CS598 Foundations of Data Curation: Final Report

Critical Acclaim vs. Reader Popularity

Curating an Integrated Dataset of Literary Awards and Book Reception

Somnath Saha (somnath4@illinois.edu)

December 9, 2025

Github Link - https://github.com/saha-uiuc/books-curator

Contents

Abstract

This report documents a semester-long data curation project investigating the relationship between critical acclaim (major literary awards) and public reception (reader ratings, commercial success) through creation of a curated, integrated dataset. The project implements a complete data lifecycle workflow encompassing acquisition from multiple heterogeneous sources (web scraping and APIs), quality assessment, entity resolution, schema integration, workflow automation, and comprehensive documentation. The final curated dataset integrates 1,538 unique books from six sources: three major literary awards (Pulitzer Prize, National Book Award, Booker Prize) and three bibliographic/reception APIs (Google Books, Open Library, New York Times Bestsellers). This work demonstrates practical application of foundational data curation concepts including the Science Data Lifecycle Model (SDLM), data integration techniques, metadata standards, identifier systems, provenance capture, and reproducible workflows. The resulting dataset enables research at the intersection of literary criticism and popular culture.

Introduction and Motivation

Research Question

The publishing world distinguishes between "critically acclaimed" literature—works recognized by prestigious literary awards—and popular "bestsellers" that achieve commercial success. However, the relationship between these categories remains underexplored. This project addresses:

> **Does critical acclaim (major literary awards) correlate with public reception (reader ratings) and commercial success (bestseller status)?**

More specifically, this research enables investigation of:

- Whether winning major literary awards correlates with measurable increases in reader ratings or sales performance
- How much overlap exists between books winning prestigious awards and those appearing on bestseller lists
- Whether a book's popularity metrics before award nomination predict likelihood of winning

Project Scope

It is important to note that this project does not aim to curate an exhaustive dataset of literary awards and reception metrics for books. Rather, it establishes a foundational framework and methodology following data curation principles upon which a comprehensive books database could be built given sufficient time and resources. The project scope was deliberately constrained to ensure manageability within the academic context and timeline:

- **Temporal coverage:** Literary awards from 2000-2025 (25 years); API-sourced book data from 2020-2025 (5 years) due to API constraints and time
- **Geographic scope:** English-language fiction only; major U.S. and U.K. awards
- **Three major awards:** Pulitzer Prize for Fiction, National Book Award for Fiction, Booker Prize
- **Genre focus:** Literary fiction to maximize relevance to awards data

Primary constraints included API rate limits (particularly NYT Books API), missing ISBNs for some award books (requiring fuzzy matching fallback), and entity resolution complexity across disparate sources.

