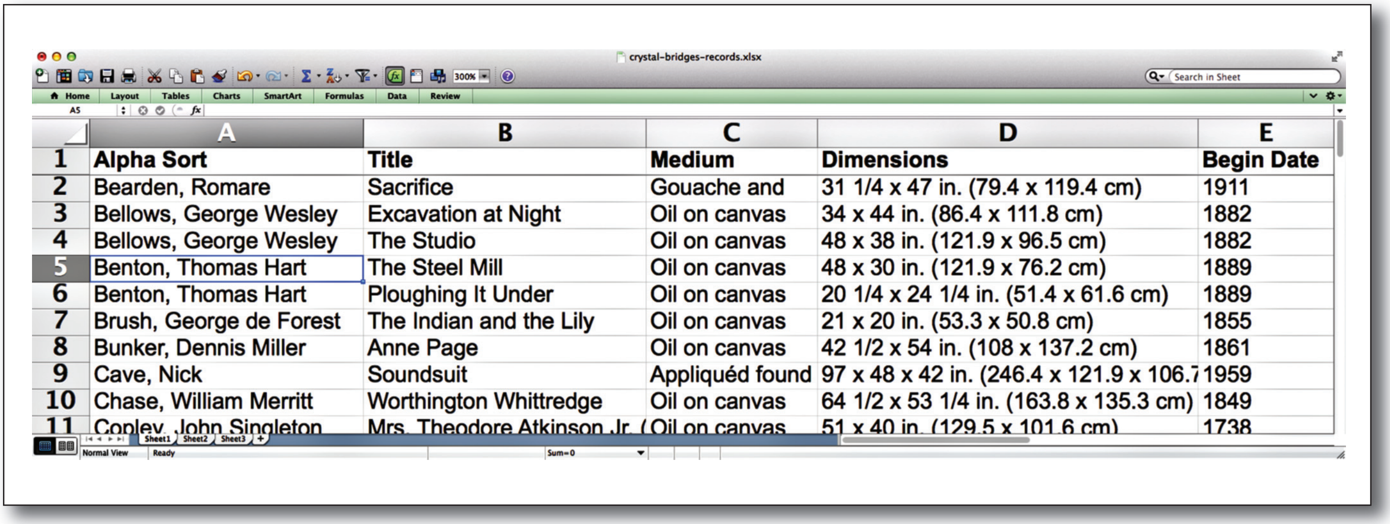
Exploiting Semantics for Big Data Integration

Craig A. Knoblock, Pedro Szekely

■ *There is a great deal of interest in big data, focusing mostly on data set size. An equally important dimension of big data is variety, where the focus is to process highly heterogeneous data sets. We describe how we use semantics to address the problem of big data variety. We also describe Karma, a system that implements our approach and show how Karma can be applied to integrate data in the cultural heritage domain. In this use case, Karma integrates data across many museums even though the data sets from different museums are highly heterogeneous.*

Big data is widely described as having three dimensions: volume, velocity, and variety. Volume refers to the problem of how to deal with very large data sets, which typically requires execution in a distributed cloud-based infrastructure. Velocity refers to dealing with real-time streaming data, such as video feeds, where it may be impossible to store all data for later processing. Variety refers to dealing with different types of sources, different formats of the data, and large numbers of sources. Much of the work on big data has focused on volume and velocity, but the problems of variety are equally important in solving many real-world problems.

In this article we focus on exploiting semantics to solve the problem of big data variety. In particular, we describe an approach to integrate data from multiple types of sources (for example, spreadsheets, relational databases, web services,



