BANG! Scripts

We used various scripts to analyze different data sources and set up code for viewing Fuseki data.

LOC Hub Analysis

- We used client-side AJAX queries to retrieve the first 10,000 hubs from LOC and then navigate to related works and instances to analyze how many LOC Hubs provide two or more instances with ISBNs or LCCns.
    - https://github.com/LD4P/blacklight-cornell/blob/bang/app/assets/javascripts/bang/evalHub.js
        - This code looks at how many hub to hub relationships provide LCCNs or ISBNs. To avoid throttling issues, the code queried 500 hubs at a time. When bringing up the page that ran the code, we would set the starting hub number, effectively paging through the first 10,000 hubs returned from LOC.
        - Hubs were retrieved using this call "https://id.loc.gov/search/?q=cs:http://id.loc.gov/resources/hubs&count=" + this.sampleSize + start + "&format=json" where the sample size and starting hub number could be specified.
        - --parse button--
        - Related view: https://github.com/LD4P/blacklight-cornell/blob/bang/app/views/bang/eval_hubs/index.erb
    - https://github.com/LD4P/blacklight-cornell/blob/bang/app/assets/javascripts/bang/evalHubAggregation.js
        - This code retrieves unique ISBNs for every hub that has more than 1 work. This code also uses the same sample size and paging approach as the code above.
        - Related view: https://github.com/LD4P/blacklight-cornell/blob/bang/app/views/bang/eval_hubs/same_hub.erb
    - https://github.com/LD4P/blacklight-cornell/blob/bang/app/controllers/bang/eval_hubs_controller.rb
- We wrote scripts to further analyze these groupings of hubs to see how many catalog matches we could get.
    - LCCN analysis
        - Finding catalog matches for LCCN sets grouped under LOC Hubs
            - This file (HubSetsLccn.csv) lists an LOC Hub on each line followed by a list of LCCNs from instances that fall under that hub.
            - A script (processlccn.rb) reads in this file and then generates the file (lccnhubonlyfirst) which lists the LCCN rows that matched at least two catalog items, and then ends with a summary. (The output says "ISBN" but is in fact "LCCN" because the same code was copied/used the ISBN analysis).
        - Finding catalog matches for LCCN sets grouped under LOC Hub to Hub relationships
            - Each line in the file (prophublccnsets.csv) lists the name of the relationship (e.g. "hasTranslation") that links two different hubs, followed by the LCCNs that fall under those hubs.
            - A script (processrellccn.rb) reads in this file and then generates the file (lcchubrels) which starts with a list of the property and LCCN groups that resulted in at least two catalog matches (e.g. "hasTranslation : 2017328875,92911176,93910013") followed by a summary of the total number of rows and LCCNs in the original file and the number of matching rows/LCCNs. In addition, the file also lists those hub relationship and LCCN groupings from the original CSV file that resulted in exactly one match in the catalog.
    - ISBN analysis
        - Finding catalog matches for ISBN sets grouped under LOC Hubs
            - The script (processcsv.rb) analyzes the file (HubSets.csv), which lists LOC Hubs with the groups of ISBNs that fall under each hub, and generates a file (tenthousandresults). This resulting file first lists the sets of ISBNs from the original CSV where each set has at least two catalog matches. The file ends with a summary of the total number of rows processed from the original file and the rows that matched at least two catalog items (i.e. ISBN sets).
        - Finding catalog matches for ISBN sets grouped under LOC Hub to Hub relationships
            - The script (processrelcsv.rb) analyzes the file (prophubsets.csv) . This CSV file (you can sense a pattern now) which lists the property connecting two LOC Hubs followed by a list of ISBNs that fall under the two hubs related by this property. The analysis results in the file (updateHubRelResults) which lists the relationship and ISBN groups that result in at least two catalog matches. The file ends with a summary of total rows processed from the original file and the number of rows which resulted in two catalog matches.

