

Prototype System of Managing Stanford Law Faculty Publications

Sudip Das
11 Dec, 2025

1. Introduction

This work proposes a practical approach to managing Stanford Law faculty publications. A sustainable system architecture for this prototype covers the full pathway of the data: identifying and ingesting records from reliable sources, assessing and cleaning inconsistent entries, validating key fields, and storing them in a structure that supports different kinds of users and use cases. With this foundation in place, the library could more easily answer questions about individual faculty output, track trends over time, and generate internal reports, as well as produce a consistent, high-quality annual faculty bibliography formatted in Bluebook style.

2. Data Collection & Ingestion

The first step is to be explicit about what we want to capture for each publication (e.g., title, authors with roles, faculty affiliation, publication type, date, DOI/ISBN/ISSN) and which sources are most authoritative for each field. Some popular data collection sources are:

- External databases and indexes: Google Scholar, HeinOnline, Westlaw, Lexis, SSRN, Crossref, publisher sites, and law review platforms.
- Faculty and library provided sources: CVs, self-reported updates via a simple web form, internal spreadsheets, and items identified by librarians through news, alerts, or subject searching.

For data ingestion, the initial ecosystem would draw on four main streams:

2.1. Structured imports

This is preferred as it would have the least chance of any errors. Wherever an export or API exists, the system should ingest structured metadata rather than scraping HTML. This includes pulling or uploading records from Crossref, HeinOnline, SSRN, and similar sources via APIs or standard exports (BibTeX, EndNote XML, CSL JSON, CSV).

AI or a rule based LLM output can be used here mainly for field normalization and deduplication (e.g., normalizing journal titles/names, matching DOIs, consolidating duplicates across sources). However AI should not be directly used here to guess any missing value.

2.2. Semi-structured sources

This includes faculty CVs, personal pages, and lab/center sites which often contain long, relatively consistent publication lists. AI-assisted citation parsers / NLP models can be used on these data to split these documents into individual references, classify publication type, and extract structured fields (title, authors, volume, pages).

All the parsed records should land in a staging area, where a library staff can quickly scan, spot obvious errors, and approve or correct. This significantly reduces manual re-typing but is error-prone on complex legal citations and non-standard outputs. Human review remains necessary, especially before records feed into anything used for reporting or formal bibliographies.

2.3. Web scraping

Web scraping can be used where no proper API is available, however it is important to check site terms of service and licensing as some legal websites don't allow web scraping. Some popular web scraping tools are Selenium, Scrapy and Playwright

2.4. Manual entry and faculty self-reporting

Even with good automation, there will be gaps and corrections. Therefore, there will be a simple web form provided for faculty and staff to submit new publications or fix records, with fields aligned to the internal schema. This can include features such as: paste a DOI/ISBN, then automatically pull metadata from Crossref or another service, which users can review and confirm. Manual entry is especially important for edge cases and for correcting AI/parser errors.

3. Data Assessment

The provided CSV represents a snapshot of faculty publication data after the initial collection stage. It includes 1,927 rows and 31 columns, with several key descriptive fields:

- Core descriptive fields: ID, Faculty Contributor, Title, Publication Type, Date, Citation, Designation. These are largely populated with some noticeable gaps and inconsistencies. There are duplicate or near-duplicate rows that appear to refer to the same publication but with slight variations.
- Additional metadata: Status, Serial Title, Publisher, Publication Title, plus identifier and link fields such as Source Link, Stanford Link, DOI, ISBN, ISSN, PURL, SSRN. The high-value identifier fields are sparsely populated. This suggests that either the current upstream sources do not consistently provide these identifiers, or existing extraction workflows are not yet capturing them. These are critical for unambiguous matching, linking, and long-term reliability, and therefore they should be filled, where possible, by querying authoritative services (e.g., Crossref) using known metadata (title, authors) and merging information from multiple records that have been confidently matched to the same publication.

- Curation flags: columns such as Italics, Corrected, In ORCID, which indicate whether records have been reviewed or normalized.