	A	B	C	D	E
1	Alpha Sort	Title	Medium	Dimensions	Begin Date
2	Bearden, Romare	Sacrifice	Gouache and	31 1/4 x 47 in. (79.4 x 119.4 cm)	1911
3	Bellows, George Wesley	Excavation at Night	Oil on canvas	34 x 44 in. (86.4 x 111.8 cm)	1882
4	Bellows, George Wesley	The Studio	Oil on canvas	48 x 38 in. (121.9 x 96.5 cm)	1882
5	Benton, Thomas Hart	The Steel Mill	Oil on canvas	48 x 30 in. (121.9 x 76.2 cm)	1889
6	Benton, Thomas Hart	Ploughing It Under	Oil on canvas	20 1/4 x 24 1/4 in. (51.4 x 61.6 cm)	1889
7	Brush, George de Forest	The Indian and the Lily	Oil on canvas	21 x 20 in. (53.3 x 50.8 cm)	1855
8	Bunker, Dennis Miller	Anne Page	Oil on canvas	42 1/2 x 54 in. (108 x 137.2 cm)	1861
9	Cave, Nick	Soundsuit	Appliquéd found	97 x 48 x 42 in. (246.4 x 121.9 x 106.7	1959
10	Chase, William Merritt	Worthington Whittredge	Oil on canvas	64 1/2 x 53 1/4 in. (163.8 x 135.3 cm)	1849
11	Copley, John Singleton	Mrs. Theodore Atkinson Jr. (	Oil on canvas	51 x 40 in. (129.5 x 101.6 cm)	1738

Figure 1. Crystal Bridges Data in an Excel Spreadsheet.

and others) and in widely different formats, including both relational and hierarchical data (that is, XML or JSON). The use of semantics in this integration process is key to building an approach that scales to large numbers of heterogeneous sources.

We implemented our approach to using semantics for big data integration in a system called Karma. It allows a user to (1) import data from a wide variety of sources, (2) clean and normalize the data, (3) quickly build a model or semantic description of each source, and (4) integrate the data across sources using this model. Karma has been used on a variety of types of data, including biological data, mobile phone data, geospatial data, and cultural heritage data.[1]

In order to illustrate the approach to integrating data in Karma, we will use an example from the cultural heritage domain. Consider the task of building a virtual museum that integrates the metadata about artwork from tens, hundreds, or even thousands of museums. Museums have their own internal collection management systems with the data stored in a variety of formats and organized using different schemas. The challenge is how to quickly import, normalize, model, and integrate the data from many different museum data sources.

This article is organized by describing the end-to-end process of integrating the data across a set of heterogeneous sources. First, we import data from any of a variety of sources, including databases, spreadsheets, or web services that provide XML or JSON. Second, we clean and normalize the data, identifying any outliers and normalizing the data such that the same formats are used across all of the related sources. Third, we create a semantic description of each source and the system attempts to automate as much of this source modeling process as possible. Fourth, we integrate the data by converting all records into a uniform format using the semantic descriptions and then integrating the data within this unified framework. Finally, we conclude by describing other work on this topic and presenting a discussion of directions for future work.

