Linked Open Research Data for Earth and Space Science Informatics

Contents

Linked Open Research Data for Earth and Space Science Informatics

Tom Narock and Eric Rozell

A 2011 ESIP Funding Friday Project

Abstract: Earth and Space Science Informatics (ESSI) is inherently multi-disciplinary, requiring close collaborations between scientists and

information technologists. Identifying potential collaborations can be difficult, especially with the rapidly changing landscape of technologies and informatics projects. The ability to discover the technical competencies of other researchers in the community can help in the discovery of collaborations. In addition to collaboration discovery, social network information can be used to analyze trends in the field, which will help project managers identify irrelevant, well-established, and emerging technologies and specifications. This information will help keep projects focused on the technologies and standards that are actually being used, making them more useful to the ESSI community.

We address this problem with a solution involving two components: a pipeline for generating structured data from AGU-ESSI abstracts and ESIP member information, and an API and Web application for accessing the generated data. We use a Natural Language Processing technique, Named Entity Disambiguation, to extract information about researchers, their affiliations, and technologies they have applied in their research. We encode the extracted data in the Resource Description Framework, using Linked Data vocabularies including the Semantic Web for Research Communities ontology and the Friend-of-a-Friend ontology. Lastly, we expose this data in three ways: through a SPARQL endpoint, through Java and PHP APIs, and through a Web application. Our implementations are open source, and we expect that the pipeline and APIs can evolve with the community.

Related Links

Software Repository: Google Code (http://code.google.com/p/linked-open-data-essi/)

ESSI Linked Data: http://essi-lod.org/instances (http://essi-lod.org/instances)

owl:sameAs Identifier: owl:sameAs List (http://wiki.esipfed.org/index.php/LORD_ESSI_sameAs)

Useful Tools

DBPedia Spotlight (http://spotlight.dbpedia.org):

- text - the text you want annotated
- confidence - a threshold for terms that are annotated. upon annotating a term, the spotlight service examines the surrounding text and examines if the annotation makes sense given the context. the user supplied confidence value tells spotlight how confident it should be in the annotation, i.e. only return annotations with a confidence greater than or equal to the supplied confidence
- support - the minimum number of inlinks a Wikipedia page must have for annotation
- Example Service Call (http://goo.gl/2q7Nl)

OpenCalais (http://www.opencalais.com/GetStarted)

Workflow Documentation

For the latest workflow documentation, please go to ESSI-LOD Python Framework.

Step 1: Scrape abstracts from AGU

- Summary
  - Crawls the abstract listings from: http://www.agu.org/focus_group/essi/essi_search.html
  - Parses the HTML into Java Objects
  - Serializes objects into specified format

- Dependencies
  - SVN
  - Maven 2.*

- Download the source repository:

```
svn co https://linked-open-data-essi.googlecode.com/svn/ linked-open-data-essi
```

- Compile the Java source:

```
 mvn install
```

- Run the web crawler:

```
 mvn exec:java -Dexec.mainClass="org.agu.essi.data.AguSessionCrawler" -Dexec.args="--outputDirectory
path/to/dir --outputFormat [xml|rdf/xml]
```

Step 2: Convert scraped data to RDF

- Java code base accepts "rdf/xml" as an output format, meaning there are two options...
- Option 1: Use default "rdf/xml" generation from code base
  - Same as procedure in step one, using "--outputFormat rdf/xml"
- Option 2: Use XSLT to make custom transformations of our XML output into RDF
  - Our XML Schema Declaration is located at: http://www.agu.org/focus_group/essi/schema/agu.xsd

Step 3: Convert ESIP membership and meeting attendance data to RDF

- run org.esipfed.data.MeetingAttendance
- Inputs
  - 0 = Directory containing ESIP membership data
  - 1 = Output path and file name for FOAF Person RDF file
  - 2 = Output path and file name for FOAF Organization RDF file
  - 3 = Output path and file name for FOAF Meeting RDF file