Importantly, these identifiers should not be directly approximated by AI models. For persistent identifiers, the system should rely on rule-based matching and authoritative lookups. AI can assist in normalizing fields (names, titles), clustering records to detect likely duplicates and flagging suspicious or incomplete records for human review. The final assignment of DOIs/ISBNs/ISSNs and other critical identifiers should be grounded in verifiable external data.

4. Data Quality, Curation & Validation

Now that we verified that the collected data are messy, inconsistent, and partially incomplete, the next step is to define and enforce a clear schema and validation rules. The goal is to ensure that key columns are populated, internally consistent, and usable across the lifecycle of the data. In this prototype, that means:

- Standardizing core fields (names, dates, titles, publication types).
- Validating formats (e.g., date strings, identifiers).
- Detecting and resolving duplicates
- Filling missing values

AI can assist with pattern recognition (e.g., clustering duplicates, proposing normalized forms), but the system should still rely on explicit rules and human review for high-stakes fields and final decisions.

4.1 Standardizing Faculty Names

A core requirement is to ensure that all Name fields appear in a consistent format across records. In the CSV, names currently appear in several patterns, for example:

- Faculty Contributor: LastName, FirstName
- Faculty Co-Authors/Editors: “LastName, FirstName”, “LastName, FirstName”
- Co-Authors: FirstName LastName; FirstName LastName
- Editor(s): FirstName LastName; FirstName LastName or comma-separated variants

Some fields start with first name, others with last name; some use commas, others semicolons; some wrap names in quotes. This is not a major issue for manual reading, but it becomes a problem for automated matching, deduplication, or linking to authority files. For example, a simple regex-based comparison will treat Gregory Ablavsky and Ablavsky, Gregory as different strings.

For the prototype, I standardized all name fields to a single canonical pattern:

- LastName, FirstName MiddleName
- Multiple contributors separated by a consistent delimiter (e.g., semicolon):
Ablavsky, Gregory; Doe, Jane

Middle names and initials are preserved in whatever form the individual uses, but always appear after the first name. This improves downstream tasks such as matching contributors across different columns and sources and running analytics on faculty output.

AI or NLP can help with parsing raw name strings into components (first, middle, last), however a simple regex matching is more efficient and less likely to hallucinate for this case. However AI can be useful in detecting likely duplicates or variants of the same name.

In the long run, we could leverage unique identifiers like an internal faculty ID or ORCID ID for each author to link their records. According to ORCID guidelines, linking authors with an ORCID helps to collect all name variants and works, furthermore improving search and disambiguation. This means if an author's name appears differently (due to accents or formal name), the system can still recognize them as the same person via the ORCID or internal ID.

4.2 Normalization of Titles and Citations

String based columns such as Title, Serial Title, Publisher, Publication Title and Citation should pass through basic NLP checks that trim whitespaces, any special characters and HTML tags.

Some titles can appear in multiple languages. Therefore a language detection and tagging system can be implemented which detects the language and sets a Language field automatically (e.g. en, fr, es). This can be later used to track journals that are published in multiple languages rather than being considered as new journals. A text embedding model can be used here to embed the titles and perform similarity checks to flag likely matches for human review before consolidating them into a single authority entry.

4.3 Dates & Categories

The dates column in the CSV is populated but not standardized: some entries include full dates, others only a year or a year and month, and all are stored as strings. For the prototype system, the first step is to parse these values into a consistent internal format. A practical approach is to normalize dates to a YYYY-MM representation, with the option to retain the specific day where it is reliably known.

In terms of publications, the Year is more important than the specific day or month. As the Year column is missing a value, it can be extracted from the Date column. One thing to consider here

is that the raw Date is in string format while Year is in Integer format and therefore it is important to convert while extracting.

