CREATING A **FAIR** DATA CATALOG TO SUPPORT SCIENTIFIC MODELING

*Basel Shbita, Binh Vu, Minh Pham, Dan Feldman*
*Craig Knoblock, Jay Pujara, Yao-Yi Chiang*
Motivating Example

• Finding, preparing, and cleaning datasets dominate time devoted to scientific inquiry
  – Example: flooding prediction takes **months** for data prep.

• Need a Data Catalog
  – **Retrieve datasets** by variables (e.g., SVO) and other metadata (temporal and geospatial coverage)
  – **Transform dataset** for different use cases with minimal effort
    • Reprojection, cropping, or format conversion
    • Joining multiple datasets
  – **Visualize dataset** to get insight of the data
Challenges in building the Data Catalog

• Registering data in the data catalog
  – Huge number of datasets with **massive amount of data**
  – Lots of **manual effort** for curating the datasets
    • Ambiguous data definition: different communities using **diverse terms** for same phenomena
    • Dataset are stored in **different formats** and layouts: NetCDF, CSV, spreadsheets...

• (Semi)-automatic data transformation
  – Generate & execute a transformation plan based on input and **desired output**
  – Easy to **reuse** existing transformation libraries or add new transformation
Overall approach

Data Transformation

MINT Dataset Registration

Data Catalog

Registration

Unit detection

Fuzzy Search

Automated table analysis

Building dataset representation

Information Sciences Institute
Registering data

• Dataset registration
  – Can be done via API or UI
  – Provide basic metadata: variable, spatial and temporal coverage

• Finding correct variable (fuzzy search)
  – Semantic knowledge in Wordnet
  – Statistical associations with Word2Vec
  – Topic modeling and other string similarity metrics
Building rich data understanding

• Automated table analysis [2]
  – Input: tabular datasets
  – Output: dataset layout
    • Header/Attribute/Value blocks
    • Their relationships
Building rich data understanding (cont.)

- Dataset representation (D-REPR [3]). Why?
  - Different **formats** (NetCDF, spreadsheet, CSV, JSON)
  - Different **layouts** (matrix tables, hierarchical tables)
  - **Same interface** to access to many kinds of datasets

- Automatically generate D-REPR file
  - Table understanding
  - D-REPR GUI for curation
Building rich data understanding (cont.)

- Four steps to create a D-REPR file
  1. Specify resources
  2. List attributes (or variables) in the datasets
  3. Simple rules to join values of the attributes
  4. Map attributes to predicates, classes in domain ontologies
Building rich data understanding (cont.)

- **Unit detection (CCUT [1])**
  - **Identify**, parse, and map compound units of measurement to QUDT ontology (**semantic representation**)
  - Enable **automatic** unit conversions
  - Enhance the dataset **representation**
Querying data from data catalog

- Search datasets (API-based) by
  - dataset name
  - variables
  - temporal coverage
  - spatial coverage (bounding box)
- Return a subset of data in different format
  - Using data transformation component

Example queries can be found in the [Api Demo](#) notebook.
Data Transformation

• Construct **transformation pipeline**

• **Structured Representation**
  – Uses D-REPR representation
    • Supports **complex data** (N-dim, data across multiple files)
    • Captures and leverages a **semantic meaning** of the data
    • Can be mapped to any **ontology** (i.e. SVO, DataCube)
  – Makes the data **format-independent**

• **Building blocks** architecture

• **Building blocks** architecture
  – Easy to **reuse** existing building blocks
Data Transformation Architecture

• Adapters (building blocks/‘components’)
  - Three types:
    • Reader (entry point)
      - Reads input file (data) and description of it (variables, relations, semantics)
    • Transformer
      - Does not materialize the data, just reproduces it
    • Writer (exit point)
      - Writes output file (data) based on a description of it (variables, relations, semantics)
  - Enable input data validation and compatibility checking

• Pipeline
  - Define the required inputs for some adapters
  - Wire some inputs to outputs (‘concatenate’ the components)
  - Execute the pipeline!
1. Explore adapters

2. Construct pipeline
Data Transformation UI (cont’d)

Apply (online) updates and modifications

3. Execute pipeline

Generates output file

(optional) Save/Load pipelines

Clear pipeline

Load from a ready config file

Save to config file
Future work

• Dataset Registration
  – **Data discovery** for finding and adding more datasets
  – **Registering high volume** of data sitting behind servers (RESTful API)
  – Improve **automatic data understanding**
    • Table understanding
    • Unit Detection

• Dataset Query
  – GUI for **browsing and visualizing datasets** in the Data Catalog

• Data Transformation
  – Generating transformation plan (semi-)**automatically** (“transformation reasoning”)
Summary

• The Data Catalog allows dataset registration, search, transformation and visualization

• Automated tools to support adding additional meta-data to the Catalog

• Transformation framework that allows constructing, executing and validating a transformation pipeline
  – Easy to use and easily extended (no need to re-implement a complete flow)

• More information can be found at
  – https://mint-project.org
  – https://github.com/mintproject
References

• [1] B. Shbita, A. Rajendran, J. Pujara, and C. Knoblock, Parsing, Representing and Transforming Units of Measure, in Modeling the World’s Systems, 2019