Step 4: Create mappings between ESIP data and AGU data

- We need to create mappings between instances (i.e., people) using either a manual or automated process.
- We will likely use "owl:sameAs" statements as mappings; it is still up for debate whether we will use a sameAs reasoner.
- Option 1: Automated Mapping
  - Create a modular system for asserting equivalence mappings (probabilistically?)
    - System could be trained on a small amount of labeled data
    - Could be based on simple heuristics (e.g., email and co-authors for people, cities and authors for affiliations)
- Option 2: Crowd-Sourcing
  - Create a tool that requests authors or organizations by name and supports equivalence assertions/deassertions

Step 5: Annotate converted data with named entities in abstracts

- Determine appropriate Spotlight settings (e.g., confidence and support)
- Option 1: Use default annotation service and surface form dictionary
  - For each abstract
    - Feed abstract to Spotlight service
    - Add triples for disambiguated entities
    - Add triples for provenance (e.g., confidence and support)
- Option 2: Use disambiguation service and use Microsoft Web N-Grams Service for surface forms
  - For each abstract
    - Use Web N-Grams Service and surface forms dictionaries to identify surface forms
    - Update abstracts with surface forms
    - Add triples for provenance (e.g., N-Gram probabilities)
    - Feed abstracts with surface forms to disambiguation service

    - Add triples for disambiguated entities
    - Add triples for provenance (e.g., confidence and support)

Step 6: Load annotated data in a persistent triple store

- Candidates: Virtuoso (http://virtuoso.openlinksw.com/), TDB (http://openjena.org/TDB/), OWLIM (http://www.ontotext.com/owlim), AllegroGraph (http://www.franz.com/agraph/allegrograph/), ... (not exhaustive)

Step 7: Apply Linked Data Publishing Tool

- Candidates: LODSPeaKr (http://lodspeakr.org/)

Evaluation

- Option 1: Abstract clustering from disambiguation data
    - Summary: Use the disambiguations as sparse feature sets to cluster AGU abstracts. Perform a purity test based on occurrence of tags within clusters.
    - Related Work: Document clustering of scientific texts using citation contexts (http://www.springerlink.com/content/p278617582u5x3x1/)

- Option 2: Precision and recall using human-annotated abstracts
    - Summary: Request annotations from past AGU presenters. Perform disambiguation and evaluate precision and recall on entities.
    - Related Work: DBpedia Spotlight: Shedding Light on the Web of Documents (http://i-semantics.tugraz.at/scientific-track/accepted-papers)

Open Issues

Unique identification of organizations

- Summary: We would like to be able to identify organizations that show up in multiple publications.
- Use Case: Eric would like to find all abstracts at AGU written by people in affiliation with Woods Hole Oceanographic Institution.
- Problem: AGU affiliation data is unstructured. We would need to separate the research group, the department, the organization, and the address using some heuristics.
- Solutions:

1. We could use the Google Geocoding service to get a lat/long for the unstructured address. We can then use this lat/long with the GeoNames service to get URIs of nearby points of interest (e.g., cities, counties, universities). Unfortunately, this solution will not result in 100% precision. It also does not address the issue of sub-organizations, such as departments and research groups.

- Discussion:

Resolved Issues

Use of roles in ontology and RDF data

- Summary: The SWRC ontology treats classes like Employee and Student as subclasses of Person. This is an inaccurate representation of reality, as Employee (or Student) is a role played by a particular Person. See the use case that follows.
- Use Case: Eric Rozell was an employee of Microsoft Research. In one of his publications, his affiliations included both Microsoft Research and RPI. In a later publication, after leaving Microsoft Research, Eric's affiliation should only be RPI.
- Problem: If a person is (at some point in time) affiliated with an organization, they will be affiliated in all scenarios. For listing publications with accurate affiliations, we want only the specific affiliations listed for that publication, not all affiliations for the person.
- Solutions:

1. Use the Tetherless World Constellation ontology, where affiliations are attached to specific roles played by people.
2. Create a unique instance for each combination of affiliations needed for a specific person.