Columns such as Publication Type, Status, Designation are categorical columns and therefore should have the list of categories initialized in the schema. Then the extracted data is mapped to one of these categories to make sure it only has values from that list. This will ensure different formats such as ‘Book, Review’, ‘Book Review’ or ‘book review’ are all set to one preset format. In this context, AI can be helpful for suggesting mappings from noisy values to the controlled vocabulary and for flagging unusual or ambiguous labels. Human review is optional here as the chance of errors are very low in this task, however it is always best to check a few random samples to ensure AI is not hallucinating.

The Status field requires particular care for values like “Forthcoming.” These records represent works that may later appear again in the data stream as fully published items. The system should periodically check forthcoming records against external sources to determine whether they have moved to a published state. When a potential match is found, titles and key metadata (such as authors and year) should be compared to confirm that they refer to the same work. AI can play a useful role here by proposing candidate matches but the actual merge or update should be based on deterministic criteria and explicit rules. Once confirmed, the status can be updated, and the older row can either be merged into the new record or retired to avoid double-counting the same publication. This approach keeps dates and categories internally consistent while using AI primarily as a triage and suggestion mechanism rather than an authoritative decision-maker.

4.4 Data Deduplication and Uniqueness

The ID column is intended to serve as a stable key for searching and retrieving records from the database, so it must be unique and non-null. In the current CSV, there is one null ID and only 1,786 unique IDs across 1,927 rows, which immediately indicates the presence of duplicates. Deduplication should therefore proceed in stages, starting with the simplest, highest-confidence checks and moving toward more nuanced, AI-assisted decisions only where necessary.

The first pass focuses on exact duplicates. For each ID, rows that match across all columns can be safely collapsed to a single record, with the redundant copies removed. This can be implemented by temporarily flagging exact-duplicate rows, confirming that all fields are identical, and retaining only one instance. After this pass, it is good practice to recompute the number of remaining duplicate IDs and verify that only near-duplicates remain.

The second pass addresses these near-duplicates. Here, we want to decide which row in each duplicate-ID group should be treated as the canonical record. Rather than dropping rows

arbitrarily, the system should apply explicit rules and use AI only where those rules are insufficient. The Corrected column is particularly useful here. If a duplicate-ID group contains exactly one row marked as corrected or reviewed, that row should be kept and the others removed without involving any model at all. In other words, existing human curation is treated as authoritative whenever it is unambiguous.

For the remaining groups, those with multiple “corrected” rows or none at all, the system can use a LLM in a tightly constrained, rule-based way. Only the small subset of conflicting records within each duplicate-ID group is passed to the model, along with decision rules that prioritize more complete records (fewer missing values), more consistent metadata (matching dates, types), and stronger identifiers (DOI, ISBN, ISSN, stable URLs). The model’s task is simply to select the best candidate within the group. To reduce the risk of hallucination, the prompt restricts the output format to a single integer indicating which row to keep, rather than allowing free-form text. A temporary field (for example, a keep/remove flag) can then encode this decision, and a human reviewer can spot-check a sample of groups to confirm that the model is behaving as expected.

Once this process is complete, a final sanity check ensures that there are no remaining duplicate ID values in the dataset. At this stage, a separate check on titles, citations, and URLs may still reveal duplicate content attached to different IDs. Those cases are better handled after missing values have been filled and key fields normalized, since better metadata will improve both rule-based and AI-assisted detection of duplicate publications across the entire collection.

4.5 High Missing Values (DOI, ISSN, ISBN, URLs)

Metadata such as DOI, ISSN, ISBN, Links, Volume, Edition are key properties of a publication. However the CSV file is missing most of these values. Crossref is a great source to find matching titles using the current Title, Author (Faculty Contributor) and Year columns. A similarity score (a threshold of 0.8 on title similarity) is used to reduce the risk of false positives. Among the top 5 results that exceed this threshold, the best candidate is selected, and its metadata are used to fill in missing values in the local record. If the dataset has existing non-null columns then those don’t get replaced, rather it acts as a checking parameter on how accurate Crossref is versus the actual data in the dataset. The other missing fields are filled through this technique.

