

AUTOMATICALLY UPDATING LIBRARY HOLDINGS ON VENDOR PLATFORMS

By Clara Turp
Metadata Analyst Librarian
McGill University Library
May 10th 2018

PRESENTATION PLAN

1. Project Outline
2. Challenges
3. Proposed Solution

PROJECT OUTLINE

GOBI

Holdings on GOBI don't match the library's actual holdings.

CHALLENGES

ISBN

- In GOBI, the holdings are updated using the title's ISBN.
- The standard is misused especially when it comes to electronic resources.
- Challenges:

Electronic Resources
vs Print

Errors

Related ISBN

SUBSCRIPTIONS AND PERPETUAL TITLES

- Identified by OCLC Knowledge Base collections
- Possible overlap
- Collections change when subscription packages are revised

CHALLENGES FOR EACH DEVELOPMENT PHASE

This project is separated in two phases, both containing challenges

Original Upload

Monthly Uploads

PROPOSED SOLUTION

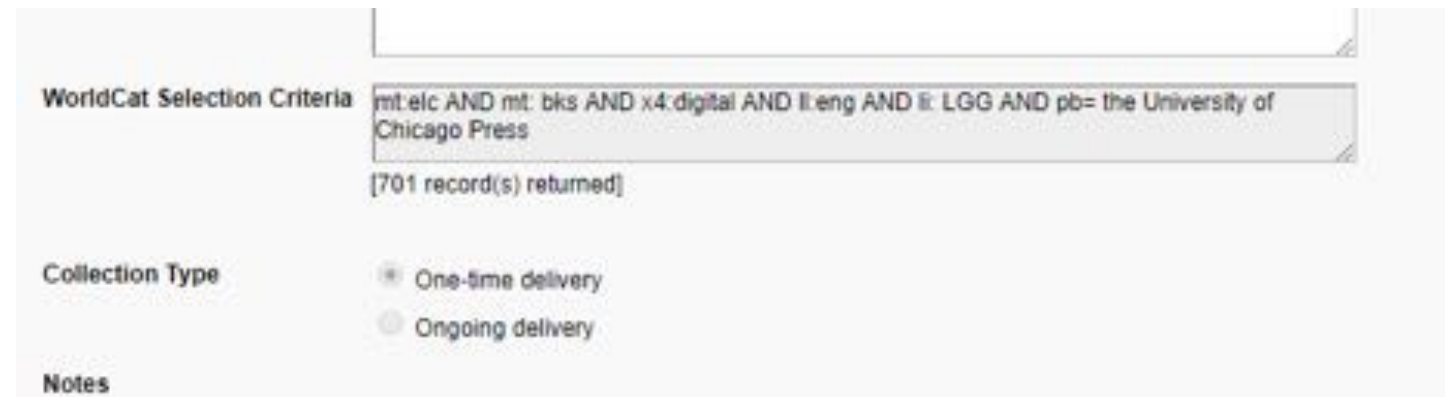
STEPS OF THE PROCESS

1. Create a Query Collection
2. Extract records with an ISBN (020 field)
3. Save as CSV File using MarcEdit.
 - a. Export 001, 020, 245 fields as tab (or comma) delimited values
4. Make sure all elsn comments are in the file.
5. If necessary, update the subscriptions list:
 - a. Upload MARC collections
 - b. Save as CSV: LDR, 001, 245, 300
 - c. Run subscription_clean.py
6. Run API code of all subscriptions to remove the ones also found as a perpetual title.
7. First phase of the code (gobi_FirstPhase.py) : Clean and select certain ISBNs
8. Take all selected ISBN (1_FirstPhase.txt)
9. Batch Search Gobi (Search all titles by ISBN)
10. Second phase of the code (gobi_SecondPhase.py) : Select only electronic books and match the titles.
11. Third phase of the code (errorHandling.py)
12. Manually find ISBN for all errors (found in error files)
13. Send all ISBN to GOBI

QUERY COLLECTIONS (OCLC)

Using query collections presented two important advantages

1. MARC records delivery
2. Use of filters



The screenshot displays the WorldCat Query Collections interface. It features a search bar at the top, followed by a section titled "WorldCat Selection Criteria" containing the query: `mt:elc AND mt: bks AND x4: digital AND ll: eng AND ll: LGG AND pb= the University of Chicago Press`. Below the query, it indicates "[701 record(s) returned]". Under the "Collection Type" section, there are two radio buttons: "One-time delivery" (which is selected) and "Ongoing delivery". At the bottom, there is a "Notes" section.