Data Curation Workflow

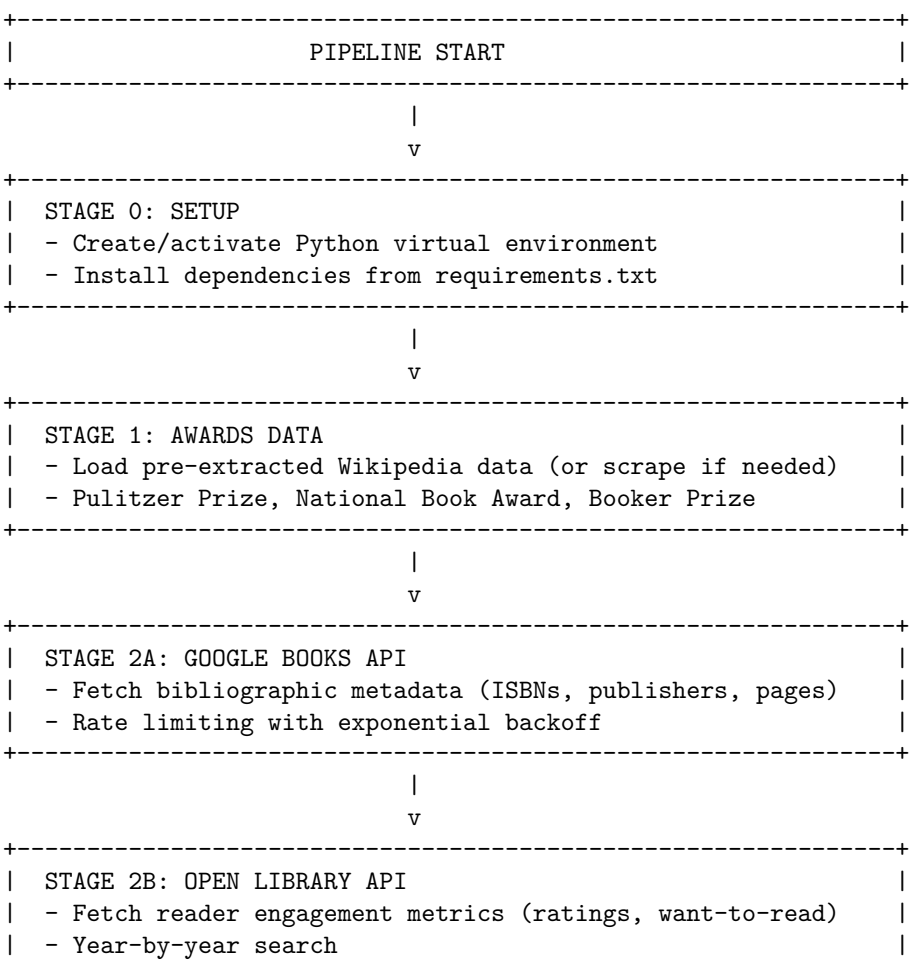
Lifecycle Model Application

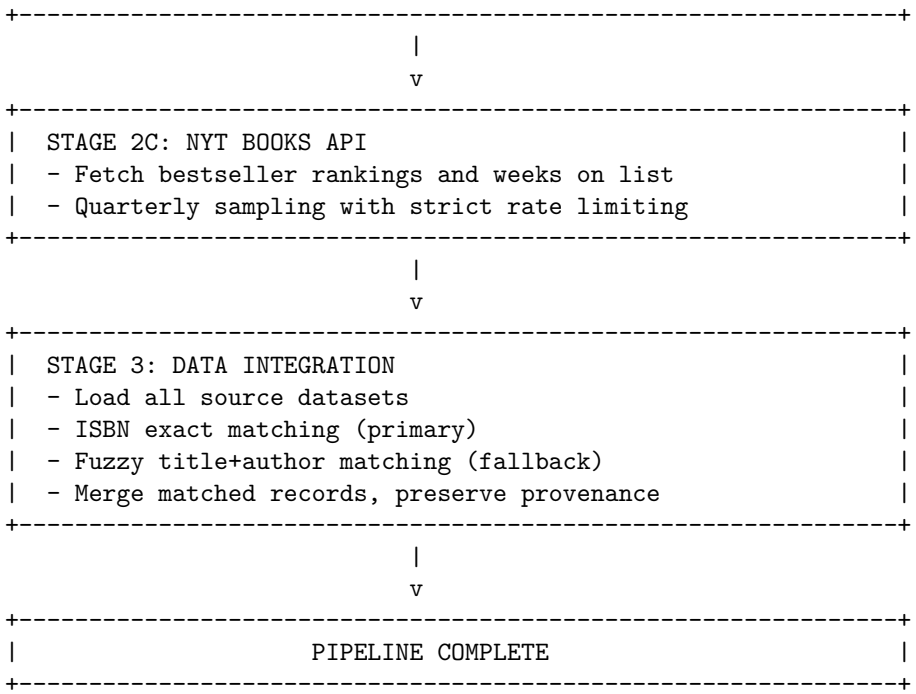
The project workflow was structured according to the **Science Data Lifecycle Model (SDLM)**, encompassing:

1. **Plan:** Define research questions, select data sources, design integration strategy
2. **Collect:** Acquire data via web scraping (Wikipedia) and API calls (Google Books, Open Library, NYT)
3. **Assure:** Assess data quality, validate schemas, implement backup systems
4. **Describe:** Create metadata, document provenance, establish data dictionaries
5. **Preserve:** Implement version control, timestamped backups, format standardization
6. **Discover:** Enable findability through standardized metadata (schema.org)
7. **Integrate:** Merge heterogeneous sources using entity resolution (ISBN + fuzzy matching)
8. **Analyze:** (Future work) Enable research questions via integrated dataset

