Qualitative and Quantitative Methods in Libraries (QQML) 6: 355-370, 2017

Identity and Access Management for Libraries

Qiang Jin[1] and Deren Kudeki[2]
[1]Senior Coordinating Cataloger, and Authority Control Team Leader, University Library, University of Illinois at Urbana-Champaign, USA
[2]Visiting Research Programmer, School of Information Science, University of Illinois at Urbana-Champaign, USA

Abstract. Linked open data will change libraries in a dramatic way. It will redesign metadata, and display metadata on the web. In order to prepare for linked open data, libraries will gradually transition from authority control creating text strings to identity and access management using identifiers to select a single identity. The key step for moving into the linked open data for identity and access management for libraries is to correct thousands of incorrect names and subject access points in our online catalogs. This article describes a case study of the process of cleaning up unauthorized access points for personal and corporate names in the University of Illinois Library online catalog. The authors hope that this article will help readers of other libraries prepare for linked open data environment.

Keywords Authority maintenance, controlled vocabularies, identity and access management, linked open data

1. Introduction

Linked Data is to use the Web to connect data, information, and knowledge on the Semantic Web using URIs and RDF.(1) Linked Data provides library users with searching a vast range of local and remote content through a single point of entry across a comprehensive index into a library's collection. Authority control is the area of Linked Data transition that has caused the most concern. (2) It is critical when we group works by authors with URIs in the linked data environment. Unfortunately, thousands of incorrect personal names, corporate names, and subjects in our online catalogs hinder users from finding library resources.

Authority control is the process of selecting one form of name or title from among available choices and recording it, the alternatives, and the data sources used in the process. It is essential for effective retrieval of resources. It provides consistency in the form of access points used to identify persons, families,

Received: 21.5.2017 Accepted: 2.9.2017 ISSN 2241-1925

corporate bodies, and works. (3) Authority control is central to the organization for information.

Authority control has gone through many changes over the last hundred years or so. One major new concept on authority control in libraries emerged in 2010 after IFLA created Functional Requirements for Authority Data (FRAD) A Conceptual Model (4), and Functional Requirements for Subject Authority Data (FRSAD) A Conceptual Model (5). These two models analyze entities person, family, corporate body, work, expression, manifestation, item, concept, object, event, place and their relationships. They describe entities of highest significance, attributes of each entity, and relationships among entities in regard to user needs. FRAD helps catalogers rethink about how catalogs should function, and establish standards. In 2010, Resource Description & Access (RDA)(6), a new international cataloging code was released adopting FRAD, and FRSAD.

Two major new concepts emerged in libraries in recent years. One is the creation of Bibliographic Framework Initiative (BIBFRAME), and the other is Schema.org. In 2012, the Library of Congress (LC) released a BIBFRAME model, a linked data alternative to MARC developed by Zepheira, a data management company. BIBFRAME is expressed in Resource Description Framework (RDF). (7) It serves as a general model for expressing and connecting bibliographic data. It is set to replace MARC 21 standards, and to use linked data principles to make library data discoverable on the Web while preserving a robust data exchange that supports resource sharing and resources discovery on the Web. In 2016, LC put out BIBFRAME model and vocabulary 2.0. The other major new concept is Schema.org, which is an initiative launched in 2011 by Bing, Google and Yahoo to create and support a common set of schemas for structured data markup on web pages. In 2012, OCLC took the first step toward adding linked data to WorldCat by appending Schema.org descriptive markup to WorldCat.org pages, making rich library bibliographic and authority data available on the Web. (8)

2. Background Information

The University of Illinois at Urbana-Champaign in the United States includes about 44,087 undergraduate and graduate students, and 2,548 faculty. The University of Illinois at Urbana-Champaign (UIUC) Library is one of the largest libraries in North America. Its online catalog holds more than thirteen million volumes, 24 million items and materials in all formats, languages, and subjects, including 9 million microforms, 120,000 serials, 148,000 audio-recordings, over 930,000 audiovisual materials, over 280,000 electronic books, 12,000 films, and 650,000 maps. (9) Although many large research libraries routinely do authority maintenance work, the UIUC Library has never done any systematic authority work for the last several decades. In order to prepare to move to the linked data environment, in 2015, the UIUC Library decided to do authority maintenance work locally due to limited budget to correct an estimated over one million incorrect personal names, corporate names, geographic names, series titles and the Library of Congress subject headings in our online catalog. Because of

