BANG! Lessons Learned

Usability testing summary

- Usability testing revealed that, in general, participants were able to find work to work relationships quite easily through the prototype. Some of these same relationships, such as translations or sequels, were more difficult to retrieve directly in the production instance because the MARC lacks this information and/or how this information is currently utilized in production. In addition, people expected the "other forms" of this work feature in the production system to lead to related works such as translations, sequels, and adaptations with different facet values enabling filtering by language and format. It is also important to note that further work and analysis should be done to review which work to work relationships users would find useful. Although some participants indicated they would find some of these relationships useful, they do not necessarily look for many of these relationships currently in the catalog.

Development and technology lessons

- The prototype we built saved the work URI in a MARC field and then retrieved that URI and related entities from Sinopia using client-side calls to the Sinopia API. If we want to implement a similar solution in production, we'd need to review which MARC field to use to hold the URI, and we'd perhaps need to rely on a data source such as Wikidata to retrieve work to work relationships where we also had the necessary identifiers, such as LCCNs, ISBNs, or OCLC work ids, to retrieve the appropriate library catalog items.
- The work page was laid out using a design similar to the Bento Box layout for the library search which retrieves results from across different library data sources. If we want to implement a similar solution in production, we would need to come up with a more flexible or dynamic column by column layout since not all relationships will be available for a given work and want the sections to wrap column to column.
- The scripts needed for our data analysis (described below) required multiple steps in the workflow and sometimes had to be re-written since our original SPARQL queries took too long against large datasets like the ShareVDE PCC data. Also, the scripts had the server addresses hard-coded, but the versions we will upload to GitHub will have the specific addresses commented out. Any future version of such scripts should use environment variables to store the server addresses. Additionally, we should review options for linking together multiple scripts in a workflow while also being able to recover gracefully from errors in any part of the process.

Data lessons

- As part of this phase, we did an analysis of various data sources to understand which work to work relationships and aggregations in these data sources may help the discovery of related items in the catalog, This analysis led to this report. Overall, we found there were opportunities for using work aggregations, such as those provided by LOC Hubs and ShareVDE Opera, to retrieve clusters of related library catalog items. Hub and Opera can provide some useful clustering of items in the catalog, but in rare occasions is it a complete cluster. Based on what we know from the OCLC work clusters there is room for improvement when reconciling Hubs, Opera, Works and Instances, but we also found examples of successful BIBFRAME clustering that we didn't see in the OCLC Works data. We found some relationships between works, but the work to work relationships that lead to Instances in our catalog were limited in number and variety. In addition, Wikidata and IMDB both provided possibilities for retrieving additional contextual and descriptive information for works. We also used LOC Hub aggregations to provide us cross-institutional matches using an older version of POD (Platform for Open Data) data.
- One of the challenges we encountered in our analysis of large linked datasets was a quick way to understand what the data afforded, i.e. which classes and predicates were represented and how often these predicates were used. To help us with our analysis, we generated a simple lightweight interface for viewing the classes, predicates, and statements in a dataset to begin navigating and understanding the data in a dataset provided through a Fuseki SPARQL endpoint.