Creating a FAIR Data Catalog to Support Scientific Modeling

Basel Shbita
Information Sciences Institute
University of Southern California
Marina del Rey, CA, US
shbita@isi.edu

Binh Vu
Information Sciences Institute
University of Southern California
Marina del Rey, CA, US
binhvu@isi.edu

Dan Feldman
Information Sciences Institute
University of Southern California
Marina del Rey, CA, US
danf@usc.edu

Minh Pham
Information Sciences Institute
University of Southern California
Marina del Rey, CA, US
minhpham@usc.edu

Arunkumar Rajendran
Information Sciences Institute
University of Southern California
Marina del Rey, CA, US
arunkumr@isi.edu

Craig A. Knoblock
Information Sciences Institute
University of Southern California
Marina del Rey, CA, US
knoblock@isi.edu

Jay Pujara
Information Sciences Institute
University of Southern California
Marina del Rey, CA, US
jpujara@isi.edu

Yao-Yi Chiang
Information Sciences Institute
University of Southern California
Marina del Rey, CA, US
yaoyic@usc.edu

***Abstract*—Scientific models often depend on complex, interrelated datasets, and finding, preparing, and cleaning these datasets often dominates the time devoted to scientific inquiry. We are addressing these problems by creating a *Data Catalog* that provides a central clearinghouse for metadata about scientific datasets, supports fuzzy searching for data variables using NLP techniques, provides a number of automated, data-understanding tools to make data curation easier, and automates the processes for transforming, joining, and formatting datasets for different use cases. In this abstract, we provide a short description of the salient components of our system.**

I. Introduction

Scientific inquiry frequently relies on data. Finding the correct data and subsequently preparing this data by cleaning, reformatting, and transforming it are extremely laborious and time-consuming processes that hinder progress. Creating open repositories of datasets with the accompany semantic metadata and tools for allowing efficient preparation has the potential to dramatically improve the pace of scientific progress. We are currently building the knowledge technologies to support such a data repository, which we call a *Data Catalog*, drawing inspiration from the W3C Data Catalog vocabulary[1]. The Data Catalog consists of dataset metadata, such as temporal and geospatial scope, and pointers to existing data resources. A "fuzzy" search interface ensures a link between scientific ontologies and controlled vocabularies (such as the Scientific Variable Ontology[2]) and human-comprehensible terms. The main focus of the data catalog is to incorporate a *data specification* which can be populated using automated tools and improved with user feedback. These data specifications allow data to be cleaned, transformed, and formatted with minimal user effort. Through this design, the data catalog provides findable datasets, accessible metadata, interoperable data descriptions, and tools to foster reusable data. In the subsequent section, we describe each of the components of the Data Catalog and how they support FAIR data.

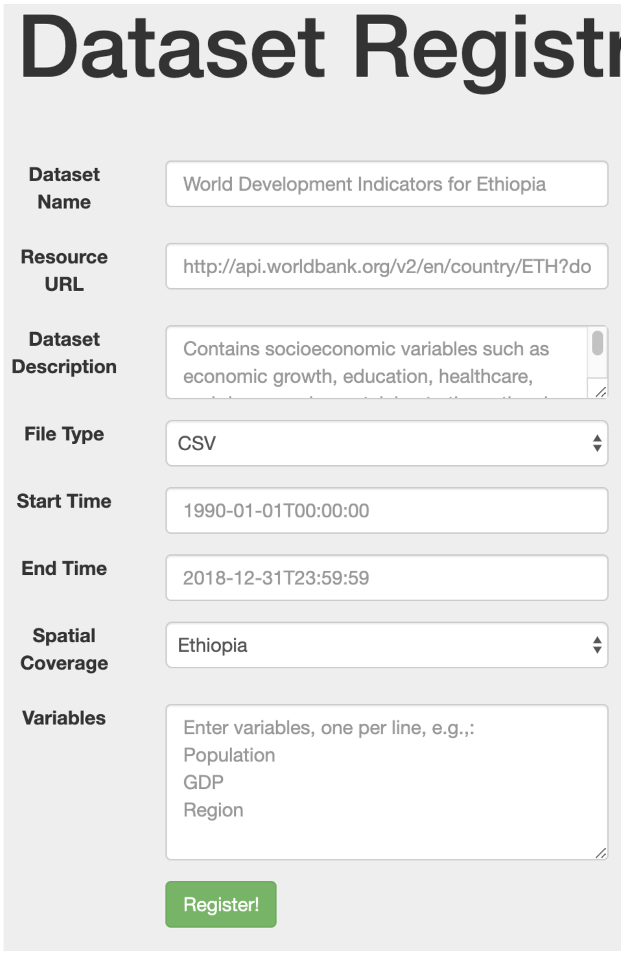
[1]https://www.w3.org/TR/vocab-dcat/
[2]http://www.geoscienceontology.org/svo/1.0.0/