Here is an example

Metadata	Before	After
Faculty Contributor	Ablavsky, Gregory	Ablavsky, Gregory

Title	Credit Nation: Property Laws and Institutions in Early America	Credit Nation: Property Laws and Institutions in Early America
Year	2021	2021
Volume	61	61
Issue	NaN	3
Pages	340	340
ISSN	NaN	0002-9319
DOI	NaN	10.1093/ajlh/njab014
Source Link(s)	NaN	https://doi.org/10.1093/ajlh/njab014

One thing to be aware of here is most free API sources come with a rate limits and therefore the whole dataset cannot be checked with a single, naive pass. Instead, records should be processed in batches, with logging to track which rows have already been queried and when. This allows the system to resume or repeat enrichment jobs without re-hitting the same records unnecessarily. For URL fields, a separate validation script can periodically check that stored links still return a valid HTTP status (for example, 200 or 201), and flag broken links for correction or replacement.

In this part of the workflow, the use of generative AI is not recommended. Persistent identifiers like DOIs, ISBNs, and ISSNs must come from verifiable, authoritative sources, not from model-generated guesses.

4.6 Bluebook Style Data Citation

A LLM with few shot prompting of Bluebook style Law citations can be useful in flagging potential citations that are not formatted properly along with a suggested citation. The responses are guardrailed to be in JSON format which can then be used to extract fields such as issues and suggestions.

There is a high chance of hallucination or incorrect response here as generating perfectly formatted Bluebook citations is notoriously complex. However, there are emerging AI-driven tools aiming to do this (e.g. LegalEase Citations). They claim to generate 100% accurate Bluebook citations in seconds but couldn't be tested for this task as it is a subscription based application.

AI citation parsing is helpful here to validate whether metadata fields such as Title, Author, Year, Publication Type, Edition etc matches or not. This is done through passing the required fields to the AI model and outputting additional columns which flags potential mismatches and issues.

5. Storage

A sensible storage design is a two-tier model:

5.1 Staging area: This holds raw and semi-processed records from different sources. It also allows librarians to review AI suggestions, resolve conflicts, and merge duplicates.

5.2 Validated repositories: This holds all validated records which includes unique IDs per publication, faculty names coming from faculty ID or ORCID to ensure all formats are included. This will make it easier to explore trends and relationships such as journals to publishers or faculty authors to number of publications.

This system is cleaner, normalized and avoids duplication but comes at a cost of time and effort whereas raw csv data can be easily used for simpler queries. However, the benefit comes for a long-term system where normalization makes it easier to maintain data and access historical data. When new publications are extracted, those can be compared with the current database to check whether identical publications have been collected or not.

6. Access and Discovery

A Web UI or Streamlit based UI can help in searching by different filters such as faculty, title, year or topic. Search shouldn't be only keyword based and can be enhanced with semantic search. This includes a RAG based pipeline with a vector database that is able to answer questions from the data. Although this sounds more exciting it can produce hallucinations. Therefore, it is crucial to create a policy page with the proper guidelines of use.

7. Maintenance and Evolution

Implementing an AI solution is not a one-time effort as data grows and changes. If a new type of publication appears (say faculty start publishing tweets or something unconventional), the AI classifier might not know how to handle it. Human curators will need to update the system (both the data and the AI parameters). The database should also be checked and validated often to ensure data freshness.