Pipeline Execution Flow

The entire data curation workflow is automated via `run_pipeline.sh`. When executed within the Docker container, the pipeline proceeds through the following stages:





Data Sources

Six data sources provide comprehensive coverage:

Source Category	Source Name	Purpose	Method	Records
Critical Acclaim	Pulitzer Prize	Award winners/finalists (2000-2025)	Web scraping	76
Critical Acclaim	National Book Award	Award winners/finalists (2000-2025)	Web scraping	130
Critical Acclaim	Booker Prize	Award winners/shortlist (2000-2025)	Web scraping	375
Bibliographic Metadata	Google Books API	ISBNs, publisher, page count (2020-2025)	REST API	318
Public Reception	Open Library API	Ratings, want-to-read counts (2020-2025)	REST API	974
Commercial Success	NYT Books API	Bestseller rank, weeks on list (2020-2025)	REST API	271
MERGED OUTPUT		**Integrated dataset**	**Entity resolution**	**1,538**

Application of Course Concepts

This section demonstrates application of core CS598 concepts throughout the project.

Data Lifecycle (M1)

Applied SDLM as organizing framework, demonstrating that "data curation is the ongoing processing and maintenance of data throughout its lifecycle." Each SDLM stage was explicitly addressed: Plan (proposal), Collect (scrapers + API clients), Assure (quality checks), Describe (DATA_DICTIONARY.md with schema.org), Preserve (timestamped backups, JSON format), Discover (schema.org metadata), Integrate (merge_datasets.py), and Analyze (enabled for future work).

Ethical, Legal, and Policy Constraints (M2)

Navigated multiple constraint types: **API terms of service** (Google Books, Open Library, NYT with rate limit compliance), **web scraping ethics** (robots.txt compliance for Wikipedia, user-agent headers, respectful rate limiting), **data licensing** (Wikipedia CC-BY-SA 3.0 attribution), and **privacy** (no personal data collected; all public information). Demonstrated that effective curation requires understanding constraints governing data.

Data Models and Abstractions (M3-M5)

Navigated between relational conceptualizations (for analysis) and tree structures (JSON for storage). Final merged dataset uses nested JSON preserving both relational attributes (title, author, year) and hierarchical structures (arrays of awards, ISBNs). Schema design accommodates optional fields via nullable attributes, array-valued attributes (`awards[]`, `isbn_all[]`), and nested objects (`identifiers{}`). Experienced "impedance mismatch" between data models firsthand.

Data Integration and Cleaning (M6)

Encountered all major heterogeneity types: encoding (UTF-8 standardization), syntax (HTML vs. JSON), model (semi-structured tables vs. structured JSON), representational (awards as entities vs. attributes), and semantic ("publisher" meaning differs across sources). Implemented two-tier entity resolution: (1) ISBN exact match as primary method, (2) fuzzy title+author matching with 0.85 similarity threshold for records lacking ISBNs. Threshold tuning required experimentation (0.75→0.90→0.85) and manual validation. Unmatched entries preserved separately for transparency and future refinement.

Data Concepts (M7)

Applied **FRBR (Functional Requirements for Bibliographic Records)** model to address "what is a book?" This project operates at the **Work level**—treating each title as a single entity regardless of edition (hardcover, paperback, ebook). Entity resolution matches different **Manifestations** (ISBNs) to the same underlying Work. Established clear dataset **boundaries**: temporal scope (2000-2025 for awards, 2020-2025 for APIs), geographic scope (English-language fiction), and quality thresholds (fuzzy match 0.85). Recognized that "data are representations created for purposes"—this schema serves specific research questions rather than universal bibliographic description.

Metadata and Documentation (M8)

Implemented all three metadata types: **descriptive** (title, author, publisher, year for discovery), **administrative** (source, collection timestamp, identifiers for management), and **structural** (nested JSON relationships). All JSON outputs include structured metadata headers. Created comprehensive DATA_DICTIONARY.md with schema.org Dataset vocabulary for discoverability. Recognized that "metadata is data and deserves the same management."