budget cuts for the last several years, the UIUC Library staff cutbacks and the expanding need for professional librarians to work on digital collections, dealing with controlled vocabularies has become more stringent. Cleaning controlled vocabularies are clearly critical to the success of linked data since they are the basis for the URIs that create linkages. The goal is to provide enhanced discovery of library data, bringing together the comprehensive collections of content with indexes for deeper searches with millions of unique descriptive data components for books, images, microforms, etc.
In 2015, a small team was formed at the UIUC Library including one tenured librarian, one academic hourly, and two graduate assistants, and started working on authority maintenance work. The team discussed and decided to begin with fixing personal and corporate names. Even though the Library has over 13 million volumes, there are actually around 8 million bibliographic records in our online catalog. The team ran reports using SQL queries (10) to find “see” references for personal and corporate names in our bibliographic records because a major part of our problem with our personal and corporate names belong to this category. We separated the results of the queries into csv files holding up to 5,000 broken names each to run through the script we created locally. We solved various issues with the script before we were able to run all the files successfully. For each “see” personal or corporate name in our bibliographic record, the script looks for an authorized access point in the Library of Congress Authority File, and WorldCat trying to find matches. If a “see” reference finds an authorized access point and WorldCat also lists that access point, we consider a match. If no good access point is found or WorldCat does not list the access point, we save that name for human intervention in the future. Out of nearly 8 million bibliographic records in our online catalog, we corrected around 300,000 personal and corporate names successfully by machine, but we still have over 100,000 personal and corporate names that need to have people go over them one at a time checking the Library of Congress Authority File and correct them in our online catalog. Our precaution is necessary because our catalog is comprehensive.

3. Literature Review

Many experts in the cataloging field have stated the importance of authority control for decades. Michael Gorman in his paper indicates that authority control is central and vital to the activities we call cataloging. (11) Tillett says that authority control is necessary for meeting the catalog’s objectives of enabling users to find the works of an author and to collocate all works of a personal or corporate body. (12) Hillman, Marker, and Brady mention that the basic goals of a controlled vocabulary are to “eliminate or reduce ambiguity; control the use of synonyms; establish formal relationships among terms and test and validate terms. (13)
For decades, many libraries have either done in-house or have hired commercial vendors for their authority maintenance work. Some libraries have done in-house authority maintenance work due to their small scales of catalogs, or their budgets. For example, the Wichita State University Libraries did authority

maintenance work locally. They designed their workflow, and redesigned their cataloging staff structure for authority maintenance work. At the end of the project, they felt that they needed continuing support of their library administration's commitment to allocate more staff in order to continue their authority maintenance work. (14) Commercial vendors usually offer several types of authority control services including authority work after cataloging bibliographic records, retrospective cleanup to supply their authority control after, or as part of a conversion project, and ongoing authority control for libraries. (15) Not only vendors help libraries do authority work for their print collections for decades, they have started to do authority work for library digital collections in non-MARC. In 2013, Backstage Library Works did authority work for the University of Utah's Library digital collection in non-MARC. Their project demonstrates that it is possible to complete major updates to records in order to bring them in line with the authorized terms in commonly used controlled vocabularies without a large of amount of manual work. (16)

4. Initial Resources

The goal of this project is to replace variant name access points in our bibliographic records with their authorized form. We are able to detect variant name access points through an SQL query that selects name access points that have been labeled as "s" in our database, which means the access point is a "see" reference with linked bibliographic records. While this provides a large number of access points to fix, it is worth noting that only unauthorized name access points that have been labeled as a "see" reference are being addressed by this process. If a name access point is not authorized and not labeled as a "see" reference, it will be ignored entirely. The reason we chose "see" reference with linked bibliographic records was because the big percentage of our incorrect personal names, corporate names, and subject headings belong to this category that lists name access points from our catalog that match references in authority records based on the text string only. The query we used is from Consortium of Academic and Research Libraries in Illinois: CARLI since we are part of the consortium, which consists 134 member libraries in Illinois.

This query was used to determine the scope of this project, by running on 8 blocks of 1,000 bibliographic records to sample the more than 8 million bibliographic records in our online catalog. We found that there were 1,251 "see" references across the 8,000 bibliographic records examined. Using this rate, we estimate that there should be roughly 1 million "see" references in our database, all of which need to be examined for this project. There are five different kinds of access points, each of which may have its own intricacies for finding the correct authorized form. The table below shows what share of the "see" references each group contains.