- Discussion:
    - Tom: option 2 doesn't seem ideal, and probably not scalable. option is fine with me, but would this ontology break the mobile app, which currently runs on SWRC? Also, is the TWC ontology available online?
    - Eric: This would likely break the mobile app, but the updates required should be simple enough. The TWC ontology is available online at http://tw.rpi.edu/schema/.
    - Eric: UPDATE - The new mobile app that TWC is working on allows for configuration, so we can configure the display of authors by writing a SPARQL query that will select the person and affiliations for each author role.

- Resolution: Applied Solution 1

Representation of AGU Sections (e.g., ESSI)

- Summary: What sort of thing is an AGU Section with respect to the SWRC ontology?
- Problem: We need to identify whether AGU Sections have a correspondent class in SWRC, or if we should create a new class.
- Solution:

1. Find a SWRC class that corresponds to AGU Sections. If an AGU meeting is a swrc:Meeting, then the ESSI section could also be a swrc:Meeting (and with property swc:isSubEventOf the AGU meeting), and finally a session is a swc:SessionEvent (with property swc:isSubEventOf the AGU Section).
2. Create a new class for AGU (and potentially other meeting) sections.

- Discussion:
    - Tom: this depends on if we use SWRC or switch to TWC ontology as mentioned above. How would AGU Sections fit into the TWC ontology?
    - Eric: I'd like to use terms from both SWRC and TWC. I've updated option 1 to reflect some terms that we can use. I'm hesitant to call the AGU Fall Meeting a conference (so I chose swrc:Meeting instead).

- Resoultion: Applied Solution 1

AGU Abstract Special Characters Encoding

- Summary: The AGU abstract HTML uses something other than UTF-8 for a character encoding.
- Problem: The data we parse from the abstract HTML does not use UTF-8 character encoding. Many XML tools only support UTF-8 encodings.
- Solutions:

1. Use GNU libiconv to convert the files at the command line after they are generated.
2. Use a Java solution, here is some pseudocode:

```
byte[] bytes = origString.getBytes("UTF-8");
String convertedString = bytes.toString();
```

- Discussion
    - Eric: I believe that the AGU HTML character encoding is CP1252. I have successfully implemented the Java solution above.
    - Eric: Another issue is the XML parser for a triple store candidate. It doesn't support the use of some of the "&<charName>;" entity encodings that are produced by the Java HTML encoder. We could delete these characters with a special function (read- hack), or we could use a different triple store that has a better XML parser.

- Resolution: Applied Solution 2

Crawler is crawling links which do not lead to abstracts

- Problem: The crawler currently throws exceptions when it tries to crawl links that do not lead to abstracts.

- Solutions:

1. Try to detect bad links before sending them to the abstract parser.
2. Create a new class of Exception when the abstract parser determines the HTML is not for an abstract.

- Discussion:
  - Tom: I vote for option 2. I think we should add an output log feature that would capture these exceptions and the urls that caused them. If its an issue with the AGU system outputting incorrect urls then this would give us specific cases to take back to them.

- Resolution: Applied Solution 2, created a subclass of java.lang.Exception (org.agu.essi.util.exception.AbstractParserException)

Matching Equivalent Keywords

- Summary: Keywords often have multiple numbers associated with them.
- Problem: The scheme for assigning numbers to keywords is not currently understood by members of this project.
- Solution:

1. Parse the keywords in the AGU index terms list (http://www.agu.org/pubs/authors/manuscript_tools/journals/index_terms/), citing the hierarchical relationships and "related" relationships (see discussion below) using the SKOS vocabulary. Here is an example RDF description of one the AGU index term "3252 Spatial analysis (0500, 1980, 4319)"

```
<rdf:Description rdf:about="&esip;Keyword_3252">
  <rdf:type rdf:resource="&swrc;ResearchTopic" />
  <dc:identifier rdf:datatype="&xsd;string">3252</dc:identifier>
  <dc:subject rdf:datatype="&xsd;string">Spatial analysis</dc:subject>
  <skos:broadMatch rdf:resource="&esip;Keyword_3200" />
  <skos:related rdf:resource="&esip;Keyword_0500" />
  <skos:related rdf:resource="&esip;Keyword_1980" />
  <skos:related rdf:resource="&esip;Keyword_4319" />
</rdf:Description>
```

- Discussion:
  - Eric: Here is what the Keyword looks like (sometimes): #### Phrase (####, ...), concrete example "3252 Spatial analysis (0500, 1980, 4319)"
  - Eric: Tom pointed me to the official list of index terms (http://www.agu.org/pubs/authors/manuscript_tools/journals/index_terms/) for AGU publication. Tom also inferred that the identifiers in parentheses are "related" terms.

