Duplicates in the repository: remediation and reconciliation in three systems, including DataCite

Sunni Wong (Pratt Institute School of Information; Columbia University Libraries, Ask a Librarian Intern)
Frédéric Duby (Columbia University Libraries)
Esther Jackson (Columbia University Libraries)
Kathryn Pope, (Columbia University Libraries)

Takeaways from this presentation

- ➔ Everyone has dupes!
- ➔ What can you do to stop duplication before it happens?
- ➔ What can you do to remediate dupes efficiently, when they occur?
- ➔ Duplicates are their own ball of wax, so any remediation process you create is one that is outside of canonical workflows, and can't be fully automated.
- ➔ Duplicate remediation will teach you more about the interconnectedness of your systems.

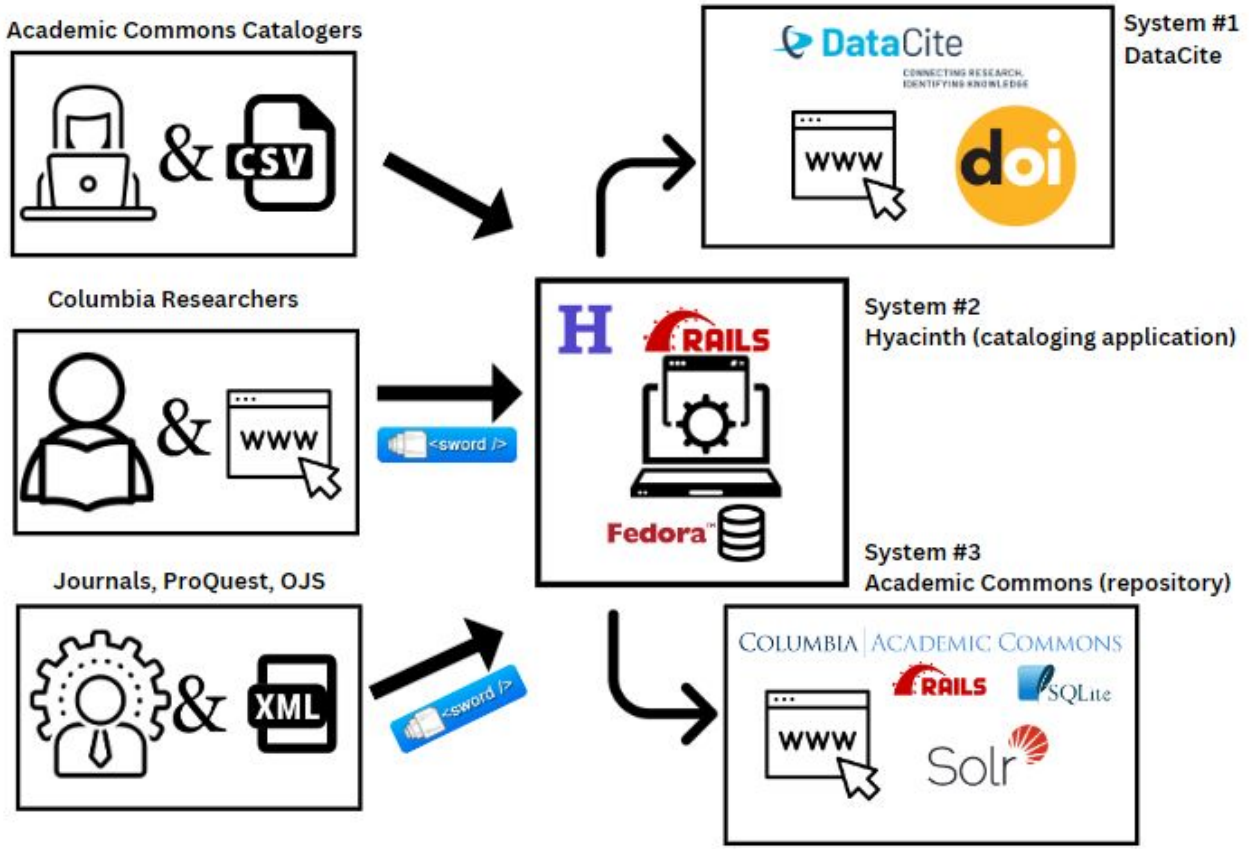
Where do duplicates come from?