Group	Share of "see" References
Name-title	3.6%
Subject	45.7%
Title	3.7%
Name (corporate)	22.1%
Name (personal)	25%

Because of the large amount of data involved, and the special considerations needed for each group, the team decided to develop our tools and processes for authority maintenance around one specific group, while also building a general structure that can be used in the future when the other groups are given specific attention. We chose to focus our efforts on fixing personal names, because the specific considerations needed for finding the appropriate authorized access point for any given problematic personal name access points are relatively simple. Also, since personal names make up about a quarter of the problematic access points we can detect, it's easy to produce a large sample to work on, and fixing personal names fixes a good portion of all the problematic access points. Once we finished development on fixing personal names, we were able to expand to fixing corporate names relatively quickly thanks to the similarities between personal and corporate names, and a code structure built to support the addition of modules for fixing the other groups.
We chose to develop our tools in Python due to the large number of third party libraries available. Thanks to third party libraries, we are able to easily establish Z39.50 connections and convert MARC-8 records into unicode. We chose to run our queries in SQL Developer because of its speed.

5. Approach

The general process we developed to update unauthorized name access points is splitting into three distinct steps. The first step is querying our database for the unauthorized name access points in our bibliographic records. The second step is processing the results of that query. The final step is uploading the changes found in the processing. The specific implementation of this process for personal and corporate names is as follows:

5.1. The Query

We run a query in SQL Developer over a given range that returns a list of unauthorized personal and corporate names in bibliographic records in that range that are recognized as "see" variants, rather than an authorized name. For each of these problematic names, the query returns several important pieces of information. It gives us the unauthorized name, complete with all associated subfields. It gives us the bibliographic id (BIBID) number, which is the unique identifying number of the bibliographic record containing the name in our database. And finally, it gives us the OCLC number of that same bibliographic record. (17)

5.2. Processing the Query Results

The query results are run through a Python script that searches for the authorized name that best fits the problematic name from the record. If an authorized name cannot be found, the problematic name is added to a list of other unresolved names that are meant to be reviewed by humans. If an authorized name can be found, the correction is applied to the bibliographic record, which is added to a master collection of corrected records.

To begin the processing of the query results, the problematic names are all read into the script, and grouped by BIBID. Each full problematic record is then retrieved from the database, one at a time, using a Z39.50 request. Each problematic name from the query is then matched with a personal or corporate name field in the bibliographic record. This match is made by calculating the Levenshtein distance between each name from the query, and each name in the record, and associating the pairs of names with the smallest calculated difference.

Once all the problematic names have been found in the record, each name is processed individually to find the authority record that best matches it. The first step of this process is to call a web application program interface (API) with the selected problematic name. For personal names, the Virtual International Authority File (VIAF) AutoSugest API is called, and for corporate names the VIAF SRU Search API is called. The API returns a list of suggested authorities in VIAF, which is then reduced to a list of authorities that are listed as either personal or corporate names and have a Library of Congress Control Number.

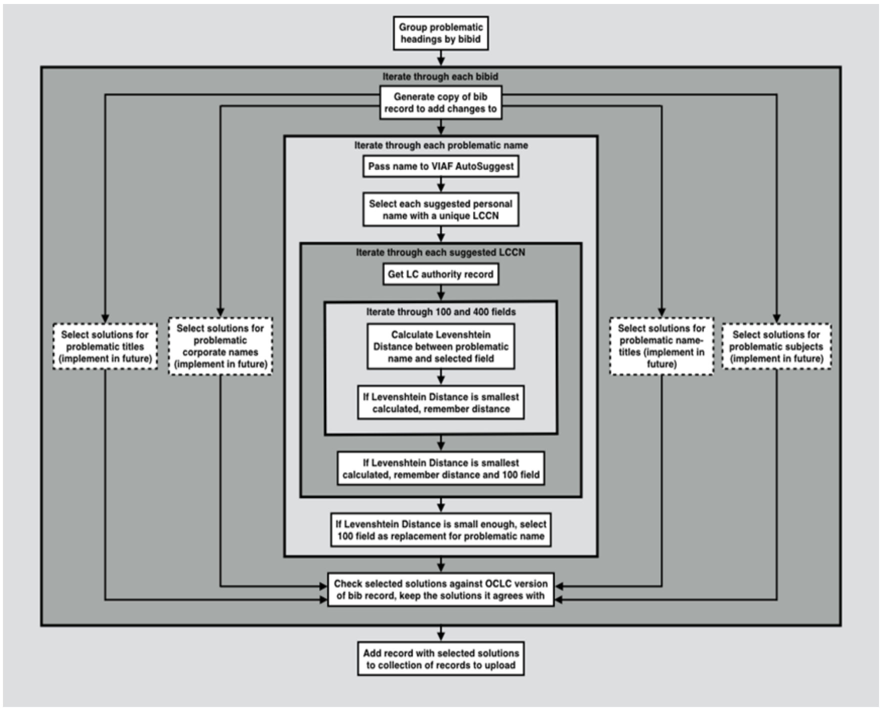
The list of Library of Congress Control Number (LCCN) is used to retrieve the Library of Congress (LC) authority record for each personal name through a series of Z39.50 queries. For each authority record, the Levenshtein distance is calculated between the problematic name, and all the versions of the name listed in the record. The smallest Levenshtein distance is found across all suggested authority records, and if that Levenshtein distance is small enough, the authorized name in that record is selected as the solution for the problematic name being examined. If the smallest Levenshtein distance found is not small enough, the problematic name, along with the authorized name with the smallest Levenshtein distance, is placed in a list of names that should be assessed by a human being.

