Series of Tubes

Moving Subjects from MARC to ASpace

Ruth Tillman, Penn State, rkt6@psu.edu

Problem

- Inconsistent description practices between catalog and archival systems
- Better subjects on most MARC records

Historically, archival collections had MARC records created in cataloging and were separately described in special collections. Some finding aid records were based off MARC, but many were done separately.

Most access points, especially subjects, were better on the MARC records.

So we needed to figure out how to bring these over into ArchivesSpace and attach them to the correct resource records.

Goals

- Build spreadsheet of subjects/ArchivesSpace resource IDs
- Create necessary subjects in ArchivesSpace
- Connect subjects to ArchivesSpace resource records

I broke the problem down into three deliverables I needed. First, I needed to create a spreadsheet which paired subjects and the resource IDs of the ArchivesSpace objects to which they beloned.

I then needed to create the new subject records in ArchivesSpace (if they didn't already exist)

And finally, I needed to find a way to connect the two.

Tools

- ASpace MySQL Database
- MarcEdit
- OpenRefine
  - Particularly the “cell.cross” function.
- Python
  - ArchivesSnake and the ArchivesSpace API

I used the following tools--the ArchivesSpace MySQL database, to which I connected via MySQL Workbench; MarcEdit for getting data from the extract of MARC records; OpenRefine for connecting data, particularly the cell.cross function which lets you use it like a database; and Python scripts, especially the ArchivesSnake library to use the ArchivesSpace API.

photo by skrubu

I’ll walk through the process of getting data from point A to point B, a series of tubes.

Process Overview

---

1. Extracting MARC and ASpace Data
2. Pairing MARC Records/Subjects and ASpace Resources
3. Managing ASpace Resources and Subjects
4. Identifying Unique Subjects and Building Subject Records
5. Adding Subjects to Resources

This is an overview of the steps involved.

1a. Extracting MARC Data

Tool: MarcEdit

Product: CSV with the following fields

catkey	245	650 (pipe-separated, including subfields)	856u

First, I worked with our batch MARC team to get an extract of all records for archival collections. I then used MarcEdit to produce a CSV with the following fields -- the catkey, which is our local catalog identifier, the title (245), the subjects (650s), and the 856u or finding aid URL(s). When extracting the 650, I had to include subfields for later parsing by my subject-building script. I had MarcEdit use pipe characters as separators so that I could break them into new lines in OpenRefine.

1a. Extracting MARC Data

Sample data:

catkey	245	650 (pipe-separated, including subfields)	856u
a6937169	Pennsylvania State University, Creamery records, 1891-1984	$aCreameries$zPennsylvania$zUniversity Park$vArchives|$aDairy products industry$zPennsylvania$zUniversity Park$xHistory$y19th century$vSources|$aDairy products industry$zPennsylvania$zUniversity Park$xHistory$y20th century$vSources	https://libraries.psu.edu/findingaids/217.htm

This is a sample data extract. You'll note the pipe characters in the 650s. In the 856u, the 217 near the end is the old EAD_ID. This will become important later.

1b. Extracting ASpace Data

Tool: MySQL Database (MySQL Workbench)

Query: SELECT id, title, ead_id FROM pennstate.resource where title !="unspecified" && repo_id = "3" && publish = "1";

Product: CSV with the following fields

id	title	ead_id

Over in MySQL Workbench, I ran the query shown to get all published resources where the title wasn't "unspecified" -- which appeared to be our version of a blank title. I only got titles from repository 3, our live repository. I extracted the id, title, and ead_id. I'll refer to the main id as the ArchivesSpace or ASpace ID moving forward.

2. Pairing Records

---

1. Ideal -- pair on title, using Excel Fuzzy Matching
    a. Complication: Multiple records with the same title
2. Actual -- pair on EAD_ID / 856u filename.
    a. Complication: 856s are incomplete
3. Future -- pair on title but review duplicate titles
    a. Pair on titles
    b. Select unique titles for a batch
    c. Review non-unique titles for a second batch

In an idea world, it would be easy to pair MARC records with ArchivesSpace resource records just using the titles. Excel Fuzzy matching helps. However, there were some significant differences in titles and some items had multiple MARC records.

Instead, I paired the first batch on the ead_id which was the same code as the filename part of the finding aid URL. This did not capture all matches, but provided a group of several hundred on which to start.

Going forward, I'll work on all title matches which didn't have 856s but weren't duplicates and then review duplicates from the original title match.

2. Pairing MARC Records/Subjects with ASpace

Pairing by 856u

1. Derive data from 856us.
2. Upload spreadsheet of all records with an 856u to OpenRefine
3. Upload spreadsheet of all ASpace records (including ead_id) to OpenRefine
4. Use cell.cross to insert the ASpace id field into a new column in the MARC spreadsheet when ead_id & 856u_id match

In pairing records, I stripped all but the filename slug from the 856u field, which I then called the 856u_id. I added both projects to OpenRefine and used the cell cross function to perform the equivalent of a database key match. When the ead_id and 856u_id matched, I inserted the ASpace ID into a new column in the MARC extract spreadsheet.