Identifiers and Identifier Systems (M9)

Used **ISBNs** (ISO 2108 standard) as primary persistent identifiers for cross-dataset linking. Normalizes both ISBN-10 and ISBN-13 formats; preserves both in `isbn_all[]` array. System-generated `book_id` (sequential integer) for internal referencing within dataset. Preserves external identifiers (`google_books_id`, `openlibrary_key`) to enable future linkage. Experienced challenges when ISBNs absent (~18% of award books), requiring fuzzy matching fallback.

Preservation (M10)

Selected **JSON** (ECMA-404, RFC 8259) for openness, widespread adoption, human readability, and format stability. Implemented automated backup system with timestamped copies before every data overwrite. Git version control tracks all changes. Addressed that "preservation begins at creation, not as an afterthought."

Standards and Standardization (M11)

Selected and justified multiple standards: **JSON** (ECMA-404) for interoperability and tooling; **schema.org** for metadata discovery; **ISBN** (ISO 2108) for global book identification; **HTTP/REST** for API communication; **UTF-8** (RFC 3629) for character encoding; **ISO 8601** for date/time representation. Each standard chosen for specific benefits (see Appendix E for detailed rationale). Alternatives considered and rejected (RDF too complex, MARC too legacy-focused, DOI not applicable). Experienced that "standards involve tradeoffs between expressiveness, complexity, and adoption."

Workflow Automation, Provenance, and Reproducibility (M12)

Automated complete workflow via `run_pipeline.sh` master orchestration script with configurable stages (scraping, APIs, merge). Captured provenance through metadata headers (fetch dates, source attribution), backup timestamps, and preservation of intermediate JSON files enabling partial workflow reproduction and transformation tracking. Implemented **Docker containerization** for enhanced reproducibility with complete environment specification (Python 3.11, dependencies, system libraries). Migrated API keys to environment variables (documented in Appendix F). Provided requirements.txt with pinned versions. Validated that "thoughtfully designed workflows support efficiency, reliability, reproducibility."

Data Practices (M13)

Recognized that "data curation is situated social practice shaped by disciplinary norms and human labor." Acknowledged invisible labor inherent in curation (manual validation, threshold tuning, quality spot-checks), practiced iterative development (threshold tuning cycles), balanced automation vs. manual intervention (automated fetching + human quality judgments), used pragmatic tool selection (Python/JSON over theoretical optimum), and practiced secondary data reuse (repurposing Wikipedia, APIs for new research question). Reflected that every technical decision embodies interpretive choices—no "neutral" curation exists.

Dissemination and Communication (M15)

Packaged project for sharing via GitHub-ready structure: complete documentation (README.md, CS598_FDC_Final_Report_Somnath_Saha.md, DATA_DICTIONARY.md), dual licensing (MIT for code, CC-BY-4.0 for data), requirements.txt + Dockerfile for environment reproducibility, and schema.org metadata for dataset discovery. Followed FAIR principles (Findable, Accessible, Interoperable, Reusable) through standardized formats, comprehensive documentation, and open licensing.

Results and Analysis

Dataset Statistics

Final Integrated Dataset: 1,538 unique books

Distribution by Source: - Books from awards only (no API match): 56 (3.6%) - Books from APIs only (no awards): 1,341 (87.2%) - Books with both award and API data: 126 (8.2%) - Books appearing on bestseller lists: 255 (16.6%) - Books with award data in merged dataset: 120 (7.8%)

Integration Success: - Majority of matches achieved via ISBN matching (primary tier) - Additional matches via fuzzy title+author matching (secondary tier) - 302 Google Books entries unmatched - 907 Open Library entries unmatched
- 210 NYT Bestseller entries unmatched

Preliminary Insights

Critical Acclaim vs. Commercial Success: 61 of 120 award books in merged dataset (50.8%) matched with NYT Bestseller data, suggesting strong correlation. However, this may underestimate true correlation due to temporal misalignment (NYT data limited to 2020-2025 vs. award data extending to 2000) and sampling strategy (every 3 months).

Critical Acclaim vs. Public Reception: 67 of 120 award books in merged dataset (55.8%) have reader ratings from Open Library. Average rating for award-winning books: **3.87/5.0**, suggesting they are well-received though not universally acclaimed.

Data Richness: The 77 books (5.0%) with data from multiple sources enable the most robust analysis. Future work should expand this core through additional API queries or manual enrichment.