- Resolution: Applied Solution 1

Implementing AGU Abstract Classes from Heterogeneous Sources

- Summary: We now have a variety of different sources that can contain information about an AGU abstract (e.g., HTML from AGU, XML from this project, linked data, and SPARQL services). We need an "interface" that specifies the behavior of an AGU Abstract, regardless of its source.
- Problem: Our initial solution had a single class implementing the AGU Abstract, and depended on different constructors to specify how the abstract would be parsed. However, the HTML-based constructor used a single parameter constructor ("public Abstract(String html)"), which blocked any other source from using String as an input to the Abstract constructor.
- Solutions:
  - Using a Java interface: We can create an interface for AGU abstracts that can be implemented as new data sources emerge.
  - Using a Java abstract class: We can create an abstract class for AGU abstracts (sorry for the overlapping terms) that can be extended as new data sources emerge.

- Discussion:

  - Eric: A major benefit for the abstract class is the ability to have default implementations for things like XML and RDF serializations. A major detriment against the abstract class is the requirement for the abstract class to be used as a base class (limiting the multiple inheritance capability).

- Resolution: Applied Solution 2.

Issues With AGU Abstract Database

- Summary: The way we are using the AGU abstract database does not return all abstracts.
- Solutions:

1. Tom has contacted AGU about this, and they offered to provide a dump of the abstract data.
2. Implement a solution that requests smaller numbers of abstracts (e.g., abstracts by day, abstracts by session)

- Discussion:
  - Eric: Tom has implemented a solution to request smaller numbers of abstracts by iterating through days of the meetings
  - Eric: I have implemented a solution that parses section pages (e.g., this (http://www.agu.org/cgi-bin/sessions5?meeting=fm09&sec=IN)) and requests abstracts from individual sessions

- Resolution: We have implemented two candidates for Solution 2.

Keeping unique IDs consistent across iterations of the pipeline

- Summary: Coining unique IDs for the first time is easy, we need to enable the reuse of those IDs when future data is added.
- Use Case: I've coined a unique ID for the person, Eric Rozell. Later, when encountering the same person in future AGU data, I'd like to use that original unique ID.
- Problem: How do we ensure that the IDs we coin now are reused when future data is added via the pipeline?
- Solutions:

1. Do not worry about reusing IDs when future data is run through the pipeline, instead perform post-analytics to determine "same as" relationships. Still, we need to perform collision detection for the URIs that are coined.
2. Load all the past data before running the pipeline on the new data and perform identification as usual.
3. Use URI conventions. For example, the name "Eric Rozell" might yield the URI esip:Eric_Rozell.
4. Create a new interface for specific data sources, which is capable of both matching entities and coining new URIs. For instance, based on the "Crawler" data source for AGU, the Crawler would run and pull all the abstract data into memory. Using this body of abstracts in memory, it can match people, organizations, sessions, etc. based on the given information. To clarify, another example might be the SPARQL endpoint data source. It is not likely that all the RDF data behind the SPARQL endpoint would have to be pulled into memory, since queries could be constructed to do the necessary matching.
5. Create an interface for "matching" sources. This is similar to Solution 4, however, it distinguishes data sources, which are actually used to populate Java instances, from matchers, which are only used to determine if unique identifiers already exist for an entity. We can use a SPARQL-based solution to determine if there already exist instances for an entity.