- Self-upload form (Columbia authors)
- SWORD client batch-deposits (dissertations, articles)
    - Some scripts assume that if there is one failure in a batch, usually due to max size being exceeded, then all other deposit attempts within that batch are also failures, and will attempt to redeposit all
- Incorrect galleys from vendors
- Academic Commons catalogers (manual entry and bulk deposit process)

Systems relationship

Key terms:

- Hyacinth
- Academic Commons
- DataCite



Definitions

- **Item**: Parent metadata record for a work
- **Asset**: Child(ren) file(s) for a work (e.g. .pdf of an article, .csv dataset for an article, .mp4 of a podcast)
- **DOI**: Digital object identifier (unique), registered, in our case, through DataCite
    - In our repository, each published item and asset has a DOI

Our project

- The issue
- Project planning & preparation
- Process
  - Developing the dupes list (Google Sheets & Python script)
  - Remediating the duplicates
    - Academic Commons (Rails, Blacklight, Solr, MySQL)
      - Rake task
    - Hyacinth (Rails, Fedora)
      - Rake task
    - DataCite
      - Python script
- Outcome
- Lessons learned
- Future work

Issue

➢ Over the course of 15 years, duplicate items have been introduced into Columbia University's institutional repository, Academic Commons.

Removing duplicates is not a simple process:

- Manual records review necessary
  - Identifying duplicates requires more than title-field matching
  - Difficult to create general rule about which copy to keep
- Dupes (usually) include a parent/item and one or more child(ren)/asset(s)--but not always!
- Need to merge duplicate view and download usage stats w/ remaining copy stats
- Varied status of duplicates: published w/ DOIs, not published, etc.
- Need to re-direct DOI of duplicate to point to remaining copy

Legacy process of identifying duplicates

- Dupes identified by Libraries staff over time
- Added “!DNP--DUPLICATE record: ” in the titles
- Would not unpublish due to concerns of stats reports
- Tracked duplicates in shared Google Sheets

New Process

Planning and preparation

- Repository managers would create a final list of duplicate items and assets to be remediated.
- Intern Sunni Wong would use Python to help organize the required metadata.
- Repository developers would use this metadata to delete items and assets and remediate metadata in the following systems:
  - Repository application (Academic Commons)
  - Metadata management system (Hyacinth)
  - DOI registration service (DataCite)

Process 1: Revise the process of Identifying duplicates

- Review metadata using OpenRefine clustering
  - Discovered more duplicates!!!
- Review asset file checksums
  - Even more duplicates!
- Examine and select the best item to keep
  - General points of consideration:
    - Submission date
    - Metadata quality
    - Child assets quality
  - Items were assessed manually because there was no simple rubric to define which item to keep
- Continue to use Google Sheets to track duplicates as they are discovered, for later batch remediation