II. Registering Data in the Data Catalog

Data is initially ingested into the Data Catalog through user submission. Figure 1 shows a simple registration interface for the data catalog, where the user provides a dataset name, a pointer to the resources within the dataset, the temporal and geospatial extent of the dataset, and the variables contained within the dataset. Although this simple interface collects the essential metadata, the data catalog provides an API that supports programmatic access and allows the user to register more complex datasets and provide more detailed metadata (e.g., datasets with many files or additional provenance information).

III. Finding Datasets with Fuzzy Search

Data registered in the data catalog is indexed by all of the provided metadata, allowing searches based on time, space, variable name, or a combination of these queries (among others). However, basic searches over metadata are often insufficient to meet the needs of a diverse use base. For example,


Dataset Regist

Dataset Name	World Development Indicators for Ethiopia
Resource URL	http://api.worldbank.org/v2/en/country/ETH?do
Dataset Description	Contains socioeconomic variables such as economic growth, education, healthcare,
File Type	CSV
Start Time	1990-01-01T00:00:00
End Time	2018-12-31T23:59:59
Spatial Coverage	Ethiopia
Variables	Enter variables, one per line, e.g.,: Population GDP Region

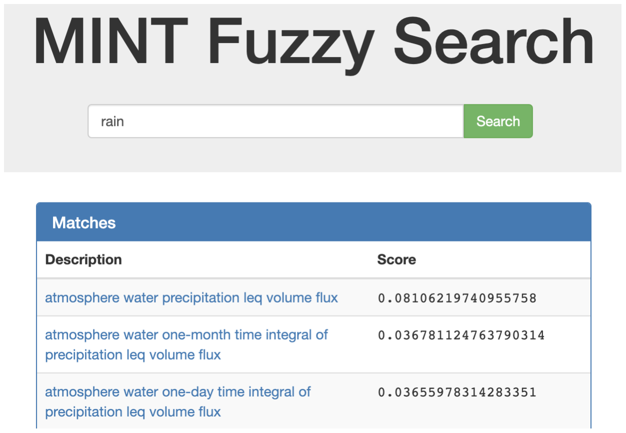
Register!

Fig. 1: Users can provide very basic metadata about a dataset to register it in the data catalog. An API provides more powerful, programmatic access to data registration

technical data definitions can be a barrier to data reuse, particularly when different communities use diverse terms for the same phenomena or a novice user is beginning to acquire data. To help mitigate these barriers, our data catalog supports a "fuzzy" search that allows the entry of human-interpretable keywords and connects them to datasets based on the scientific variable description and dataset contents. Figure 2 shows an example of how a user interested in finding data about rain is guided to the scientific variables for precipitation flux and provided with links to the formal definition of this term. The fuzzy search capability uses several strategies for finding related concepts, including semantic knowledge in WordNet, statistical assocations mined with Word2Vec, and topic models trained on scientific literature as well as scoring functions that use string similarity metrics and TF-IDF weighting.

