Hybrid linked data approaches in traditional discovery environments using Share-VDE linked data

Jim Hahn, University of Pennsylvania Libraries
jimhahn@upenn.edu
Hybrid Linked Data

https://guides.library.upenn.edu/linked-data
It is often possible to include linked data in more traditional representations, or to make connections between linked data and more familiar formats.

We expect these mixed-format or “hybrid” linked data environments to be the most common way in which linked data is used in production in the next few years.

https://guides.library.upenn.edu/linked-data
Traditional Discovery
"Strong Medicine speaks"

Title: "Strong Medicine speaks"
Subtitle: a Native American elder has her say : an oral history /
Author: Hearth, Amy Hill, 1958-
Format: Book
Language: English
SVDE Linked Data
The Share-VDE project (https://svde.org) is a collaborative discovery environment based on linked data. Explored in this talk are several lesser known and non-intuitive uses of Share-VDE linked data.

**Share-VDE Data**

**Deliverables from Share-VDE**

<table>
<thead>
<tr>
<th>Enriched MARC</th>
<th>RDF triples</th>
<th>SVDE.org</th>
</tr>
</thead>
</table>
"Strong Medicine speaks"

Title: "Strong Medicine speaks"
Subtitle: a Native American elder has her say : an oral history /
Author: Hearth, Amy Hill, 1958-
Format: Book
Language: English
Published: New York
Call number: E99.D2 H437 2008
ISBN: 9780743297790 and 0743297792

Amy Hill Hearth


Amy Hill Hearth is an American journalist and author who specializes in stories about women. She is the author or co-author of eight nonfiction books, including the oral history Having Our Say: The Delany Sisters' First 100 Years, a New York Times bestseller for 113 weeks, according to its archives. - Wikipedia

More options →
SVDE Linked Data for Alma Automated Subject Assignment
Alma Sandbox Experiment

Alma/Annif Nightshift

jimhahn@upenn.edu

The project was inspired by the BookOps Nightshift project (https://github.com/BookOps-CAT/NightShift), a copy cataloging bot that used OCLC numbers to match brief records. This variation may be useful to enhance brief records when OCLC matching isn't possible.

The data flow begins from Alma Brief record reports. The titles (and/or authors) are sent to a pre-packaged machine learning service for FAST subject assignment using the Annif codebase (https://github.com/NatLibFi/Annif).


The data in the customized Fast Annif API were collected from Penn Libraries and IvyPlus POD data (https://pod.stanford.edu/).
Get a row of the Alma report of titles to be processed

```python
# Import the data as string data types
nosubjectsdf = pd.read_csv('/Users/jimhahn/Documents/GitHub/alma-nightshift/empty')
```

```python
# What does the data look like?
nosubjectsdf.head()
```

<table>
<thead>
<tr>
<th>Type / Creator / Imprint</th>
<th>Title</th>
<th>Barcode</th>
<th>Inventory Number</th>
<th>Receiving Number</th>
<th>Library</th>
<th>Library Unit</th>
<th>Temporary Library</th>
<th>Creation Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book By Cartarescu, Mircea (2022)</td>
<td>Premio FIL de Literatura en Lenguas Romances /</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>LIBRA</td>
<td>NaN</td>
<td>NaN</td>
<td>03/30/2023 16:24:59</td>
</tr>
</tbody>
</table>

1 rows x 44 columns
Annif-client (Public)

master 3 branches 2 tags

juhoinkinen Bump version: 0.3.0 → 0.3.1 16fd749 on Mar 11, 2021 42 commits

- .github/workflows Create python-package.yml 2 years ago
- tests Add minimal test suite (only tests projects property) 4 years ago
- .gitignore Initial commit 5 years ago
- LICENSE.txt switch license to Apache 2.0 (same as used by Annif) 5 years ago
- README.md update README to mention PyPI install 5 years ago
- annif_client.py Merge branch 'master' into issue1-support-for-learn... 4 years ago
- setup.cfg Bump version: 0.3.0 → 0.3.1 2 years ago
- setup.py Bump version: 0.3.0 → 0.3.1 2 years ago