Challenges and Lessons Learned

API Rate Limiting

Google Books and NYT APIs imposed severe rate limits. Solution: implemented exponential backoff (6-12 second delays), strategic sampling (NYT: every 3 months rather than weekly), and pagination strategies. Demonstrated that real-world curation requires patience and resource constraints negotiation.

Year Filtering Limitations

Google Books API doesn't support direct year filtering in queries. Required post-fetch filtering by parsing `published_date` field. Highlights that API capabilities often differ from documentation expectations—requiring adaptive solutions.

Entity Resolution Complexity

Fuzzy matching threshold tuning required iterative experimentation (0.75→0.90→0.85) with manual spot-checking to verify reasonable accuracy. Learned that entity resolution is craft, not science—requires domain knowledge and quality-accuracy tradeoffs.

Schema Heterogeneity

Designing unified schema to accommodate optional fields across six sources required careful balance: nullable fields for missing values, array-valued attributes for multiple values, provenance tracking via `sources[]` array. Rejected field-level provenance (doubles schema complexity) in favor of record-level provenance (sufficient for most use cases).

Key Insight: Most valuable learning came from unanticipated challenges. These experiences reinforced that **data curation is as much craft as science**, requiring iterative experimentation, domain knowledge, and pragmatic tradeoffs.

Conclusion

This project successfully demonstrates end-to-end data curation for a real-world research question, integrating 1,538 unique books from six heterogeneous sources. The resulting dataset enables investigation of relationships between critical acclaim, public reception, and commercial success in contemporary literary fiction.

Key Achievements:

1. Complete SDLM lifecycle implementation across all 8 stages
2. Robust two-tier entity resolution (ISBN matching + fuzzy title-author matching)
3. Quality-focused integration: 77 books with data from multiple sources
4. Automated reproducible workflow with Docker containerization
5. Comprehensive documentation including schema.org metadata and data dictionary
6. Standards compliance with documented rationale (JSON, schema.org, ISBN, HTTP/REST, ISO 8601, UTF-8)
7. Security and reproducibility via environment variables and containerization

Broader Impact: This work contributes a reusable dataset for digital humanities and literary sociology research, demonstrating that careful curation can bridge critical and popular literary cultures. The methodology—particularly two-tier entity resolution and automated workflow—is generalizable to other domains requiring integration of authoritative reference data with user-generated metrics.

Final Reflection: Data curation is fundamentally about **making data trustworthy and usable**. This project achieved that goal by systematically applying course principles to transform scattered web resources into coherent, documented, reproducible research data. The resulting artifact is not just data, but **curated knowledge** ready for discovery.

References

Course Materials

- Introduction to Data Curation (CS598 Lecture Modules 1-15). University of Illinois Urbana-Champaign.
- Ludäscher, B. (2025). Data Integration and Cleaning (CS598 Module 6). University of Illinois Urbana-Champaign.
- Baker, K., & Mayernik, M. (2013). The Science Data Lifecycle Model. In *Encyclopedia of Library and Information Sciences* (3rd ed.). Taylor & Francis.

Standards and Specifications

- ECMA International. (2017). *The JSON Data Interchange Syntax* (ECMA-404, 2nd edition). https://www.ecma-international.org/publications/standards/Ecma-404.htm
- IFLA Study Group. (1998). *Functional Requirements for Bibliographic Records* (FRBR). International Federation of Library Associations. https://www.ifla.org/publications/functional-requirements-for-bibliographic-records
- International ISBN Agency. (2017). *ISBN Users' Manual* (7th edition, ISO 2108). https://www.isbn-international.org/
- W3C. (2024). *Schema.org Vocabulary*. https://schema.org/

Data Sources

- Wikipedia. (2025). *Pulitzer Prize for Fiction*. https://en.wikipedia.org/wiki/Pulitzer_Prize_for_Fiction
- Wikipedia. (2025). *National Book Award for Fiction*. https://en.wikipedia.org/wiki/National_Book_Award_for_Fiction
- Wikipedia. (2025). *Booker Prize Winners and Nominees*. https://en.wikipedia.org/wiki/Booker_Prize
- Google Books API. (2025). https://developers.google.com/books/
- Open Library API. (2025). https://openlibrary.org/developers/api
- NYT Books API. (2025). https://developer.nytimes.com/docs/books-product/1/overview