IV. Automated Table Understanding Tools

Another barrier to FAIR data is the diversity of formats data is presented in. Although humans are adept at understanding the structural layout of datasets and mapping these into semantic data intuitively, generating linked data from a file is


MINT Fuzzy Search

rain

Search

Matches

Description	Score
atmosphere water precipitation leq volume flux	0.08106219740955758
atmosphere water one-month time integral of precipitation leq volume flux	0.036781124763790314
atmosphere water one-day time integral of precipitation leq volume flux	0.03655978314283351

Fig. 2: A "fuzzy" search allows users to find variables and data in a formal scientific ontology using human-interpretable keywords. The search uses Word2Vec, WordNet and topic models to suggest appropriate variables.

still a cumbersome process. To aid in this process, we have developed tools for automatically understanding structured, tabular data. Our tools first classify each cell or value in the dataset based on its expected role, then group related cells into functional blocks, and then finally try to predict the relationships between these functional blocks.Figure 3 illustrates the major steps in this process. For example, scientific datasets often contain numerical data, attributes, and metadata in different cells. These cells are spatially organized so that related values (e.g., temperatures) are grouped together while the associated attributes (e.g., location) are similarly grouped and have a distinct spatial relationship. Our table understanding system labels cells using conditional random fields, identifies blocks of related cells using a decision-tree style entropic measure, and predicts block relationships using link prediction also implemented as a conditional random field.

V. Identifying Units in Scientific Data

The identification of units of measurement that are associated with data is a challenging task because it requires having some domain knowledge about the process that produced the data. Frequently, units appear in files within datasets in a textual representation that is not easily recognized and does not carry any semantic or dimensional meaning. We implemented a prototype system, called CCUT [1], which uses grammar tools to automatically parse the different components in a unit found in textual data in files and map them to elements of a standard ontology called QUDT[3] to form a structured semantic output. The output depicts the different relationships, attributes and semantics of units and allows users to have a better understanding of their data. Figure 4 shows an example of a compound unit present in cell `B8`, `'A/cm^2'` (marked in a red box), in a spreadsheet file and its corresponding output. The evaluation of the system has demonstrated early results

[3]http://www.qudt.org/

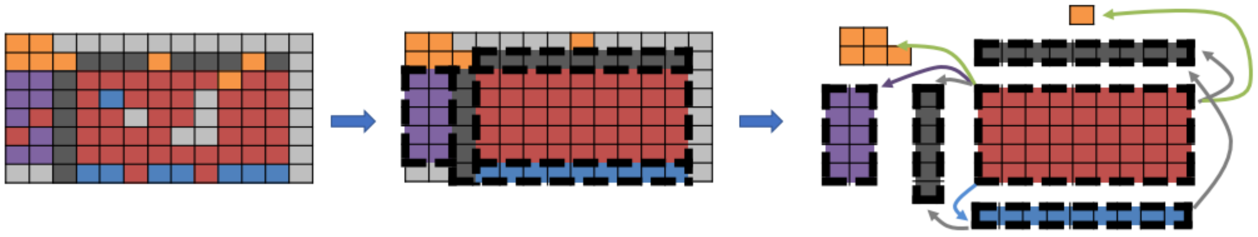
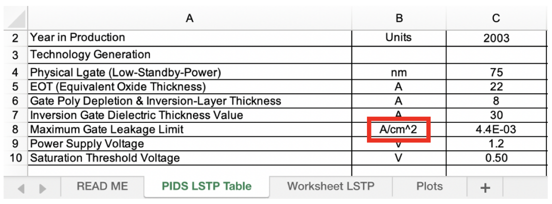


Fig. 3: An automated table understanding system is used to determine the format and structure of tabular, scientific datasets by classifying cell types, identifying regions of related cells, and determining layout relationships.

and can be beneficial for scientists to perform a fast process of data analysis and understanding.



	A	B	C
2	Year in Production	Units	2003
3	Technology Generation		
4	Physical Lgate (Low-Standby-Power)	nm	75
5	EOT (Equivalent Oxide Thickness)	A	22
6	Gate Poly Depletion & Inversion-Layer Thickness	A	8
7	Inversion Gate Dielectric Thickness Value	A	30
8	Maximum Gate Leakage Limit	A/cm^2	4.4E-03
9	Power Supply Voltage	V	1.2
10	Saturation Threshold Voltage	V	0.50

(a) a compound unit in `xls` spreadsheet

```
{
  "PEDS LSTP Table": {
    "B": {
      "8": [ {
        ccut:hasDimension: "L-2 I",
        ..
        ccut:hasPart: [
          {
            ccut:hasDimension: "I",
            qudtp:quantityKind: "http://data.nasa.gov/qudt/owl/unit#Ampere",
            qudtp:symbol: "A"
          },
          {
            ccut:exponent: "-2",
            ccut:hasDimension: "L",
            ccut:prefix: "http://data.nasa.gov/qudt/owl/unit#Centi",
            ccut:prefixConversionMultiplier: 0.01,
            qudtp:quantityKind: "http://data.nasa.gov/qudt/owl/unit#Meter",
            qudtp:symbol: "cm"
          }
        ]
      } ]
    }
  }
}
```