About

Python client library for accessing Annif REST API

annif

- Readme
- View license
- Activity
- 2 stars
- 5 watching
- 2 forks

Report repository

Releases

- 2 tags

Contributors 2

https://github.com/NatLibFi/Annif-client
if __name__ == '__main__':
    annif = AnnifClient()

    # select the title from the nosubjectsdf
    title = nosubjectsdf['Title'].iloc[0]

    # send it to Annif API
    url = 'http://jimhahn-dev.library.upenn.int:5000/v1/projects/nn-ensemble-Fast',
    payload = {'text': title}
    req = requests.post(url, data=payload)
    print(req.raise_for_status())
    print(req.json()['results'])
    #print(req.json()['results'][0]['url'])
    #save the response as a key value pair in pandas dataframe called subjectsf
    subjectsf = pd.DataFrame(req.json()['results'])
    print(subjectsf)

http://annif.info/
Evaluate the subject recommendations looking for any have a confidence score above 0.5

```python
# check if the subjects data frame has any values above 0.5
# and if so, add the subject to a new column in the nosubjectsdf and the subject
if subjectsdf['score'].iloc[0] > 0.5:
    nosubjectsdf['subject'] = subjectsdf['uri'].iloc[0]
    nosubjectsdf['label'] = subjectsdf['label'].iloc[0]
else:
    print("No subject found")
```

Get the brief record from Alma for the title so that we can add the subject to the MARC record

```python
# use for statement in https://api-na.hosted.exlibrisgroup.com/almaws/v1/bibs/:mm:
# to download the marc records
# create a for loop to iterate through the mmsid list
# and download the marc record for each mmsid
# and save it to a folder on the desktop
# create a list of mmsids from the nosubjectsdf
mmsids = nosubjectsdf['MMS_ID'].tolist()
print(mmsids)

# create a for loop to iterate through the mmsid list
# and download the marc record for each mmsid
# and save it to a folder on the desktop
for mmsid in mmsids:
    print(url)
    r = requests.get(url)
    # parse the response as xml
    root = ET.fromstring(r.content)
    # print(root)
    # print(ET.tostring(root, pretty_print=True))
    # we only want the record
    record = root.find('://record')
    # print(record)
    # print(ET.tostring(record, pretty_print=True))

    # save the record as a file
    filename = mmsid + '.xml'
    print(filename)
    with open(filename, 'wb') as f:
        f.write(ET.tostring(record, pretty_print=True))
    f.close()
```
Before we can add the subject to the MARC record we need to know the type of subject we are working with here

# we need to check the type of subject
# we can lookup FASTAll/lookup/nt to see the type of subject
# If the subject is found in FASTChronological.nt it is a Chronological subject
# If the subject is found in FASTCorporate.nt it is a Corporate subject
# If the subject is found in FASTEvent.nt it is an Event subject
# If the subject is found in FASTForm.nt it is a Form subject
# If the subject is found in FASTGeographic.nt it is a Geographic subject
# If the subject is found in FASTNamedEvent.nt it is a Named Event subject
# If the subject is found in FASTPersonal.nt it is a Personal subject
# If the subject is found in FASTTopical.nt it is a Topical subject
# If the subject is found in FASTUniformTitle.nt it is a Uniform Title subject

#query the FASTAll/lookup/nt/ folder to see if the subject is there
# if it is, then add the subject type to the nosubjectsdf
# open the nt and read it
# if the subject is found in the nt, then add the subject type to the nosubjectsdf

# create a list of subjects from the nosubjectsdf

#make the FASTAll/lookup/FASTTopical-nt.csv into a dataframe
fasttopicaldf = pd.read_csv('./FASTAll/lookup/FASTTopical-nt.csv', dtype=str)

#make the FASTAll/lookup/FASTChronological-nt.csv into a dataframe
fastchronologicaldf = pd.read_csv('./FASTAll/lookup/FASTChronological-nt.csv', dtype=str)

#make the FASTAll/lookup/FASTCorporate-nt.csv into a dataframe
fastcorporatedf = pd.read_csv('./FASTAll/lookup/FASTCorporate-nt.csv', dtype=str)

#make the FASTAll/lookup/FASTEvent-nt.csv into a dataframe
fasteventdf = pd.read_csv('./FASTAll/lookup/FASTEvent-nt.csv', dtype=str)

