Building Your Own OpenRefine Reconciliation Service

Code4Lib MDC 2015

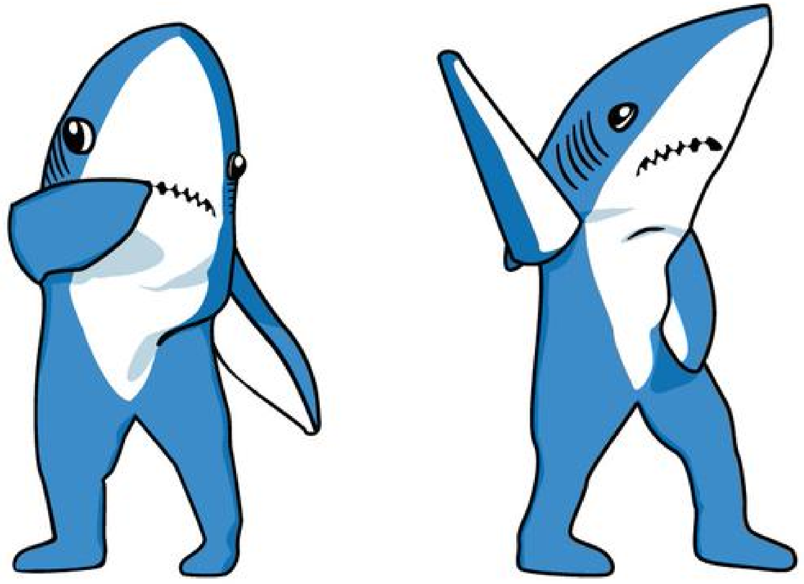
About This Workshop

Christina Harlow, @cm_harlow

Group Questions - What do you want to know

From perspective of a data munger, not a developer

Left-sharking It



LibTech Developers to Thank

Chad Nelson, @bibliotechy

Ted Lawless, @lawlesst

Trevor Muñoz, @trevormunoz

Owen Stephens, @ostephens

Slides, Examples, + Install

Slides, Examples, Prompts, Sample Data:

https://github.com/cmh2166/c4lMDCpres

Installation:

http://openrefine.org/download.html

DERI RDF Extension:

http://refine.deri.ie

Agenda

- Very brief Intro to OpenRefine
- OpenRefine Reconciliation options
  - Add column by fetching URL...
  - Standard Recon Service API
  - DERI RDF Extension
- Breakouts

Quick Intro to OpenRefine

- OpenRefine = power data tool
- Since 2012, its community-sourced
- OpenRefine.org, github.com/OpenRefine/Openrefine
- Java (and Jetty) tool that runs locally
- GUI runs in your chosen browser

OpenRefine Functionalities

- Import/Export Options
- Cleans
- Facets
- Clusters
- GREL
- Extensions
- Can be a breakout

OpenRefine Reconciliation

Reconciliation broadly: Compare values in my dataset with values in an external dataset, if deemed a match, link and pull in external datapoint information

OpenRefine Recon Options

- Add column by fetching URL...
  - HTTP requests to external data API in UI
  - takes far longer to pull data
  - requires parsing returned data with GREL
- Standard Recon Service API
  - RESTful API between OpenRefine and external data
  - requires tinkering knowledge of API building
  - can host for easier use
- DERI RDF Extension
  - no longer actively supported
  - Standard Recon Service API to work with RDF, SPARQL endpoints
  - RDF docs held in memory
  - SPARQL recon dependent on SPARQL server details

Add column by fetching URL...

Access an external data set by building an external data API queries for each cell, then adding a column with the results of that API call, then parsing the results for the match.

- 'Edit column' > 'Add column by fetching URLs'
- Give the results column a name
- Enter the GREL to create the API URL query, then 'Add column'
- Wait possibly a very, very long time
- Use GREL on the results column to parse response

Examples: Add column by fetching URL...