MARCEDIT

- Extract records with an ISBN (020)
- Save as CSV selecting particular fields
 - OCN, ISBN, Title

The image shows two overlapping windows from the MARCEDIT software. The top window, titled 'Modify Selected MARC Records', displays a list of records with checkboxes. Records 2 and 3 are selected. The 'Display Field (245\$a)' is set to '245\$a'. The bottom window, titled 'Export Tab Delimited Records', shows 'Step 2: Define Fields/Subfields to Export'. The fields 001, 020, and 245 are listed in the export field box. The 'Normalize field data' checkbox is unchecked. The 'Field' dropdown is set to 'F#:020'. The 'Add Field' and 'Delete Field' buttons are visible. The 'Export' button is highlighted.

Modify Selected MARC Records

Record Nu...	Display Field (245\$a)
<input type="checkbox"/> 0	Ray Harryhausen: special effects.
<input type="checkbox"/> 1	Samaras 1974 :
<input checked="" type="checkbox"/> 2	Prints from blocks, Gauguin to now /
<input checked="" type="checkbox"/> 3	Technics and creativity II.
<input type="checkbox"/> 4	Video from Tokyo to Fukui and Ky...
<input type="checkbox"/> 5	fauvismo /
<input type="checkbox"/> 6	Georg Baselitz, monumental prints.
<input type="checkbox"/> 7	Ben Shahn /
<input type="checkbox"/> 8	German painting and sculpture :

[Does Not Match](#) | [Invert Selections](#)

Search: Field:

☐ Retain Checked Items

Export Tab Delimited Records

Step 2: Define Fields/Subfields to Export

001
020
245

☐ Normalize field data

Field: Subfield: [Add Field](#) [Delete Field](#)

[Back](#) [Export](#) [Close](#) [Settings...](#)

PYTHON SCRIPTS

Cleans the ISBN and makes a selection using comments
(020 \$q)

Separate
Subscriptions

Triage using
comments

Clean up
(active, 13, x)

```
(Adobe digital ed)
Adobe digital ed
(Adobe digital ed ;
(Adobe digital ed);
Adobe digital ed;
(digital)
digital
(digital ;
(digital);
digital;
(ebk)
ebk
(ebk ;
(ebk);
ebk;
ebook
(ebook)
(ebook ;
(ebook);
ebook;
eBook
(eBook)
(eBook ;
(eBook);
eBook;
(e-book)
e-book
(e-book ;
(e-book);
```

PYTHON SCRIPTS

```
#import all possible comments in array
isbnComments = []
commentFile = open("EisbnComments.txt", "r", encoding = "utf-8")
reader = commentFile.read()
isbnComments = reader.split("\n")
commentFile.close()

subsOcn = []
subs245 = []
subscriptionFile = open("eBookSubscriptions.csv", "r", encoding = "utf-8")
reader = csv.reader(subscriptionFile)
for row in reader:
    subsOcn.append(row[1])
    subs245.append(row[2])

myArray = []
myErrorArray = []
SubscriptionArray = []
with open("MITPress.csv", "r", encoding = "utf-8") as f:
    reader = csv.reader(f)
    for row in reader:
        if row[0] in subsOcn:
            SubscriptionArray.append(row)
        elif row[2] in subs245:
            SubscriptionArray.append(row)
```

```
#if there is a $q, match the comment.
else:
    if len(isbn2[0]) > 12:
        if isbn2[1] in isbnComments:
            if len(secondLevelArray) < 2:
                secondLevelArray.append(isbn2[0])
            else:
                anotherArray.append(row[0])
                anotherArray.append(isbn2[0])
                anotherArray.append(row[2])

        if len(anotherArray) == 3:
            myArray.append(anotherArray)
            anotherArray = []

secondLevelArray.append(row[2])
if len(secondLevelArray) < 3 :
    myErrorArray.append(secondLevelArray)
else:
    myArray.append(secondLevelArray)
```