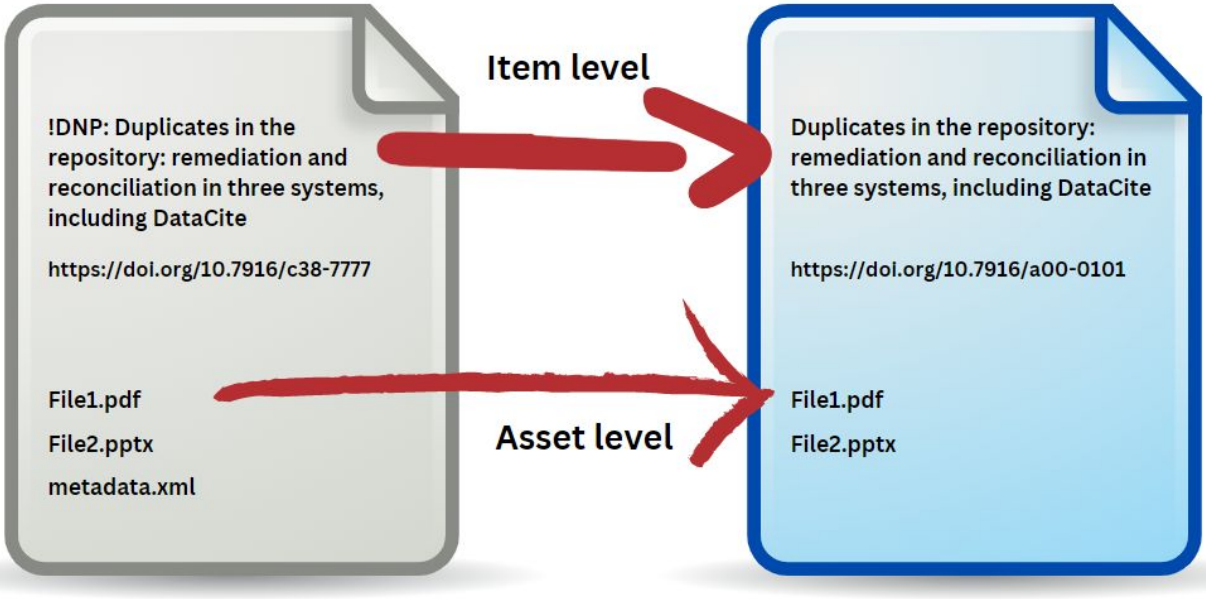
Duplicates Review Spreadsheet (CSV)

B	C	D	E	F	G	I	J	
YES dupe	Ignore	Further review	delete--PID	delete--DOI	delete--first_published	keep--PID	keep--DOI	OR Title 1 > Sort Portion
TRUE	FALSE	FALSE	ac:dncjsxkssw		2021-07-13T12:16:56Z	ac:b2rbnzs7nj	doi:10.7916/d8-3ben-8q24	!DNP--DUPLICATE RECORD: Explorir
TRUE	FALSE	FALSE	ac:qrfj6q578j		2021-07-13T12:15:32Z	ac:b2rbnzs7nj	doi:10.7916/d8-3ben-8q24	!DNP--DUPLICATE RECORD: Explorir
TRUE	FALSE	FALSE	ac:1jwstqjq65	doi:10.7916/d8-5hwh-e728	2021-07-13T12:15:05Z	ac:b2rbnzs7nj	doi:10.7916/d8-3ben-8q24	!DNP--DUPLICATE RECORD: Explorir
TRUE	FALSE	FALSE	ac:2547d7wm	doi:10.7916/d8-srhb-sg93	2021-07-13T12:14:59Z	ac:b2rbnzs7nj	doi:10.7916/d8-3ben-8q24	!DNP--DUPLICATE RECORD: Explorir

- PID = internal identifier
- DOI = DOI
- First Published & Title used for assessing which copy to keep and which to remove

Process 2: Mapping items & assets

1. Item level mapping
2. Look up child assets
3. Child level mapping



https://github.com/sunniw/ColU_AcademicCommons

```
# **Import the AC exported full dataset**

df= pd.read_csv('ac_export_data.csv', dtype='string')

# df.head()      # Sample data
# print(df.columns.tolist())     # Total 8092 columns
```
[14]

```
# **Import the PID list from "Duplicates in AC" to compare with the main list.**

currentACDupe = pd.read_csv('currentPIDList.csv')

# currentACPID_list.head()       # Sample data.
```
[15]

```
# Extract only the relevent columns from the full AC data to speed up process.

trimmedACData = df[['PID', '_doi', 'Digital Object Type > String Key', 'Title 1 > Sort Portio
trimmedACData.columns = ['PID', 'DOI', 'Object Type', 'Title', 'Parent PID', 'Filename', 'Ch
trimmedACData = trimmedACData.drop([0])     # Remove first row of element keys.

trimmedACData.head()        # Sample data
```
[16]