Once the best guess has been selected from VIAF's list of authority records, the selection needs to be independently verified, because some of the suggestions that come out of this process are incorrect, but very similar to the problematic name in question. To do this, the OCLC number from the bibliographic record is used to retrieve OCLC's version of the bibliographic record. Each of the names in the OCLC record is compared to the solution that has been selected. If any of the names in the OCLC record contains an exact match for all of the information in the selected name that is considered independent confirmation. If the OCLC record fails to confirm the selected name, the problematic personal name along

with our selection is placed in a list of names that should be assessed by a human being.
Otherwise, the name selected by VIAF API and the Levenshtein distance calculation has been confirmed by the OCLC record, and is now considered safe to upload to the database as a correction. Once a name is to be uploaded, the problematic name is removed from the bibliographic record that was retrieved at the beginning of the process, and replaced with the authorized name that has been selected. Once all the problematic names in a record have been processed, if any of them have been replaced, the updated record is written to a collection of updated records. When the script has finished running this collection of records needs to be uploaded to the database to apply the changes.

5.3. Uploading the Changes

We are able to apply the changes to our bibliographic records one at a time by importing the collection of records that have been updated into the Voyager Client and manually overwriting the each existing record with its revised version. Since there are around 300,000 bibliographic records need to be updated, we are now waiting to talk to system services people in our consortium, and ask them to upload these changes in bulk. Our authority maintenance work will also help other 133 academic and research libraries in our consortium.


Group problematic headings by bibid
Iterate through each bibid
Generate copy of bib record to add changes to
Iterate through each problematic name
Pass name to VIAF AutoSuggest
Select each suggested personal name with a unique LCCN
Iterate through each suggested LCCN
Get LC authority record
Iterate through 100 and 400 fields
Calculate Levenshtein Distance between problematic name and selected field
If Levenshtein Distance is smallest calculated, remember distance
If Levenshtein Distance is smallest calculated, remember distance and 100 field
If Levenshtein Distance is small enough, select 100 field as replacement for problematic name
Select solutions for problematic titles (implement in future)
Select solutions for problematic corporate names (implement in future)
Select solutions for problematic name-titles (implement in future)
Select solutions for problematic subjects (implement in future)
Check selected solutions against OCLC version of bib record, keep the solutions it agrees with
Add record with selected solutions to collection of records to upload

6. Development

The project goal of fixing an unauthorized access point is to find the authority record that lists the unauthorized access point as a variant, and to replace the unauthorized access point in the bibliographic record with the authorized access point from the authority record. This would be easy if the unauthorized access point listed some sort of unique id number that points to the authority record that the access point is meant to be associated with. While this is theoretically doable, it is not generally done, and none of the bibliographic records we looked at during the development process had any direct pointers to authority records. This means that the authority record that is needed is not immediately obvious, but instead needs to be discovered by using relevant data from the unauthorized access points, and the bibliographic record that access point is in.

The tool we use to search for the correct authority file for personal names is VIAF's AutoSuggest API, which takes a string as an input, and outputs a JSON file listing all the access points that may be relevant to the query string. This tool was chosen because it is easy to send a query and get a response programmatically, which makes it convenient for automation. The AutoSuggest API may return a variety of suggestions based on the input, and we have to sort through those suggestions to see which, if any is the authority record that we are looking for.

The main challenge with AutoSuggest is sending queries that will get meaningful results back. There are simple cases to look out for, specifically when the query string ends with a comma or dash that ensure no results will be returned. This kind of pattern is easy to detect, and easy to fix. Simply removing the final character in these cases tends to return relevant suggestions. Less straightforward is how to handle queries with unusual diacritics. In some cases using all the unusual diacritics in the search will turn up nothing, but revising the query to remove those diacritics will yield relevant results. Sometimes this case is reversed, where the diacritics are the only way to get good suggestions. Our best solution for this is to always send a query with all the diacritics present, but if no results are returned and there are diacritics from outside the ASCII table, the query is re-sent with the non-ASCII characters removed.