PYTHON SCRIPTS

1. Remove ISBN unmatched by GOBI
2. Keep ISBN only for ebook titles
3. Catch all lost OCN (only errors)

```
with open("gobi_list.csv", "r", encoding= "UTF-8") as f:
    reader = csv.reader(f)
    for row in reader:
        if row[6] == "eBook":
            if row[5] not in resultIsbnList:
                resultIsbnList.append(row[5])
        else:
            if row[5] not in errorGobiIsbn:
                errorGobiIsbn.append(row[5])
f.close()

myErrorFile = open("2_allRejected.txt", "w", encoding= "UTF-8")
with open("firstResults.csv", "r", encoding= "UTF-8", errors = "ignore") as f:
    reader = csv.reader(f)
    for row in reader:
        if row[1] in errorGobiIsbn:
            for instance in row:
                myErrorFile.write(instance + "\t")
            myErrorFile.write("\n")
            errors.append(row[0])
myErrorFile.close()
f.close()

myPrintFile = open('2_SecondPhase.txt', 'w', encoding= "UTF-8")
with open("firstResults.csv", "r", encoding= "UTF-8", errors = "ignore") as f:
    reader = csv.reader(f)
    for row in reader:
        if row[1] in resultIsbnList:
            for instance in row:
                myPrintFile.write(instance + "\t")
            myPrintFile.write("\n")
            goodOnes.append(row[0])
myPrintFile.close()
f.close()

unmatchedOcn = open("2_ErrorSecondPhase.txt", "w", encoding= "UTF-8")
for i in range(0, len(errors)):
    if errors[i] not in goodOnes:
        unmatchedOcn.write(errors[i])
        unmatchedOcn.write("\n")
```

1 & 2

3

PYTHON SCRIPTS

Error handling using:

1. Up to 4 words of the title (no subtitle)
2. Remove punctuation, and certain stop words
3. Compare results between the original list of titles (245 field) and the one from GOBI's batch search

PYTHON SCRIPTS

```
import csv
import string

stopWords = ["AND", "A", "AN", "B", "THE", "OF", "FOR"]
marcSymbolArray = ["$", "/", ":", ";"]

gobiArray = []
gobiListFile = open("gobi_list.csv", "r", encoding = "utf-8", errors = "ignore")
reader = csv.reader(gobiListFile)
for row in reader:
    titleWords = ""
    wordCount = 0
    if len(row[0]) > 0:
        title = row[0].split(":")
        title = title[0]
        titleArray = title.split(" ")

        letterFound = False
        while x < len(titleLetter) and letterFound == False:
            if titleLetter[x] not in marcSymbolArray:
                myString = myString + titleLetter[x]
            if titleLetter[x] in marcSymbolArray:
                letterFound = True
                matchFound = True
            x = x + 1
        titleArray[i] = myString
        titleWords = titleWords + " " + titleArray[i]
        wordCount = wordCount + 1
    i = i + 1
translator = titleWords.maketrans('', '', string.punctuation)
titleWords = titleWords.translate(translator)
titleWords = titleWords.strip()

    if titleWords not in oriArray:
        oriArray.append(titleWords)

oriArray = sorted(oriArray)
gobiArray = sorted(gobiArray)

printFile = open("3_ErrorsTitleMatch.txt", "w")
for instance in gobiArray:
    if instance not in oriArray:
        printFile.write(instance + "\n")
```



OCLC'S API

- This phase is still in development.
- Upload MARC records for all titles from a subscription package.
- Use the OCLC's Knowledge Base API to see if OCLC number is found in multiple collections

OCLC'S API

1. WorldCat Knowledge Base API
2. Entry – Search

```
param = "oclcnum=85855312"  
myKey = "&wskey=" + myKey  
myUrl = "http://worldcat.org/webservices/kb/rest/entries/search?" + param + myKey
```

3. XML Parsing (using python and regular expressions)

```
for i in re.findall("<kb:collection_uid>", data):  
    y = y + 1  
for m in re.findall("<kb:collection_uid>[A-Za-z0-9._-]*</kb:collection_uid>", data):
```

CONTACT

<https://github.com/ClaraTurp/updateHoldings>

clara.turp@mcgill.ca

QUESTIONS?

Thank you!