(b) partial representation of the detected unit (`'A/cm^2'`)

Fig. 4: An example of a detected compound unit and its representation

VI. Specifying Data Layouts

Shared datasets in the data catalog are often in different formats (e.g., CSV, Spreadsheet, JSON, or NetCDF) and in various layouts (e.g., row-based or matrix tables). Therefore, simple tasks such as extracting a subset of data or transforming units of variables are difficult and laborious. To address this problem, we have developed a data description language, D-REPR, for modeling datasets. Specifically, users describe attributes (or variables) and the locations of their values in a dataset. In D-REPR, the values of each attribute form a separated column array, to create a table that contains all records in the dataset, users define rules for joining values of these attributes together (i.e., combining columns to form the table). Finally, the semantic meaning of each attribute (or column) and the relationships between the attributes are specified using domain ontologies, which users can choose.

Given a D-REPR model of a dataset, we can convert data in the dataset into a common representation (e.g., RDF Graph or high-dimensional arrays) that allows us to transform or query the data easily.

VII. Combining, Transforming and Reformatting Datasets

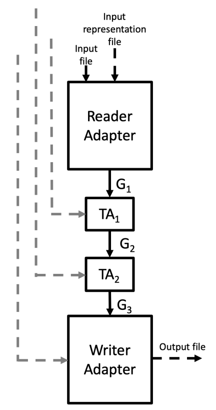
In order to combine, transform or reformat datasets, users must select, understand, and align them manually. To address this issue we implemented a framework which constructs a transformation pipeline based on some specification from users. The framework uses our data layout module presented in section VI to represent the actual data which may need to be transformed into one standard format for later uses.

The idea is that we use smaller components (we refer to them as adapters or building blocks) which we 'concatenate' to form a transformation flow. This modular design allows us to reuse existing modules and wrap ready-scripts to create a language-independent module and pipeline. There are three types of components (adapters):

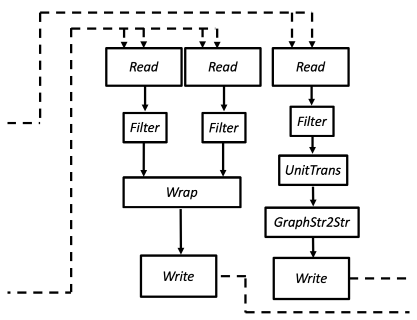
- Reader Adapter. Used as an entry point in the pipeline. It reads an input file (data) and a description of it (D-REPR language).
- Transformation Adapter. A class which performs a transformation done in a form of an API endpoint (remote or local) or an in-code script or library (i.e. using python or R). It does not materialize the data into an output, it just reproduces the data.
- Writer Adapter. Used as an exit point in the pipeline. It writes an output file based on a description file (D-REPR language)

Each adapter is declared using a semantic description of its attributes (i.e. inputs and outputs). The description enables input data validation and compatibility checking between the concatenated adapters and allows an easier construction of the transformation pipeline based on some simple input from the user. Figure 5 depicts the general idea of our architecture that is based on building-blocks and components that can be concatenated. In figure 5a we show a simplified scheme of a transformation pipeline involving a reader adapter, two transformation adapters ('TA') and a writer adapter. Figure

5b shows a transformation pipeline that involves two smaller pipelines that is used to create two output files. Each pipeline utilizes the different adapters to create files that are required to be in a clearly defined format and will be used by a specific software. For example, the component *UnitTrans* uses the CCUT service we mentioned in section V to perform a unit conversion on the data. Other components, such as *Wrap* and *GraphStr2Str*, are used to join and reformat the same resource without changing the actual content of the data.



(a) a simplification of a general transformation pipeline.



(b) an example flow involving different types of adapters.

Fig. 5: An abstraction of the data transformation pipeline

Acknowledgements: This material is based upon work supported by United States Air Force and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8650-17-C-7715 and award W911NF-18-1-0027.

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