2. Pairing MARC Records/Subjects with ASpace

Sample data:

aspaceID	catkey	650	856u_id
1406	a6937169	$aCreameries$zPennsylvania$zUniversity Park$vArchives|$aDairy products industry$zPennsylvania$zUniversity Park$xHistory$y19th century$vSources|$aDairy products industry$zPennsylvania$zUniversity Park$xHistory$y20th century$vSources	217

This is a sample of the resulting data. I've removed the title and will soon remove the 856u_id.

3. Managing ASpace Resources and Subjects

---

1. Remove all columns except ASpace ID + 650s.
2. In OpenRefine, split 650s by |
3. Fill down ASpace IDs

What I actually need to make the updates is just a spreadsheet with ASpace ID and the subjects which I should add. However, the subjects are still in string format and are pipe-separated. In OpenRefine, I can use edit to split multi-value cells in a column on a character, including pipe. Then in the ASpace ID column, I used the “fill down” option to add IDs to each row.

3. Managing ASpace Resources and Subjects

Sample data:

aspaceID	650
1406	$aCreameries$zPennsylvania$zUniversity Park$vArchives
1406	$aDairy products industry$zPennsylvania$zUniversity Park$xHistory$y19th century$vSources
1406	$aDairy products industry$zPennsylvania$zUniversity Park$xHistory$y20th century$vSources

This is an example of the previous data split and then filled-down. Now we have ID/value pairs.

3. Managing ASpace Resources and Subjects

---

4. In OpenRefine, sort by 650, **reorder permanently**, blank down 650s.

We then need to derive just the unique strings. By reordering and blanking down, we get a column which we can export/paste into a text document. After sorting, one *must* make the new order permanent in order for this to work.

3. Managing ASpace Resources and Subjects

Sample data:

aspaceID	650
1406	$aCreameries$zPennsylvania$zUniversity Park$vArchives
1491	$aCreameries$zPennsylvania$zUniversity Park$vArchives
2804	$aCreameries$zPennsylvania$zUniversity Park$vArchives

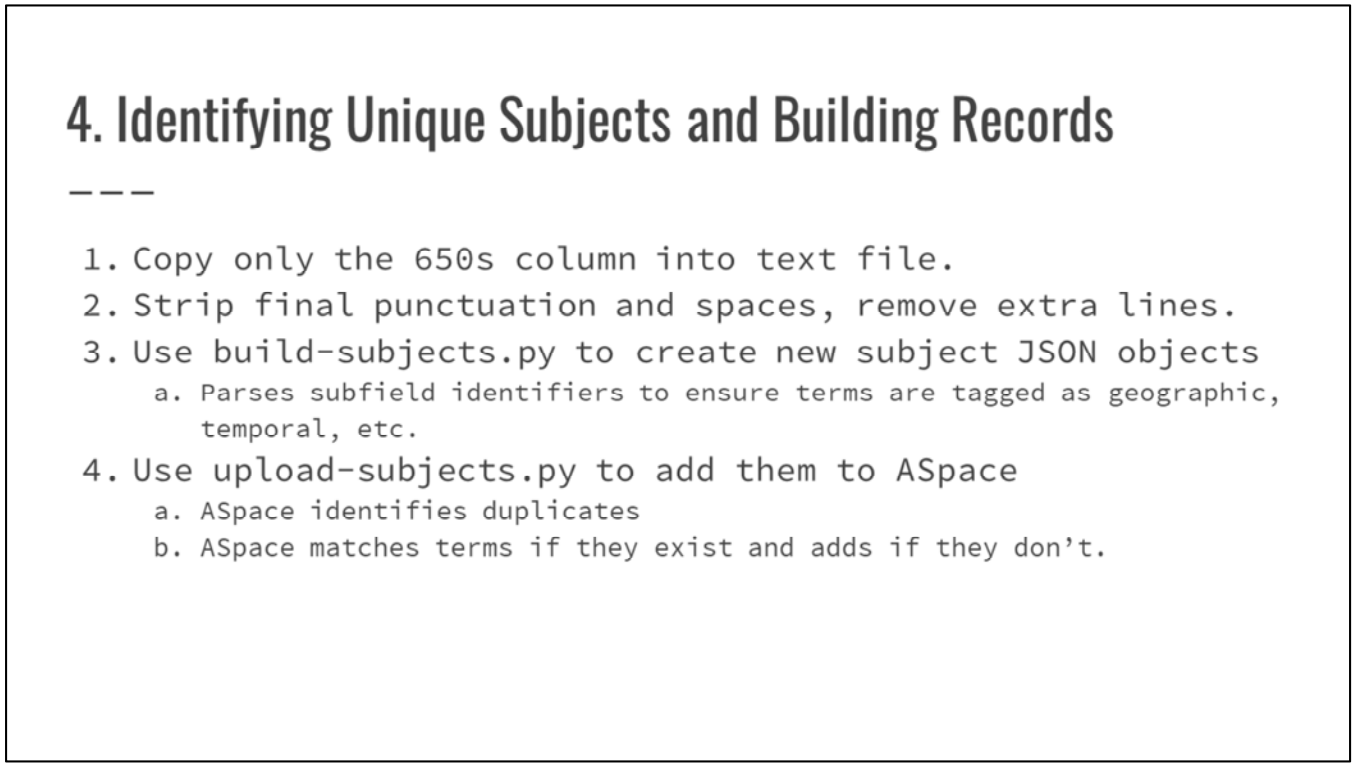
Ordered by subject.