- Discussion:
  - Tom: does option 2 imply that the IDs from past runs will change when pipeline is run on new data? If so, I don't think that's an appropriate solution.
  - Eric: Option 2 implies that we will first load all the IDs from the past, and use those old IDs in the old data, so IDs should remain consistent. It will be unfortunate if we have to use "same as" entailment in virtually every use case we have, so I'd vote for option 2 over option 1.
  - Eric: I've added Option 3, but it does not seem like a reasonable solution. It is likely that it will create too many false positives.
  - Eric: I've also added Option 4. I am in favor of this solution. It is similar to solution 1, but extends the solution with the possibility that all prior data does not need to be loaded (instead it could be queried). This is a more generic solution that can reuse past data that has been converted and adapt to new sources of AGU abstracts. One of the downsides is that it requires auxiliary matchers (such as matchers for the ESIP membership data) to reuse the classes from the AGU code.

    - Eric: I'm going with Option 5. It has been successfully tested on data from the AGU IN section. It has also been deployed for data across all AGU sections. See the package org.agu.essi.matcher.

- Resolution: Applied Solution 5.

Representing Named Entity Annotations from DBPedia Spotlight

- Summary: After we send abstract text to DBPedia Spotlight, we need a way of capturing the results in RDF.
- Use Case: Ideally, I would only need to run the abstract text on DBPedia Spotlight once, the results of which can be reused in a variety of information retrieval applications.
- Problem: There is no vocabulary for associated RDF resources with annotated equivalents (i.e.,, an esip:Abstract and its annotated counter-part). There is also no vocabulary for describing annotated text in general.
- Solutions:

1. Coin a URI for associating any RDF Resource with an annotated equivalent. Coin a URI for representing a general class of DBPedia Spotlight annotated text. Coin a URI for representing individual text annotations from DBPedia Spotlight. Coin URIs for capturing all the necessary connections in annotated text and individual annotations.
2. Use the annotation vocabulary Tom found (*Tom can you please update this solution*).

- Discussion:
    - Eric: I think the URIs we coin here should not belong to the ESIP namespace. I think we should pick a new namespace (at purl.org) and come up with a name, acronym, and prefix for the ontology of Spotlight annotations. We can then contribute this back to the Spotlight project and also to the NLP2RDF project (http://aksw.org/Projects/NLP2RDF?v=3sg).

To Do/Other Ideas

- To Do
    - Major
        - create RDF of ESIP meetings (currently we have only people and their associated organizations - we don't have RDF describing the actual meetings)
        - SPARQL data source implementation
        - Implement the ultimate test: Crawl data as RDF, load in SPARQL endpoint. Load data from XML source, use SPARQL endpoint as "URI" source, compare the URIs generated for people, organizations, keywords, meetings, sections, sessions, etc.

    - Minor
        - create separate Maven projects for the content in org.agu.essi and org.esipfed
        - log4j is not initialized properly in the java code
        - CSV output for abstracts
        - finish and test geocoding/geonames service
        - Fix main classes for DataSources (use better collision detection for directories/file writing)
        - Change all occurrences of System.err.println to commons logging

- Other Ideas
    - ESSI keywords were introduced in Fall 2009. Maybe we should set up a web form to allow authors to go back and annotate older abstracts with ESSI keywords.
    - Chris Lynnes suggested measuring ESIP's impact - need to think of how to do this using our data.
    - Peter mentioned uncovering hidden/non-explicit network
    - could the rdf statements be extended with time tags (owl time) to infer the esip/essi network at different points in time?

Notes

- License
    - This code is released under GPL version 3. A copy of the GPL license, as well as licenses of imported code, can be found in the

License directory.
- We use the Maven license plugin to manage annotating the source code with license information
  - The file resources/header.txt is a template for the maven license plugin. The text in this file will be added to the top of each source code file when the plugin is run.
  - Running the plugin (year should be replaced with the current year. this value is inserted into the copyright field of the GPL text)
    - Verify that all source code files have the required license text: mvn verify -Dyear=2011
    - Add license text to all source code files that do not already contain it: mvn license:format -Dyear=2011

Retrieved from "https://wiki.esipfed.org/w/index.php?title=Linked_Open_Research_Data_for_Earth_and_Space_Science_Informatics&oldid=44375"