8. System Architecture

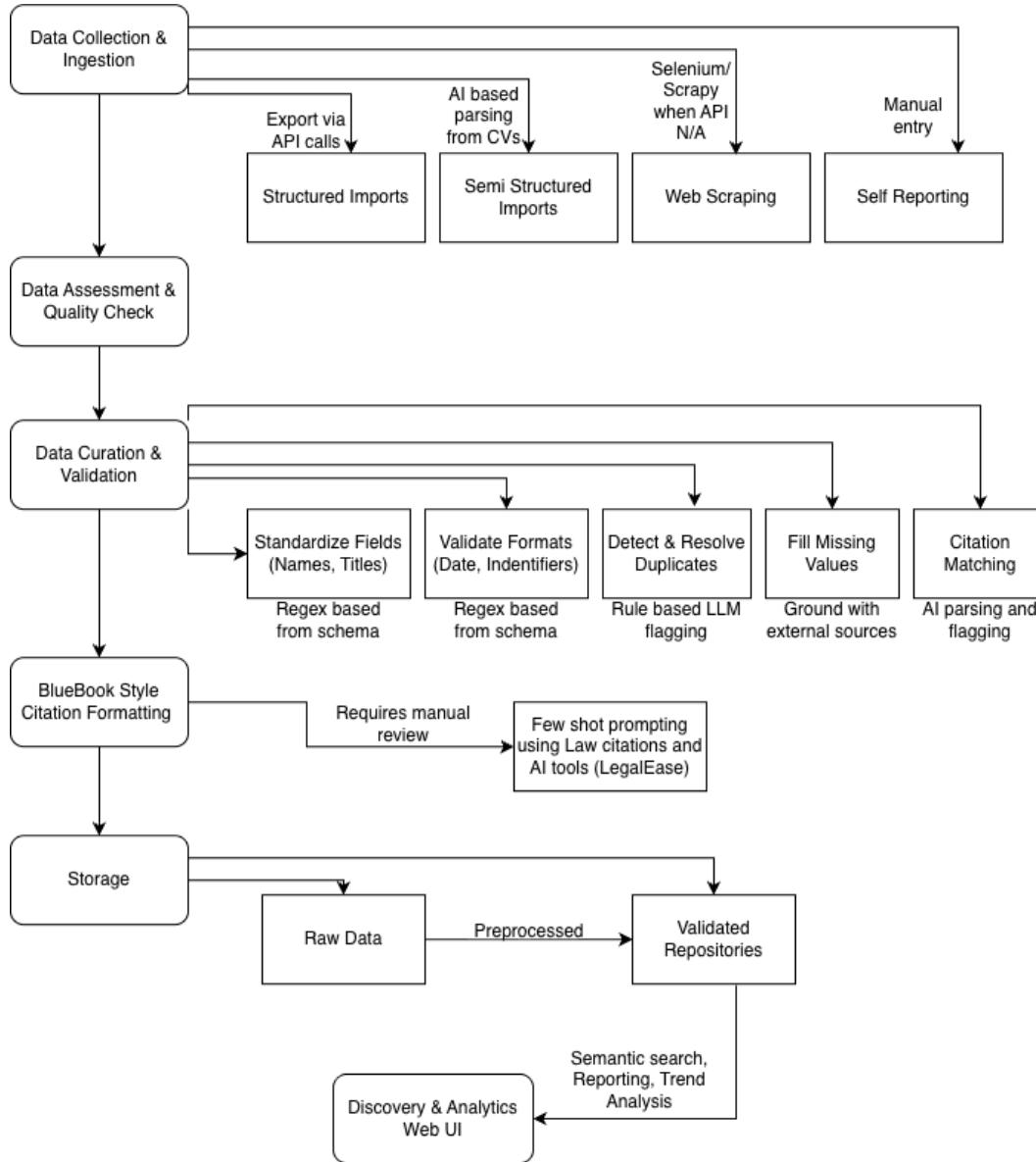


Figure 1. Proposed System Architecture Diagram

9. Conclusion

The prototype system outlines a sustainable workflow for managing faculty publication records. The design deliberately combines rule-based processing, authoritative external services, and carefully scoped AI assistance. Deterministic checks and schemas handle things that must be strictly correct (identifiers, dates, controlled vocabularies), while AI is used where it genuinely adds value: parsing semi-structured sources, suggesting normalizations, and flagging potential

issues for human review. By separating a staging layer from a validated repository, and by treating librarians' decisions and external identifiers as the final source of truth, the Robert Crown Law Library can maintain an up-to-date, high-quality record of faculty publications with less repetitive manual data entry and more attention to quality control, policy, and user service.