3. Managing ASpace Resources and Subjects

Sample data:

aspaceID	650
1406	$aCreameries$zPennsylvania$zUniversity Park$vArchives
1491	
2804	

And blanked down.



I then copied only the 650s into a text file. I stripped final punctuations and removed extra linebreaks. The script in my linked repository, build-subjects.py, takes each line, breaks it apart by subfield, and creates JSON objects which can be uploaded to ArchivesSpace.

When uploading to ArchivesSpace, it determines the subject isn't a duplicate and adds term IDs. This means I didn't have to figure out what the ID was for the existing geographic term, Pennsylvania. The system handled that part.

4. Identifying Unique Subjects and Building Records

Sample data (in a text file)

```
$aCreameries$zPennsylvania$zUniversity Park$vArchives

$aDairy products industry$zPennsylvania$zUniversity
Park$xHistory$y19th century$vSources

$aDairy products industry$zPennsylvania$zUniversity
Park$xHistory$y20th century$vSources
```

This is a sample of what the text file looked like.

5. Adding Subjects to Resources

---

1. Extract Subject IDs and full subject strings from Log into spreadsheet
    a. For dupes in log, use conflicting record, e.g. "conflicting_record": ["/subjects/1112"]}
2. In OpenRefine, cell cross on full subject strings and insert Subject IDs into spreadsheet.
3. Delete subject string column

Using the logged output of the script, which involves the API response, I extracted the newly-made subject IDs and full strings, as well as the existing subject IDs identified for a few of them. As shown above, it uses “conflicting_record” in the JSON to return the subject URI.

I made this into a spreadsheet of subject IDs and strings.

In OpenRefine, I again used cell cross. This time, I matched full subject string content and inserted the IDs into the spreadsheet.

I then deleted the subject string column. Now it was just ID pairings.

5. Adding Subjects to Resources

Sample data:

aspaceID	subjectId
1406	2938
1491	
2804	
1406	2939

This is an example of the resulting data.

5. Adding Subjects to Resources

---

4. Fill down IDs

5. Sort by ASpace ID, **reorder permanently.**

I then used fill down to populate IDs in each pairing. I sorted by the ASpace ID and reordered permanently. This took us back to an earlier step, but now with better data.

5. Adding Subjects to Resources

Sample data:

aspaceID	subjectId
1406	2938
1406	2939
1406	2940
1491	2938

Sample resulting data.

5. Adding Subjects to Resources

---

6. Blank down ASpace IDs

7. Join multi-valued Cells

Now we need just one ASpace ID and a second column with all the subject IDs. I blanked down the ASpace ID column, then edited the subject ID column to join multi-valued cells.

5. Adding Subjects to Resources

Sample data:

aspaceID	subjectId
1406	2938|2939|2940
1491	2938
2804	2938
3201	2941|3142

Again, I used the pipe character.

5. Adding Subjects to Resources

---

8. Use update-resources.py to download and update resources

9. Use upload-updated-resources.py to upload to the system

The next script I wrote, update-resources.py, downloads resource records and updates them. Then, after some review, I used upload-updated-resource.py to upload them with the API.

5. Adding Subjects to Resources

Dive in to update-resources.py:

1. Downloads resource record
2. For each subject identifier
    a. ensures a relationship doesn't already exist
    b. if no relationship, appends to relationship to "subjects"
       (creates "subjects" if doesn't exist)

Here's a quick look into the steps of updating resources. First, one downloads the appropriate record, using its ID. Then, for each subject identifier in the right-hand column, the script -- ensures there isn't already a relationship between the two. If there isn't, it appends the relationship to the subjects section (which it creates, if one doesn't exist). It's slightly more complex in that it also preserves order of initial subjects, if any. It attaches subjects in the order indicated by the spreadsheet, which is likely unstructured. If ordering of subjects is important to you, this will require intervention. However, at the highest level it often doesn't (at least in a post Rule-of-3 era)

Do It Yourself

Documented scripts can be found here:
https://github.com/ruthtillman/ASpaceASnake

Forthcoming blog post or practice article with more detailed instructions and sample code for OpenRefine.

And contact me: rkt6@psu.edu