#make the FASTAll/lookup/FASTForGenre-nt.csv into a dataframe
fastformdf = pd.read_csv('./FASTAll/lookup/FASTFormGenre-nt.csv', dtype=str)

#make the FASTAll/lookup/FASTGeographic-nt.csv into a dataframe
fastgeographicdf = pd.read_csv('./FASTAll/lookup/FASTGeographic-nt.csv', dtype=str)

#make the FASTAll/lookup/FASTNamedEvent-nt.csv into a dataframe
fastnamedeventdf = pd.read_csv('./FASTAll/lookup/FASTNamedEvent-nt.csv', dtype=str)

#make the FASTAll/lookup/FASTPersonal-nt.csv into a dataframe
fastpersonaldf = pd.read_csv('./FASTAll/lookup/FASTPersonal-nt.csv', dtype=str)

#make the FASTAll/lookup/FASTTitle-nt.csv into a dataframe
fastuniformalldf = pd.read_csv('./FASTAll/lookup/FASTTitle-nt.csv', dtype=str)
We're going to be using pymarc to read the marc records and add a new subject field. Convert the MARCML at /data/marc/toc MARC using pymarc.

```python
mmsids = nosubjectsdf['MMS_ID'].tolist()
print(mmsids)
```

# Create a for loop to iterate through the mmsid list
# and download the marc record for each mmsid
# and save it to a folder on the desktop

```python
for mmsid in mmsids:
    with open(mmsid + '.mrc', 'rb') as fh:
        reader = MARCReader(fh)
        for record in reader:
            record.add_field()
            # Evaluate the type of topic from the nosubsjects, if topical use 650:
            if nosubjectsdf['subject_type'].iloc[0] == 'Topical':
                record.add_field()
                field = '650'
                indicators = [' ', '7'],
                subfields = [
                    '', nosubjectsdf['label'].iloc[0],
                    '7', 'fast',
                    '0', nosubjectsdf['subject'].iloc[0]
                ]
            # Evaluate the type of topic from the nosubsjects, if chronological use
            elif nosubjectsdf['subject_type'].iloc[0] == 'Chronological':
                record.add_field()
                field = '648'
                indicators = [' ', '7'],
                subfields = [
                    '', nosubjectsdf['label'].iloc[0],
                    '7', 'fast',
                    '0', nosubjectsdf['subject'].iloc[0]
                ]
            # Evaluate the type of topic from the nosubsjects, if corporate use 610
            elif nosubjectsdf['subject_type'].iloc[0] == 'Corporate':
                record.add_field()
                field = '610'
                indicators = [' ', '7'],
                subfields = [
                    '', nosubjectsdf['label'].iloc[0],
                    '7', 'fast',
                    '0', nosubjectsdf['subject'].iloc[0]
                ]
            # Evaluate the type of topic from the nosubsjects, if event use 611:
            elif nosubjectsdf['subject_type'].iloc[0] == 'Event':
                record.add_field()
                field = '611'
                indicators = [' ', '7'],
                subfields = [
                    '', nosubjectsdf['label'].iloc[0],
                    '7', 'fast',
                    '0', nosubjectsdf['subject'].iloc[0]
                ]
            # Evaluate the type of topic from the nosubsjects, if form/genre use 65:
            elif nosubjectsdf['subject_type'].iloc[0] == 'Form/Genre':
                record.add_field()
                field = '655'
                indicators = [' ', '7'],
                subfields = [
                    '', nosubjectsdf['label'].iloc[0],
                    '7', 'fast',
                    '0', nosubjectsdf['subject'].iloc[0]
                ]
```
• Adding the 650 with the FAST heading from Annif.
Concluding thoughts

• SVDE data has been useful for several non-traditional uses for Linked Data experimentation and for hybrid linked data discovery.
  • Subject Indexing
  • Knowledge Panels for author agents – bringing in related works and biographical assertions
Thank you!

- jimhahn@upenn.edu
Acknowledgement

This presentation contains information from FAST (Faceted Application of Subject Terminology) Data which is made available by OCLC Online Computer Library Center, Inc. under the ODC Attribution License.