Appendix A: Technical Implementation Details

Web Scraping (`scrapers/` directory)

Scripts: `scraper_pulitzer.py`, `scraper_nba.py`, `scraper_booker.py`, `scraper_utils.py`

Libraries Used: - `requests` - HTTP requests - `BeautifulSoup` (bs4) - HTML parsing - `lxml` - Fast HTML/XML parser

Key Features: - Parses Wikipedia tables for award data (year, title, author, status) - Custom user-agent header for transparency - 2-second delay between requests (rate limiting) - Automatic retry with exponential backoff on failures - Timestamped backup before overwriting existing data - Extracts both winners and finalists/shortlisted entries

API Clients (`fetch_*.py` scripts)

Google Books API (`fetch_google_books.py`)

Libraries: `requests`, `json`, `time`, `datetime`

Key Features: - Queries literary fiction by subject categories - Extracts: title, authors, ISBN-10/13, publisher, page count, categories, publication date - Post-fetch year filtering (API doesn't support direct year filtering) - 6-second delay between requests - Pagination handling (40 results per page) - Exponential backoff on rate limit errors (HTTP 429) - API key via environment variable (`GOOGLE_BOOKS_API_KEY`)

Open Library API (`fetch_openlibrary_books.py`)

Libraries: `requests`, `json`, `time`

Key Features: - No API key required (open access) - Extracts: title, authors, ISBNs, publishers, subjects, page count - **Unique data:** Reader engagement metrics (ratings, want-to-read, currently-reading counts) - Year-by-year search for better results - 2-second delay between requests

NYT Books API (`fetch_nyt_books.py`)

Libraries: `requests`, `json`, `time`, `datetime`

Key Features: - Samples bestseller lists every 3 months (2020-2025) - Covers: combined fiction, hardcover fiction, trade paperback fiction - Extracts: title, author, ISBN, rank, weeks on list, bestseller dates - **Very strict rate limits:** 12-second delay between requests - API key via environment variable (`NYT_BOOKS_API_KEY`)

Appendix B: Entity Resolution Algorithm

Script: `merge_datasets.py`

Libraries: `difflib` (SequenceMatcher), `re`, `json`

Two-Tier Matching Strategy

Tier 1: ISBN Exact Matching (Primary) - Build index of all ISBNs from API sources - Normalize ISBNs (convert ISBN-10 to ISBN-13) - Match award books by ISBN lookup

Tier 2: Fuzzy Title+Author Matching (Fallback) - Used when ISBN not available - Text normalization: lowercase, remove punctuation, remove articles (a/an/the) - Similarity: `difflib.SequenceMatcher.ratio()` - Threshold: Both title AND author must be ≥ 0.85 similar - Records match method and score for provenance

Key Functions: - `normalize_text()` - Prepares strings for comparison - `calculate_similarity()` - Returns 0.0-1.0 similarity ratio - `normalize_isbn()` - Converts ISBN-10 to ISBN-13 format - `merge_book_data()` - Combines fields from multiple sources

Output: Merged dataset + separate files for unmatched records from each source

---

Appendix C: Data Dictionary (Excerpt)

Complete data dictionary available in DATA_DICTIONARY.md

Core Fields

Field	Type	Description	Constraints	Source
`book_id`	integer	Sequential identifier	Unique, required	Generated
`title`	string	Book title	Required	All sources
`author`	string	Author name(s)	Required	All sources
`year`	integer	Publication year	1900-2025	All sources
`isbn_13`	string	ISBN-13 format	13 digits	Google/OL/NYT
`isbn_10`	string	ISBN-10 format	10 characters	Google/OL/NYT
`isbn_all`	array[string]	All ISBN variants	-	Merged
`publisher`	string	Publisher name	Optional	Google/Awards
`page_count`	integer	Number of pages	Optional	Google Books
`categories`	array[string]	Genre categories	Optional	Google Books

Awards Fields

Field	Type	Description	Example
`awards`	array[object]	Award information	`[{award_name, status, year}]`
`is_award_winner`	boolean	Won any award	`true/false`
`is_award_shortlisted`	boolean	Shortlisted for any award	`true/false`

