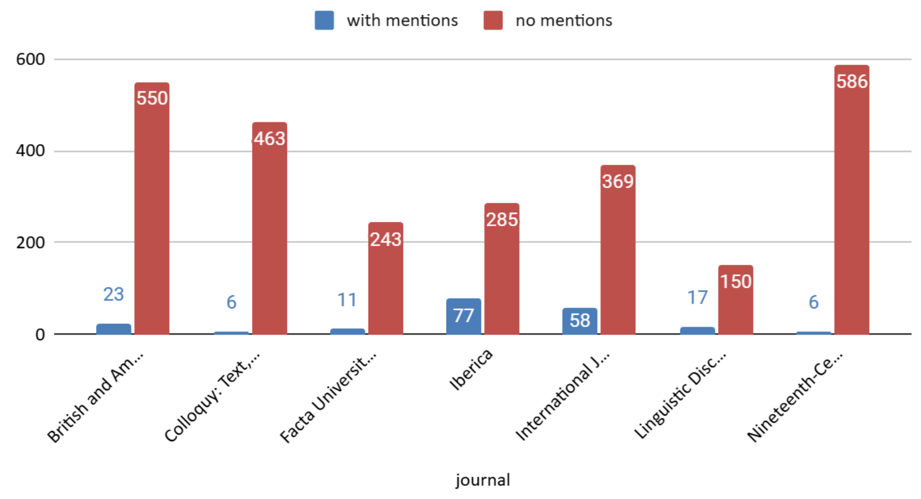
All texts were processed using the Softcite tool, which extracts software mentions through a machine-learning NER pipeline integrated with GROBID. The study adopted a "one mention per document" rule: the presence of a single software instance was sufficient to classify the publication as software-using. This prioritised large-scale scientometric trends over exhaustive linguistic annotation.

Outputs were manually validated and classified as true positives and false positives. This enabled a detailed assessment of detection performance across subdisciplines of the humanities. Mention rates were computed, allowing diachronic comparisons between TLL and DH journals. The study additionally considered abstracts to partially address limitations of the "per document" rule and to measure whether software appears in short-form metadata.

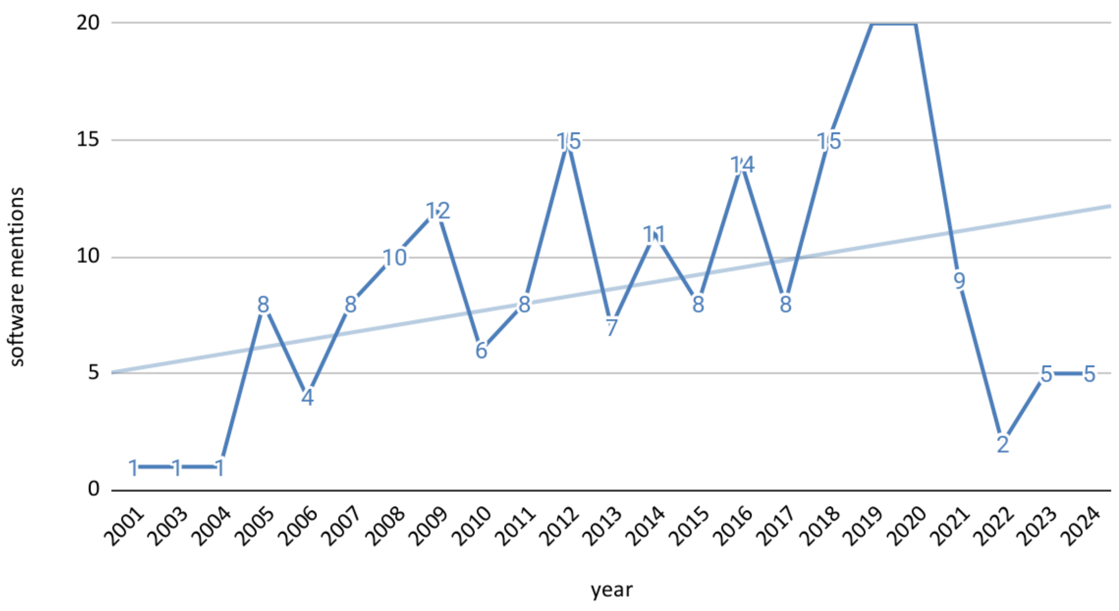
Finally, the findings were mapped onto SSH scholarly infrastructures. The study examined how automated detection could enrich metadata, support knowledge-graph construction, and contribute to discovery services such as GoTriple and the SSH Open Marketplace.