- Check 'addcolumnexamples.md' in this workshop's GitHub repo.
- Also review the Mountain West Digital Library workflow using this method with the Geonames API

Standard Recon Service API

OpenRefine Standard Reconciliation Service takes UI data, queries external dataset, then handles ranking, normalization, and returning results to UI.

= HTTP-based RESTful JSON-formatted API connecting OpenRefine to external datasets. This API can be constructed in a number of languages and frameworks.

Originally based off of Freebase extension (no longer working).

Standard Recon Service API Parts

- Recon Service Endpoint
  - GET to send service info to OpenRefine
  - POST to query data API for matches
- Recon Service Metadata
- Entity 'Types'
  - Freebase holdover
- Query/Response Handling
- Other bells and whistles

Recon Service API Metadata

"When a service is called with just a JSONP callback parameter and no other parameters, it must return its metadata as a JSON object literal with at least 3 fields 'name', 'identifierSpace', and 'schemaSpace'. Other fields are optional for reconciliation services which can make use of the default Freebase preview, suggest, etc services, but non-Freebase reconciliation services may need to implement them all." -- https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API

API Metadata Example Part 1

```
{
  "name" : "Reconciliation Service Name",
  "identifierSpace" : "http://rdf.freebase.com/ns/some.name.space",
  "schemaSpace" : "http://rdf.freebase.com/ns/type.object.id",
  "view" : {
    "url" : "http://www.externaldatasource.org//{{id}}"
  },
  "preview" : {
    "url" : "http://this-api.freebaseapps.com/preview/{{id}}",
    "width" : 430,
    "height" : 300
  },
```

API Metadata Example Part 2

```
  "suggest" : {
    "type" : {
      "service_url" : "http://this-api.freebaseapps.com",
      "service_path" : "/suggest_type",
      "flyout_service_url" : "http://www.freebase.com"
    },
    "property" : {
      "service_url" : "http://this-api.freebaseapps.com",
      "service_path" : "/suggest_property",
      "flyout_service_url" : "http://www.freebase.com"
    },
    "entity" : {
      "service_url" : "http://this-api.freebaseapps.com",
      "service_path" : "/suggest",
      "flyout_service_path" : "/flyout"
    }
  },
  "defaultTypes" : []
}
```

Query JSON Example

```
{
  "query" : "Kittens",
  "limit" : 3,
  "type" : "/fast/all",
  "type_strict" : "any"
}
```

Reconciled JSON Example

```
{
    "result" : [
      {
        "id" : "http://id.loc.gov/authorities/subjects/sh85072589"
        "name" : "Kittens"
        "type" : "/fast/all"
        "match" : "false"
      },
      ... more results ...
    ]
  }
```

Entity Types

These are the types of entities for reconciliation, usually based on the result

Depends on the service and is optional. Freebase holdover that can be used to access different indexes for an external data API.

Standard Recon Service API Templates

Some wonderful developers made templates in a variety of languages, though python/flask seems to be the language/framework du jour, for folks to plug in external data API information. These will get a basic API service up and running fairly quickly for those with limited programming knowledge or time.

Standard Recon Service Examples

- FAST Reconciliation Service (not hosted, using python)
  - Check 'pythonflaskmarkedupexample.md' in this workshop's GitHub repo.
- LCNAF - VIAF Reconciliation Service (hosted and not hosted, using python)
- VIAF Reconciliation Service Example (hosted, using PHP)
- Basic Python/Flask Template
- Extended Python/Flask Template

DERI RDF Extension Recon

Using this, can reconcile against RDF documents and SPARQL Endpoints.

No longer actively supported.

Breakouts

Each Recon Service has instructions for using existing and/or building your own. Working in groups/alone/however, go ahead and start experimenting or building. Additionally, folks can work on improving the documentation mentioned above.

Links + Contact

http://openrefine.org/

http://github.com/openrefine/openrefine

https://groups.google.com/forum/#!forum/openrefine

@openrefine, @cm_harlow