...

```
# Select rows that are marked as duplicates from the "Duplicate in AC" CSV.

currentACDupe = currentACDupe[currentACDupe['YES dupe'] == True].reset_index(drop=True)

# Create a list of duplicate PIDs.

currentDupePID = currentACDupe['delete--PID'].tolist()

currentACDupe.head()
```
[17]

Lists generated by Python script (1 & 2)

	PID	DOI	Title	Object Type	Parent PID to Keep	DOI to Map to
0	ac:05qfttdz2r	<NA>	mets.xml	asset	ac:x3ffbg79jr	https://academiccommons.columbia.edu/doi/10.79...
1	ac:08kprr4xht	doi:10.7916/D81G1ZR0	Making Cognitive Latent Variables Manifest: Di...	item	ac:193300	https://academiccommons.columbia.edu/doi/10.79...
2	ac:08kprr4xkn	doi:10.7916/d8-ppp7-1s08	!DNP DUPLICATE "Addis Ababa Bete (Home)": Cont...	item	ac:jq2bvq83gg	https://academiccommons.columbia.edu/doi/10.79...
3	ac:0gb5mkkwkn	<NA>	Opening Ceremony 2007 Photo only.pdf	asset	ac:h70rxwdbwx	https://academiccommons.columbia.edu/doi/10.79...
4	ac:0gb5mkkwmb	<NA>	mets.xml	asset	ac:05qfttdz2s	https://academiccommons.columbia.edu/doi/10.79...

Mapping duplicates to their retained equivalents for merging stats before removing from Hyacinth and AC

	Parent PID to Keep	Dupe Asset PID	Dupe Asset Filename	Keeping Asset PID	Keeping Asset Filename
0	ac:115911	ac:107680	WP_222.pdf	ac:115909	econ_0304_12.pdf
1	ac:123767	ac:188164	D._bajpai_indiamdgchallenge_2005_24.pdf	ac:123768	CGSDwp24.pdf
2	ac:123800	ac:188185	4._bajpai_outsourcing_2004_16.pdf	ac:123801	CGSDwp16.pdf
3	ac:124645	ac:127298	WP_287.pdf	ac:124646	WP_287.pdf
4	ac:125623	ac:134701	DesigningSensorsInsider.pdf	ac:125624	insider_threats.pdf

Asset level mapping that identifies the canonical, published asset of each item

Lists generated by Python script (3)

	DOI	PID	Object Type	DOI to Map to
0	doi:10.7916/D81G1ZR0	ac:08kprr4xht	item	https://academiccommons.columbia.edu/doi/10.79...
1	doi:10.7916/d8-ppp7-1s08	ac:08kprr4xkn	item	https://academiccommons.columbia.edu/doi/10.79...
2	doi:10.7916/D8VM4KQZ	ac:107680	asset	https://academiccommons.columbia.edu/doi/10.79...
3	doi:10.7916/D8HQ46FR	ac:107682	item	https://academiccommons.columbia.edu/doi/10.79...
4	doi:10.7916/D8VQ39FQ	ac:110114	asset	https://academiccommons.columbia.edu/doi/10.79...

Mapping items' DOI for the work on DataCite

A closer look at our systems

What is Academic Commons?

- “Provides open, persistent access to the scholarship produced by researchers at Columbia University, Barnard College, Jewish Theological Seminary, Teachers College, and Union Theological Seminary.”
- Part of a network of open scholarly resources

Academic Commons process: merging stats

- Items in Academic Commons have associated stats representing the number of record views and file downloads.
- Deleting the duplicate items from Academic Commons would entail losing the access stats associated with that item. Therefore, before the deletion, a ruby rake task is executed which merges the stats from the duplicate version into the stats for the canonical/retained version of the work.
- The input CSV for this rake task contains the PID of the duplicate version, as well as the PID for the canonical version.