Results

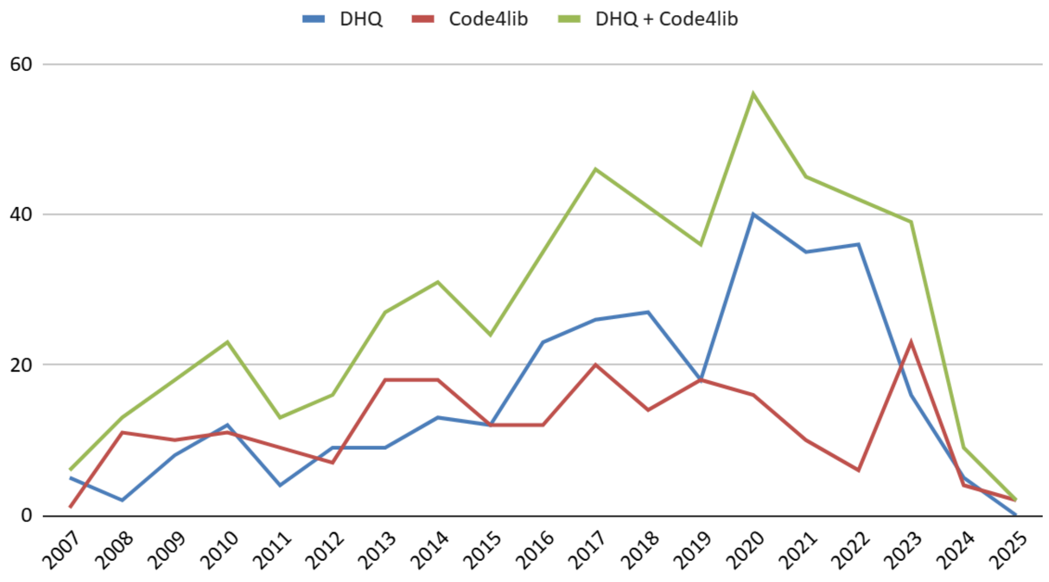
The TLL analysis showed that digital transformation is present but unevenly distributed across journals. Titles such as Iberica and IJES displayed relatively high mention rates, while literary studies journals exhibited minimal traces of software use.



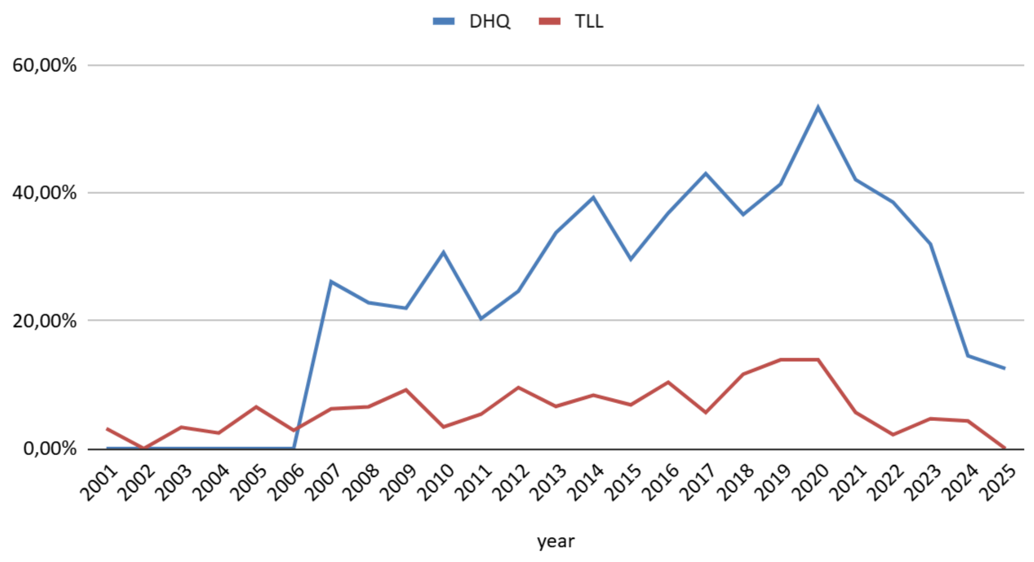
Peaks observed in 2009–2012 and 2020 indicate moments of methodological change, including the pandemic-related rise in digital tools.



In contrast, DH journals showed consistently high levels of software mentions. Code4lib and DHQ demonstrate strong upward trends, with significant growth after 2016. This reflects the established methodological reliance on computational tools within DH communities and the increasing institutionalisation of digital research practices.