Examples of personal names with unusual diacritics are:

aMi-ʾgyur-rdo-rje, cYoṅs-dge Gter-ston
aMourikē, D̲oula

Another major issue for querying AutoSuggest is knowing how many subfields should be included. Including subfields like dates, titles or numeration can make the search key more specific, but if it includes information that VIAF does not have, the results can come up blank. During the development process we found search terms that include a single additional subfield can produce a small number of relevant looking results, but if more than one subfield is included, typically no results are returned. Because of this we send multiple queries to AutoSuggest until results are returned, each query including a different subfield,

and on with just the name. All of this adds up to the potential for multiple queries being sent for a single name until we get some result to examine.
For example, for the problematic access point “aŚāhajī, cKing of Tanjore, dfl. 1684-1712.” our first two queries to VIAF contain the name and date information (http://www.viaf.org/viaf/AutoSuggest?query=Śāhajī,+fl+1684-1712 and http://www.viaf.org/viaf/AutoSuggest?query=Sahaji,+fl+1684-1712) return no results. Our third query which includes the name and title information (http://www.viaf.org/viaf/AutoSuggest?query=Śāhajī,+King+of+Tanjore,) returns one unique personal name, which is then selected as our solution for this name. In contrast, a query that combines all three subfields (http://www.viaf.org/viaf/AutoSuggest?query=Śāhajī,+fl+1684-1712+King+of+Tanjore,) returns no suggestions.

Once AutoSuggest has given us results, we need to decide if any of the authorities that suggests are what is meant by the unauthorized access points we’re looking at. During development we noticed that in some of the authorities that AutoSuggest returned, the name from the query is listed as an associate, for example a search for “Robert Craft” would return the Library of Congress control number (LCCN) for “Igor Stravinsky,” which lists “Craft” as an associate. To avoid these cases, and find the best case, we began checking the 100 and 400 fields of the authority record against the unauthorized name. This successfully excluded the obviously wrong cases, but it also excluded some cases that were actually correct, but had minor differences in punctuation, where the unauthorized name may have an extra period or comma that the 100 or 400 field did not.
Examples of such cases are “Stephenson, Andrew G.,” which is the name listed in our bibliographic record while the closest match that could be found is “Stephenson, Andrew G.”, and “Goldschmidt, Jenny ELisabeth.” from our record vs “Goldschmidt, Jenny Elisabeth” as the closest match.
700 10 Stephenson, Andrew G.
400 1 Stephenson, Andrew G.
100 1 Mohammadou, Eldridge
400 1 Mohamadou, Eldridge
This, along with a case where the unauthorized name had a pipe character it should not have, written in MARCXML as <subfield code="a">Cockrell, W. D. |q (William D.)</subfield>, made it clear that it may be impossible to find an exact match for the unauthorized name we are looking at, even if we find the correct authority record. The pipe character showed that we could not simply exclude periods or commas, as the variations may be more unpredictable than that. Instead, our solution was to look through all the 100 and 400 fields from the suggested authority records and determine how similar each of them is to the unauthorized name we are trying to fix.
We do this by calculating the Levenshtein distance, also known as the edit distance, to determine the fewest number of changes it takes to change the unauthorized name to the field we are assigning a similarity score to. Few changes mean the names are already quite similar. At this point we just need to

concern ourselves with the authority record with the smallest Levenshtein distance calculated. If that distance is too big, we can determine that none of the suggested authorities were similar enough to be considered good matches. This is where we can adjust how conservative or aggressive we want our changes to be. A larger maximum allowed Levenshtein distance will return more changes, but runs the risk of selecting more bad solutions. A smaller maximum will mean fewer changes, but those changes will be more likely to be correct. We wanted to be fairly conservative for this project and found that a Levenshtein distance of 2 was the highest where we were satisfied with the suggestions.

One of the hurdles in this process was how to retrieve the LC authority records we've been discussing. We get the LCCN for the authority record from AutoSuggest, but it was not immediately clear what the best service was to retrieve the full authority record. The easiest and most accessible service was using the LCCN Permalink service to retrieve the authority as a MARCXML file. LCCN permalinks are URLs for LC bibliographic records and authority records. This is a well-documented service and consists of simply adding the LCCN to the end of a standard URL, and sending an HTTP request with that URL. The problem is that the Library of Congress limits access to this service to one request every six seconds, which is too slow for a project on this scale. The Library of Congress allows more frequent access through their Z39.50 service, but that service isn't as well documented or straightforward. We eventually took the time to learn how to send requests over Z39.50 because the speed was so important. We did this by importing a Python library called PyZ3950 that allows us to make Z39.50 requests, and send queries formatted as "@attr 1=9 [LCCN]", which gets us the authority record we're looking for.