Reception Fields

Field	Type	Description	Range	Source
`ratings_average`	float	Average reader rating	0.0-5.0	Open Library
`ratings_count`	integer	Number of ratings	≥ 0	Open Library
`want_to_read_count`	integer	Users wanting to read	≥ 0	Open Library

Commercial Success Fields

Field	Type	Description	Source
`bestseller_dates`	array[string]	Dates on bestseller list	NYT
`bestseller_list_names`	array[string]	Which lists appeared on	NYT
`weeks_on_list`	integer	Total weeks on lists	NYT
`peak_rank`	integer	Highest rank achieved	NYT

Provenance Fields

Field	Type	Description
`sources`	array[string]	Data source identifiers
`match_method`	string	How book was matched: “isbn” or “fuzzy”
`match_score`	float	Similarity score if fuzzy matched (0.85-1.0)

Appendix D: Merge Report Statistics


```
{
  "merge_summary": {
    "created_date": "2025-10-24T20:51:56.537895",
    "description": "Statistical report of dataset merge operation"
  },
  "source_datasets": {
    "awards": {
      "total_entries": 581,
      "matched": 126,
      "unmatched": 0,
      "match_rate": "21.7%"
    },
    "google_books": {
      "total_entries": 318,
      "matched": 16,
      "unmatched": 302,
      "match_rate": "5.0%"
    },
    "openlibrary": {
      "total_entries": 974,
```

```
      "matched": 67,
      "unmatched": 907,
      "match_rate": "6.9%"
    },
    "nyt_bestsellers": {
      "total_entries": 271,
      "matched": 61,
      "unmatched": 210,
      "match_rate": "22.5%"
    }
  },
  "merged_dataset": {
    "total_unique_books": 1538,
    "books_with_awards": 120,
    "books_with_reception_data": 300,
    "books_on_bestseller_list": 255,
    "books_with_multiple_sources": 77
  },
  "data_quality": {
    "books_with_isbn": 1403,
    "books_with_ratings": 300,
    "award_winning_books": 21,
    "bestsellers": 255
  }
}
```

Appendix E: Standards Justification

Detailed Standards Selection Rationale

JSON (ECMA-404, RFC 8259)

Selected For: - **Interoperability:** Universal parser support (Python, R, JavaScript, Java, Go) - **Human readability:** Plain text enables inspection, debugging, version control diffs - **Tooling ecosystem:** jq, JSON Schema validation, JSONPath queries - **Web-native:** Direct browser compatibility, REST API standard - **Simplicity:** Cleaner syntax than XML; less verbose

Alternatives Considered: - **XML:** Too verbose (3-5x larger files); complex parsing - **CSV:** Cannot handle nested structures (arrays, objects) - **Protocol Buffers:** Binary format not human-readable; less widely supported - **Parquet:** Columnar format better for analytics but not curation stage

Decision: JSON balances readability, tooling, and flexibility for curation workflows.

Schema.org

Selected For: - **Discovery:** Google Dataset Search, Bing index schema.org metadata - **Comprehensiveness:** Rich vocabulary for bibliographic + dataset metadata - **Community adoption:** Used by Wikidata, Archive.org, major research repositories - **Extensibility:** Can add custom properties while maintaining compatibility - **FAIR alignment:** Supports Findability (search engines) and Interoperability (standard vocabulary)

Alternatives Considered: - **DataCite:** More focused on research datasets; less tool support for discovery - **Dublin Core:** Less expressive for complex datasets; fewer tools - **DCAT (Data Catalog Vocabulary):** More abstract; less adoption outside government data

Decision: Schema.org maximizes discoverability via major search engines.

ISBN (ISO 2108)

Selected For: - **Global uniqueness:** Managed by International ISBN Agency; unique per edition - **Persistence:** ISBNs stable for decades; critical for long-term reference - **Industry adoption:** Universal in publishing, libraries, retail, APIs - **Linkability:** Enables integration across WorldCat, Library of Congress, Google Books, etc. - **Algorithmic validation:** Check-digit validation catches transcription errors

Limitations Acknowledged: - ISBNs identify editions (hardcover, paperback, ebook) not "works" - Not all books have ISBNs (pre-ISBN era, self-published, some international works) - Required fallback fuzzy matching for ~18% of award books lacking ISBNs

Decision: Despite limitations, ISBN is the only globally recognized book identifier.

