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
Holdings on GOBI don’t match the library’s actual holdings.
- 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
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 elsbn 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
Using query collections presented two important advantages

1. MARC records delivery
2. Use of filters
Extract records with an ISBN (020)

Save as CSV selecting particular fields

- OCN, ISBN, Title
PYTHON SCRIPTS

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

Separate Subscriptions

Clean up (active, 13, x )

Triage using comments
# 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)
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
import csv
import string

stopWords = ['AND', 'A', 'AN', '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(" ")
    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
    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

```python
param = "oclcnum=858555312"
myKey = "&wkey=
myUrl = "http://worldcat.org/webservices/kb/rest/entries/search?" + param + myKey
```

3. XML Parsing (using python and regular expressions)

```python
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