There were a few issues concerning problems with character encoding that arose. First, the results of our SQL query returned characters outside the standard ASCII character set as question marks. This means that a name with unusual diacritics would have a question mark in the middle of it, or that a name in full Hebrew script would be entirely made up of question marks with the exception of any date information. For example the name listed as "|a בלבן, אברהם, |d 1944-." in our records was listed as "a????????, ?????????,d1944??" in the query results we were reading from. Eventually it became clear that the names in the second case are always in the 880 field, which should actually be labeled as a "see" reference, and we simply need to ignore those names.

To get around the first case, when we retrieve the full record from our database, we match the unauthorized names with the 100 and 700 fields in the bibliographic record, again calculating the Levenshtein distance, to find the name in the record. At first these full bibliographic records that we imported had character encoding issues of their own. Diacritics were being applied to the character before they should have been, the diacritics themselves were wrong, and 880 fields were being filled with gibberish instead of question marks. For example the name "|a בין אל לחיה : |b עיון ביצירתו של עמוס עוז / |c אברהם בלבן." would be incorrectly written in MARCXML as:

```
<subfield code="a">(2aio `l lgid :(B</subfield>
<subfield code="b">(2rieo aivixze yl rneq ref /(B</subfield>
<subfield code="c">(2`axdm alao.(B</subfield>
```

All this led to the conclusion that the bibliographic records were not encoded in UTF-8. The fact that the diacritics were coming before the character they should have been applied to, as opposed to following the character as is the case in UTF-8, eventually led to the realization that these records were encoded in MARC-8. To decode the MARC-8 characters, we now retrieve the bibliographic record as an .mrc file and load that into the pymarc library in Python, which has the built in ability to decode MARC-8. Once these problems were solved, the names from the bibliographic record were in good shape to be used in the query sent to AutoSuggest.

Using the methods described above to query VIAF AutoSuggest and use the Levenshtein distance to find the best suggestion ended up producing results for 93% of the names we looked at. But when we manually examined the quality of those results, we found that 5% of them were the wrong name. This project is expected to be run on around half a million personal and corporate names, and an error rate of 5% would result in tens of thousands of unauthorized access points to be overwritten with authorized but incorrect access points. It was determined that a good way of filtering out these errors would be to compare our solutions with solutions that are acquired through an unrelated method, and only allow solutions where both methods agree. No method is likely to be completely accurate, but if the two methods are independent, errors are unlikely to occur for the same name in both methods.

The independent method we chose was to compare our results to OCLC's bibliographic records. This is easy to do because all of our bibliographic records have an OCLC number listed, so it's simply a matter of retrieving the record and comparing the names. We also do not want to simply copy all the information in OCLC, because OCLC has its own errors that we do not want to blindly perpetuate and use in place of our own potentially correct information. Combining our conclusions with OCLC's records gives us the best of both sources, while avoiding replacing our current access points with incorrect ones. The percent of names we replace is reduced with this method, down to around 75%, while reducing errors to a fraction of a percentage point. While this leaves a large group of names unfixed, we output or best guesses for the unfixed names in a spreadsheet, which gives humans fixing the errors a good place to start.

7. Testing & Results

Because of the large scale of this project, an important part of the development cycle was testing the code on a sample set, seeing what errors occur, and adjusting the code to account for those errors. We want to improve our existing database without pushing many errors, which is a danger with automation, so testing and analyzing the results were a constant part of our process. For early development, we tested our code on a sample of around 550 unauthorized access points. When the code could not find a solution for an access point, we looked for ways to find a solution based on the unauthorized access points and adjusted the code accordingly. When we did find a solution, we examined the solution to determine if it looked correct, and if it was not we looked for ways to fix it, or exclude it if there was no way to fix it. As our solutions for this sample became good, we expanded our testing to another range of around 1,000 unauthorized access points to see how well our code performed and to address any problems that emerged with the new sample. For a second set of about 1,000 unauthorized access points, we assigned expected solutions to all the access points before we ran the code, and after the code ran we compared the results.

The results were as follows. The solution that the algorithm chose was wrong 1.3% of the time, but none of those cases were pushed as final solutions, instead the algorithm detected a problem with all of these answers and simply listed them as a best guess for these names that it could not find an acceptable solution for. 2.8% of the time the solutions selected by the human and the program were different, but both were valid. In other words, these were cases where multiple authority records existed for the same person, and the human and computer selected different records. In this case the human tended to choose authority records with 670 fields with the title of the work in question, or with a title that looked to be in the same field as the work in question, while the algorithm did not take the 670 field into consideration. 1.8% of the time the algorithm chose correctly, and the human chose incorrectly. In the remaining 94% of cases the human and computer agreed on the solution. The main takeaway from these results is that our algorithm is roughly on par with a human, but it is much faster and can catch its own mistakes, though we can still improve performance by checking the 670 field.