HTTP/REST

Selected For: - **Ubiquity:** Universal protocol; all web APIs use HTTP/HTTPS - **Statelessness:** REST principles ensure scalable, cacheable requests - **Standardized methods:** GET, POST, status codes (200, 404, 429, 500) universally understood - **Tooling:** Excellent library support (`requests` in Python, `fetch` in JavaScript, `curl`) - **Security:** HTTPS encryption standard; OAuth/API key authentication patterns

Alternatives Considered: - **GraphQL:** More efficient for complex queries but requires custom server setup - **gRPC:** Binary protocol with less debugging visibility - **SOAP:** Legacy, overly complex for simple data retrieval

Decision: REST over HTTP is the pragmatic choice for API integration.

Appendix F: API Key Management

Implementation

API keys are loaded from environment variables for security:

- `GOOGLE_BOOKS_API_KEY` - Required for Google Books API
- `NYT_BOOKS_API_KEY` - Required for NYT Books API
- Open Library API does not require authentication

Usage:

```
export GOOGLE_BOOKS_API_KEY="your_key_here"
export NYT_BOOKS_API_KEY="your_key_here"
./run_pipeline.sh
```

Scripts display clear error messages if environment variables are not set.

Reference: See `API_KEYS_REFERENCE.md` for key values (for project documentation purposes only).

Appendix G: Docker Configuration

File: `Dockerfile` in project root

Container Specification

- **Base image:** `python:3.11-slim`
- **Pre-loaded data:** Awards JSON files from `awards_data/` (Wikipedia scraping already done)
- **Dependencies:** Installed from `requirements.txt`
- **API keys:** Passed via environment variables at runtime

Usage

```
# Build container (includes pre-extracted awards data)
docker build -t literary-awards .

# Run pipeline (skips scraping, uses pre-loaded awards data)
docker run -e GOOGLE_BOOKS_API_KEY="..." -e NYT_BOOKS_API_KEY="..." \
  -v $(pwd)/output:/app/merged_data literary-awards ./run_pipeline.sh --skip-scraping
```

Benefits

- **Reproducibility from known data state:** Awards data frozen at extraction time
- Complete environment specification (Python version, libraries locked)
- Cross-platform (Linux, macOS, Windows)
- Skips ~10 minute Wikipedia scraping step

Appendix H: Project File Structure

```
cs598-fdc-literary-awards/
    README.md                          # Project overview and setup
    LICENSE                            # MIT + CC-BY-4.0 dual license
    requirements.txt                   # Python dependencies (pinned versions)
    Dockerfile                         # Container specification
    API_KEYS_REFERENCE.md              # API keys reference
    run_pipeline.sh                    # Master orchestration script
    run_scrapers.py                    # Scraper orchestration script
    get_awards_data.py                 # Awards data loader
    fetch_google_books.py              # Google Books API client
    fetch_openlibrary_books.py         # Open Library API client
    fetch_nyt_books.py                 # NYT Books API client
    merge_datasets.py                  # Data integration script
    scrapers/                          # Web scraping modules
        scraper_*.py                   # Award-specific scrapers
    schema/                            # JSON Schema definitions
        dataset-metadata.jsonld        # schema.org JSON-LD metadata
        award-entry.schema.json        # Award entry schema
```

```
    merged-book.schema.json         # Merged dataset schema
    *.schema.json                   # API-specific schemas
awards_data/                        # Pre-extracted awards (tracked in git)
    pulitzer_prize.json             # 76 entries
    national_book_award.json        # 130 entries
    booker_prize.json               # 375 entries
data/                               # Runtime data (not in git)
merged_data/                        # Integration outputs
    merged_literary_books.json      # PRIMARY OUTPUT (1,538 books)
    merge_report.json               # Merge statistics report
    unmatched_google_books.json     # Unmatched Google Books entries
    unmatched_openlibrary.json      # Unmatched Open Library entries
    unmatched_nyt_bestsellers.json  # Unmatched NYT entries
DATA_DICTIONARY.md                  # Field specifications + schema.org
CS598_FDC_Final_Report_Somnath_Saha.md   # Project report
```