PCC data analysis

- Analysis of ISBNs aggregated under the same Opus
    - Our first pass at queries had taken too long, so we broke the process up into separate portions.
        - First, we queried the PCC data using Dave's Fuseki server (or a copy of the data on our own Fuseki server) to retrieve a list of all Opera that had at least two works with instances with ISBNs. The query we used is captured here. Executing this query resulted in the following list of Opera URIs.
        - This script takes the list of Opera URIs and executes SPARQL queries to retrieve the ISBNs of any instances that correspond to different work URIs aggregated by that Opus. Running the script results in a file where each line has sets of ISBNs corresponding to an opus (Note that the script has the Fuseki SPARQL URL not included so running the script would require replace that part of the code with the Fuseki SPARQL URL you wish to query.)
        - Another script then takes the file with the ISBN groups to check which of these groups results in at least two catalog matches. The script outputs the ISBN groups that result in matches long with a summary (i.e. total number of rows, total number of matches, etc.), the ISBN groups that listed in only one match, and those that didn't result in a match at all. The output is captured here.
- Analysis of LCCNs aggregated under the same Opus
    - Similar to our ISBN analysis, we first queried the PCC data to generate a list of all Opera that have at least two works with instances with LCCNs. The query is captured here and the results here.
    - The same script used above to execute SPARQL queries is also used for querying this list of Opera to get the LCCNs grouped under each opus. The line used for LCCNs is commented out at the bottom of the code. For LCCNs, this script output the following file where each line has a set of LCCNs grouped under the same Opus.
    - This script analyzes these LCCN groups to see which have more than one catalog match and lists the groups that resulted in a match along with a summary of total rows processed and the number of matches. The output file is here and also contains the rows that resulted in only one catalog match and those that didn't result in any matches.
- Analysis of work to work relationships
    - It is important to note that I've had to reverse engineer some of these connections between files and scripts for this analysis based on the information made available. Time permitting, I may re-run the analyses to ensure that I have the correct process. In the meantime, this documentation should serve as an adequate reference.

    - Because a comprehensive query trying to retrieve just those works which had a relationship and which each work had an instance with an ISBN was taking too long, we broke the process into the following parts:
        - This file contains the query we used to retrieve those works that have relationships with each other and where the first work has an instance with an ISBN.
        - This file contains (what I believe to be) results of running the query above against our local Fuseki server.
        - This script then iterates through the works and predicates and sees if there are ISBNs that can be retrieved for these works. It outputs the information delimited by "|". An output file matching the format of results from this script is here.
        - This script goes through this file where each line has sets of ISBNs and the predicate which relates the works for the instances which have these ISBNs. The script checks how many rows have matches for both the left side of ISBNs and the right side. For exampe, if the row has [ISBN1, ISBN2]| predicate | ISBN3, ISBN4, this means that the first work has instance(s) with ISBN1 and ISBN2 values for ISBN identifiers, and the second work has ISBN3 and ISBN4 values for ISBN identifers, and the two works are related using the predicate listed in between. The script then checks if there are any catalog matches for ISBN1 and ISBN2, and also for ISBN3 and ISBN4, and outputs this row as a match only if there are catalog matches for both sets of ISBNs. The output is captured here which shows 184 rows that have matched in this way.

POD data analysis

- We wanted to analyze the POD (Platform for Open Data) transformation provided by Jim Hahn (University of Pennsylvania) to see if we could retrieve matches for our set of ISBNs that fell under the same LOC Hubs and that only had a single match in the Cornell catalog. This transformation provided sets of CSV files per institution, where the headers represented MARC fields and the rows contained values per MARC record mapped to those fields.
    - This file contains the list of ISBN sets that resulted in a single match in the Cornell catalog.
    - This script reads in this file and compiles the ISBNs that occur in the file. The script then reads in the transformed CSVs which contain POD data mapped to MARC fields and values. If the script finds any of the target set of ISBNS represented within an institution's transformed data, the script then outputs the transformed rows which match any of these ISBNs. The results of this script are included here, with matching rows for Brown, Chicago, Columbia, Dartmouth, Duke, Harvard, Penn, and Stanford.
        - A separate script retrieves the Cornell catalog record information for matching ISBNs, resulting in this file which lists the original set of ISBNs we were querying against, followed by the catalog id and title of the record, followed by the ISBNs for that item captured in the record itself.
    - Using the results from the previous step, this script reads in the information for MARC records matching the original set of ISBNs we are querying against, and uploads information to a Solr index we set up specifically to allow us to store and search across these multiple records. We also add the institution information to the record, to specify where the data is coming from. If there are records that are not added due to insufficient information, the output identifies those records. In this case, three records from Brown did not have 001 fields and were not added to our Solr index.
    - This script uses the original file with ISBN sets that result in a single Cornell catalog match, and queries both the LD4P3 copy of the Cornell Solr index and the POD index set up for this analysis to find which catalog records across these institutions matches these ISBNs sets. The output generated is in the form of an HTML page available here.

Fuseki UI

- We set up this lightweight code to enable the viewing of data accessible through a Fuseki SPARQL endpoint.
    - To try out this UI, please download the contents of this directory to your local machine.
    - The Fuseki SPARQL endpoint needs to be specified in the js/config.js file. To create this file, copy over this example file to the js directory, and set the SPARQL endpoint URL as the value of the property "fusekiURL" which is currently commented out.
    - You can now click on and open up the view file in your browser to review the classes and predicates present in your data. Clicking on the class and predicate links will show example entities and statements respectively. More details and screenshots can be found in this report.
    - If you want, you can see a set of random statements extracted using the "feeling lucky" page.
- We also started some preliminary work to visualize hubs and relationships here but this code probably requires a lot more review and work.