Overall, when we run the code across all three samples of training data, it looks at 2,412 unauthorized personal names, and finds what it considers to be an acceptable solution for 1,915, or 79.4% of access points. A remaining 20.6% of access points are not changed by the algorithm and require a human to examine and fix. The margin of error for these numbers is ±2%. At the time of writing, we have found that 2 of the written solutions are wrong, which is 0.08% of the unauthorized names examined.

We compared the performance of our code to MarcEdit's Validate Headings tool, which also tries to automatically fix unauthorized access points. Both tools were run on a sample of 705 unauthorized personal names, and we compared the quality of the results. In 542 cases, or 77% of the time, we found that our tool and MarcEdit both came to the same conclusion. This could mean they both chose the same solution, or they both were unable to come up with a solution. There are four cases (1% of the time) where the two tools come up with different solutions but both solutions are wrong. There are 13 cases (2% of the time) where the two tools come up with different solutions, but there's not enough information to determine which solution is better. There are 100 cases (14% of the time) where the two tools come up with different solutions, and our code produced the correct solution. The reverse case, where the two tools disagree and MarcEdit is the one that produces the correct solution, happened in 46 cases, or 7% of the time.

In addition to our code coming up with more good solutions, our code has a major advantage over MarcEdit. Whenever MarcEdit finds a solution, it writes that solution to the collection of records being processed, regardless of the quality of that result. Our code will only write a result to the records if that result can be found in the OCLC version of the bibliographic record being fixed. What this means is that every time we found that MarcEdit had made a mistake, that mistake was written to the same record that the other solutions are written to. But every time we found that our code had found an incorrect solution, the code itself had also determined that its solution was not good enough, and wrote the solution to a spreadsheet for humans to look at, and it did not write the incorrect solution to the collection of records to be updated. So MarcEdit produces fewer correct solutions and generates new incorrect data, while our code produces more correct solutions and avoids pushing errors into the records. For these reasons we feel that our code does a better job at fixing unauthorized access points than MarcEdit's Validate Headings tool.

Because of our methods, there are limitations to how many unauthorized access points we can fix. First, because of how the query is structured, we only look at names that are labeled as "see" references. Any access point that is unauthorized, but not labeled will not be collected by the query. While our code to process the query results will work with a spreadsheet of non-see reference unauthorized names, as long as the spreadsheet is formatted correctly, the code is largely untested on this sort of data, so it's not clear how good the results would be.

Second, so far the tool has only been developed and tested for personal and corporate names. The code is designed so that in the future other access point like subjects, series, titles, and name-titles can be addressed, but these methods have not been tested on anything but personal and corporate names, and nothing has been written to address specific issues that other access point types might have.

Third, there is an upper bound of names based on the double check against the OCLC records. Because we ignore results that don't agree with the OCLC record, we can not fix a name that is not already accurate in OCLC. One way to get around this limitation would be to find a second independent source to check both our result and OCLC's record against. This way if any two of the three sources agree, that can be selected as the solution.

Finally the speed of our process is limited by the number of external calls our code makes. Every record requires two external calls: one to access our local record and one to access the OCLC version of the record. Every name that is processed calls the AutoSuggest API between 1 and 16 times depending on if any usable suggestions are returned. Every suggestion from AutoSuggest results in a Z39.50 call to retrieve the authority record from the Library of Congress. The number of calls here varies based on what the results from AutoSuggest are, but there are typically only a small number of suggestions. The calls to AutoSuggest have the greatest impact on the speed of this process. It will realistically be called all 16 times for any names that VIAF doesn't have or doesn't have an LCCN for, both cases are a small, but sizable fraction of our dataset. In addition, we have observed AutoSuggest to have the slowest response time of all the services we call in this process.

There is no way to avoid the worst case scenario for AutoSuggest because a slightly different query has the potential to return relevant results, even when other queries have failed. The best way to speed up processing an entire database with this tool would be to split the data that needs to be processed into smaller chunks and to run a few of these chunks in parallel on different machines at the same time. This would have to be limited to only a few machines at a time to avoid overloading the services this tool is using, but processing the data across a few machines at once would divide the total amount of time needed to process it all.