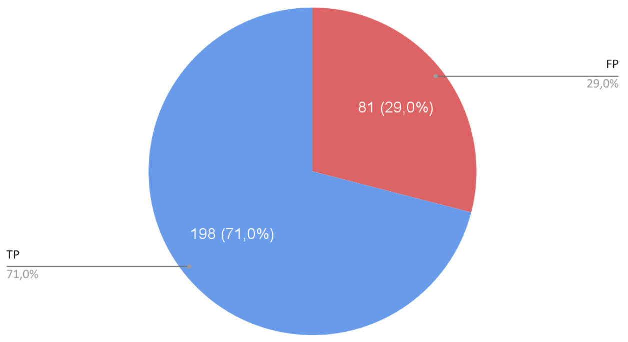
When comparing standardised mention rates, DH journals regularly reached 30–60%, whereas TLL journals rarely exceeded 10–15%. This contrast illustrates structural differences between the two fields and highlights the distinctive trajectories of digital transformation across the humanities.



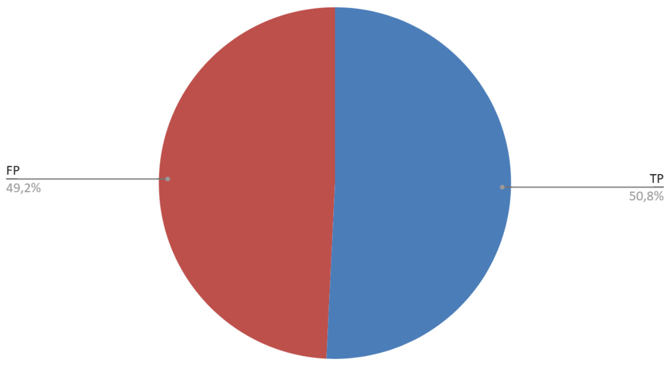
Abstracts proved largely unreliable for software-usage detection. TLL abstracts contained almost no software references, and even DH abstracts presented only isolated mentions. This finding is significant for infrastructures that rely heavily on abstracts for content indexing and search.

Evaluation

The evaluation demonstrated that model performance varies significantly across domains. In TLL journals, approximately 71% of detected mentions were true positives, reflecting the clearer and less technical vocabulary typical of these fields. In DH journals, the true-positive rate was roughly 50%, due to the greater complexity of DH terminology and the presence of numerous tools, platforms, and technical expressions that challenge disambiguation.



TLL



DH

Limitations

The study is limited by its document-level approach, which does not account for the frequency, location, or context of software mentions within texts. This approach sacrifices granularity in favour of comparability and scale. Manual validation was necessary to compensate for false positives, especially in DH journals, which limits the speed of deployment in real infrastructures. The analysis does not systematically address false negatives, as full manual annotation of the corpus would exceed project capacity. Only English-language publications were examined, which restricts the generalisability of results across multilingual SSH domains. Abstracts offer little usable information about software usage, limiting the ability to rely on them for metadata enrichment. Despite these limitations, the workflow demonstrates substantial value for scientometric and infrastructural applications.

Conclusions

Readiness of the project’s outputs for research re-use in scientometrics, bibliometrics, and cultural analytics. The outputs of this use case are suitable for reuse in several research contexts. Scientometric studies may use the corpus, validated outputs, and diachronic trends to analyse digital transformation across SSH domains. Bibliometric research can incorporate software-mention annotations into metadata schemas, citation networks, and knowledge-graph structures. Cultural analytics research gains a robust indicator of methodological change and computational uptake within humanities publications.

The validated mentions also provide additional ground truth for improving machine-learning models, particularly with regard to the disambiguation of technical terminology. Because the dataset aligns with FAIR principles, it can be easily integrated with PIDs, authority files, software archives, and SSH knowledge-graph initiatives.

Potential of the project output for automated population of EOSC services dedicated to SSH. Automated software detection can significantly enhance EOSC-connected SSH services. In GoTriple, detected software mentions could be integrated directly into the indexing pipeline, enabling search by software name or PID and contributing to an emerging SSH knowledge graph. Authors could validate detected mentions through existing user-oriented features.

In the SSH Open Marketplace, detected software could be matched to existing tool entries or used to create new ones. Publications containing mentions would serve as contextual evidence, enriching entry metadata and improving findability and interoperability.

More broadly, linking detected software mentions with persistent identifiers such as SWHIDs would allow EOSC services to populate and update their metadata automatically at scale. Automated detection thus represents a realistic, scalable mechanism for improving metadata quality, discoverability, and cross-service interoperability across SSH infrastructures.

2.4 Demonstrator 3: CORE - Software Heritage loop (scalable extraction)