duplicate_records.rake

38 lines (32 sloc) | 1.54 KB

```ruby
namespace :duplicate_records do
  desc 'Removes solr document'
  task :delete_solr_document, [:pid] => :environment do |t, args|
    rsolr = AcademicCommons::Utils.rsolr
    rsolr.delete_by_id(args[:pid])
    rsolr.commit
  end

  desc 'Merges statistics for a set of aggregator or asset'
  task :merge_stats, [:pid, :duplicate_pid] => :environment do |t, args|
    pid, duplicate_pid = args[:pid], args[:duplicate_pid]

    puts ""

    duplicate_document = ActiveFedora::SolrService.query("{!raw f=id}#{duplicate_pid}").first
    puts Rainbow("Duplicate pid (#{duplicate_pid})").yellow
    puts "active_fedora_model_ssi: #{duplicate_document['active_fedora_model_ssi']}"
    puts "number of stats: #{Statistic.where(identifier: duplicate_pid).count}"

    puts Rainbow("\nWill be merged with...\n").magenta

    document = ActiveFedora::SolrService.query("{!raw f=id}#{pid}").first
    puts Rainbow("Pid (#{pid}):").cyan
    puts "active_fedora_model_ssi: #{document['active_fedora_model_ssi']}"
    puts "number of stats: #{Statistic.where(identifier: pid).count}"

    puts Rainbow("\nAre you sure you want to merge these records' statistics? (y/n)").red
    input = STDIN.gets.strip
    if input == 'y'
      puts Rainbow("Merging statistics...").green
      Statistic.merge_stats(args[:pid], args[:duplicate_pid])
      puts "#{duplicate_pid} (duplicate) has #{Statistic.where(identifier: duplicate_pid).count} stats."
      puts "#{pid} has #{Statistic.where(identifier: pid).count} stats."
    else
      puts Rainbow("Statistics merge aborted").red
    end
  end
end
```

What is Hyacinth?

- Hyacinth is CUL's digital library metadata management and editing system. It was developed by the Libraries Digital Program Division, working with partners in other divisions of the Libraries.
- Hyacinth is a Rails application which uses Fedora as a repository to store assets.

Hyacinth processing

- In Hyacinth, items and the associated assets representing the duplicates are deleted/purged using the PID supplied in the input CSV.
- This entails removing all the metadata from the database associated with the application and the associated Fedora record for each item and asset

What is DataCite?

- DataCite is a leading global non-profit organisation that provides persistent identifiers (DOIs) for research data and other research outputs.

Updating DataCite DOIs

- A python batch script is used to update the metadata and state for the duplicate documents. The script uses the DataCite REST API (https://support.datacite.org/reference/introduction). Metadata is sent and received using the JSON format. The endpoint for the API is https://api.datacite.org.
- During development and testing of the script, the DataCite test API endpoint was used, https://api.test.datacite.org.
- Following updates are made to the DOIs for the duplicate documents, using the information supplied in the input CSV:
    - Change the state of the duplicate DOI to Registered
    - Update the URL for the duplicate DOI to the DataCite DOI url for the canonical document.
    - Add a note to the metadata for the duplicate DOI stating DOI is a duplicate.

Outcomes - numbers

- ➔ ~966 item/asset stats were merged into non-dupe items and preserved
- ➔ ~1374 duplicate items/assets were deleted
- ➔ ~1249 DOIs were remediated

Outcomes - workflows

- ➔ A **fifteen year project** was concluded
- ➔ A new workflow, along with robust cross-departmental documentation, was created for future duplicate remediation

Lessons learned - What can you do about dupes?

- Don’t be afraid of dupes–the sooner you get a sense of the size and scope of your duplicates problem, the sooner you can move forward.
- Speak with technical staff early and often when planning a large remediation project
- Incorporate a review process (metadata or checksum) into your cataloging process. Automate this if you can.
- Document everything!
- Interns are awesome 😊

Future work

- Checksum review at upload?
- Accept the dupes. Fix the dupes on a schedule.

Acknowledgements

Jack Donovan, Eric O'Hanlon, Jeremiah Mercurio, Benjamin Armintor, Brian Luna Lucero, Carla Galarza,

& all the Columbia University Libraries staff and students who introduced and tracked duplicates over the past 15 years!