Bib Record No.	**Total**	**Automatically Changed**	**Need Manual Correction**
0 - 1 million	79091	63783	15308
1 - 2 million	77073	60869	16204
2 - 3 million	65512	50427	15085
3 - 4 million	62124	46572	15552
4 - 5 million	51012	34000	17012
5 - 6 million	27994	17788	10206
6 - 7 million	34971	23184	11787
7 - 8 million	12378	3696	8682
Total	**400,500**	**295,500 (73.8%)**	**105,000 (26.2%)**

At the time of writing (April 2017), we run the query to find all unauthorized personal and corporate names in our online catalog, and discover that there are around 400,500 personal and corporate names that need to be corrected. After we run the script for multiple 5,000 files for those 400,000 personal and corporate names, we have successfully corrected around 300,000 personal names and corporate names by machine. We still have over 100,000 personal and corporate names that we need to check them by hand. Among those 100,000 personal and corporate names, we believe many of them should be correct. We set them aside for our conservative calculations.

The automatically changed records are in XML format. So we need to convert XML to MARC format using MarcEdit, and then import MARC files into Voyager and update the records. Given the large number of auto-corrected records, our team hopes that our consortium will help to set up a profile to finish the work. The records that need manual check and correction are in csv format. Catalogers who are familiar with multiple foreign languages may be hired to finish this work through exploring LC Authority files and OCLC records.

In the meantime, we are working to create a different script to correct subject access points in our online catalog. In the future, we plan to use the script and the workflow to continuously work on authority maintenance work since incorrect names and subjects appear in our online catalog every day.

8. Conclusion

A critical part of linked library data lies within the establishment of its backbone: authority data. Our script has not only corrected around 400,000 personal and corporate names in western European languages, but also has also fixed personal and corporate names in diacritics and Unicode non-Roman languages in our online catalog. We hope that our script and workflow of fixing unauthorized access points in our bibliographic records can help other libraries as they are preparing to migrate to the linked open data environment.

Notes

1. Linked data: Connect Distributed Data Across the Web http://linkeddata.org/
2. BIBFLOW: A Roadmap for Library Linked Data Transition, Prepared 14 March, 2017, MacKenzie Smith, Carl G. Stahmer, Xiaoli Li ,Gloria Gonzalez, University Library, University of California, Davis Zepheira Inc http://roytennant.com/BIBFLOWRoadmap.pdf
3. Qiang Jin, Demystifying FRAD: Functional Requirements for Authority Data (Santa Barbara, Libraries Unlimited, 2012), 3.
4. IFLA Study Group on Functional Requirements and Numbering of Authority Records (FRANAR), Functional Requirements for Authority Data A Conceptual Model: Final Report (Munchen: K. G. Saur, 2013 http://www.ifla.org/files/assets/cataloguing/frad/frad_2013.pdf
5. IFLA Working Group on the Functional Requirements for Subject Authority Records (FRSAR), Functional Requirements for Subject Authority Data (FRSAD) A Conceptual Model (Munchen: K. G. Saur, 2010) http://www.ifla.org/files/assets/classification-and-indexing/functional-requirements-for-subject-authority-data/frsad-final-report.pdf
6. Joint Steering Committee for Development of RDA: Resource Description and Access, 2010 https://access.rdatoolkit.org/
7. Bibliographic Framework Initiative https://www.loc.gov/bibframe/
8. OCLC WorldCat https://www.oclc.org/en/worldcat/data-strategy.html
9. University of Illinois at Urbana-Champaign Library www.library.uiuc.edu
10. CARLI (Consortium of Academic and Research Libraries in Illinois) https://www.carli.illinois.edu/products-services/i-share/reports
11. Michael Gorman, “Authority Control in the Context of Bibliographic Control in the Electronic Environment,” Cataloging & Classification Quarterly, 39 (2004): 13

12. Barbara B. Tillett, “Authority Control: State of the Art and New Perspectives,” Cataloging & Classification Quarterly, 39 (2004): 23.
13. D.I. Hillmann, R. Marker, & C. Brady, “Metadata standards and applications,” The Serials Librarian, 54 (2008): 1.
14. Sha Li Zhang, “Planning an Authority Control Project at a Medium-sized University Library,” College & Research Libraries, 62, no. 5, (2001): 395.
15. Sherry L. Vellucci, “Commercial Services for Providing Authority Control: Outsourcing the Process,” Cataloging & Classification Quarterly, 39 (2004): 443
16. Silvia B. Southwick, Cory K. Lampert, and Richard Southwick, “Preparing Controlled Vocabularies for Linked Data: Benefits and Challenges, Journal of Library Metadata, 15 (2015): 177.
17. System Properties Comparison Microsoft Access vs. Microsoft SQL Server vs. Oracle, https://db-engines.com/en/system/Microsoft+Access%3bMicrosoft+SQL+Server%3bOracle