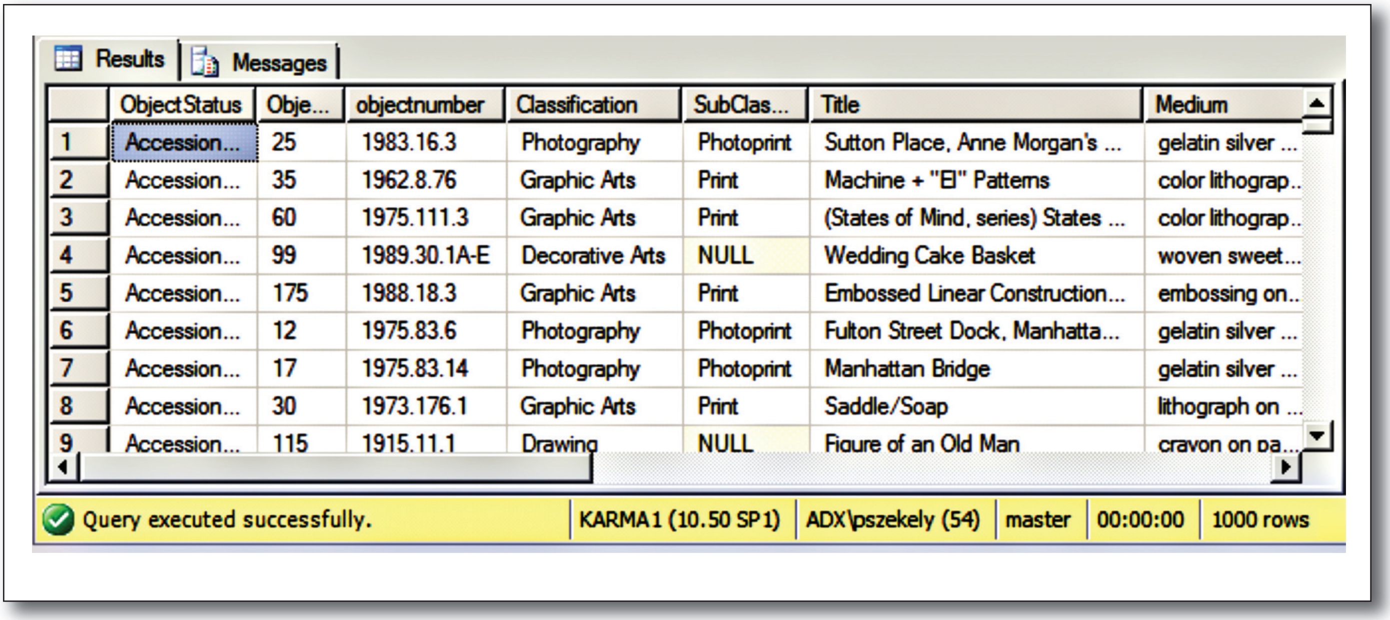
Importing

Data sets come in different formats. For example, in our museum use case, we received data in spreadsheets (figure 1), comma-separated values (CSV), JSON (figure 3), XML, and relational databases (figure 2).

One challenge in integrating diverse data sources is the ability to import different data formats into a common representation. A second challenge is that when the sources are large it is not possible to read an entire source into main memory.

Karma addresses the data format challenge by converting all data formats into a nested relational data model (Wang, Yang, and Han 2009). In this model, data is represented in tables where cells can contain scalar values (text strings, numbers, and others) or nested tables. When a column contains nested tables, these nested tables have identical structure, that is, they have the same columns. These nested tables can, and typically do, have different numbers of rows.

Figure 3 shows how Karma converts a JSON file into the nested relational model. Each attribute becomes a column. When an attribute contains arrays or objects, these become nested tables. When the value is a JSON array, the corresponding nested table has one row for each element of the array. Karma can import JSON data even if it is not complete-



	ObjectStatus	Obje...	objectnumber	Classification	SubClas...	Title	Medium
1	Accession...	25	1983.16.3	Photography	Photoprint	Sutton Place, Anne Morgan's ...	gelatin silver ...
2	Accession...	35	1962.8.76	Graphic Arts	Print	Machine + "El" Patterns	color lithograp...
3	Accession...	60	1975.111.3	Graphic Arts	Print	(States of Mind, series) States ...	color lithograp...
4	Accession...	99	1989.30.1A-E	Decorative Arts	NULL	Wedding Cake Basket	woven sweet...
5	Accession...	175	1988.18.3	Graphic Arts	Print	Embossed Linear Construction...	embossing on...
6	Accession...	12	1975.83.6	Photography	Photoprint	Fulton Street Dock, Manhatta...	gelatin silver ...
7	Accession...	17	1975.83.14	Photography	Photoprint	Manhattan Bridge	gelatin silver ...
8	Accession...	30	1973.176.1	Graphic Arts	Print	Saddle/Soap	lithograph on ...
9	Accession...	115	1915.11.1	Drawing	NULL	Figure of an Old Man	crayon on pa...

Query executed successfully. | KARMA1 (10.50 SP1) | ADX\pszekely (54) | master | 00:00:00 | 1000 rows

Figure 2. Smithsonian American Art Museum Data in a Relational Database.

ly regular: objects in an array can be heterogeneous, and can contain attributes that are missing from other objects. Karma creates a column for every attribute it finds, and if an object is missing this attribute the corresponding cell in the nested table will be empty. Highly heterogeneous collections of objects can lead to sparsely populated tables with many columns. Karma imports XML documents similarly, creating nested tables for XML elements and creating separate columns for each attribute.

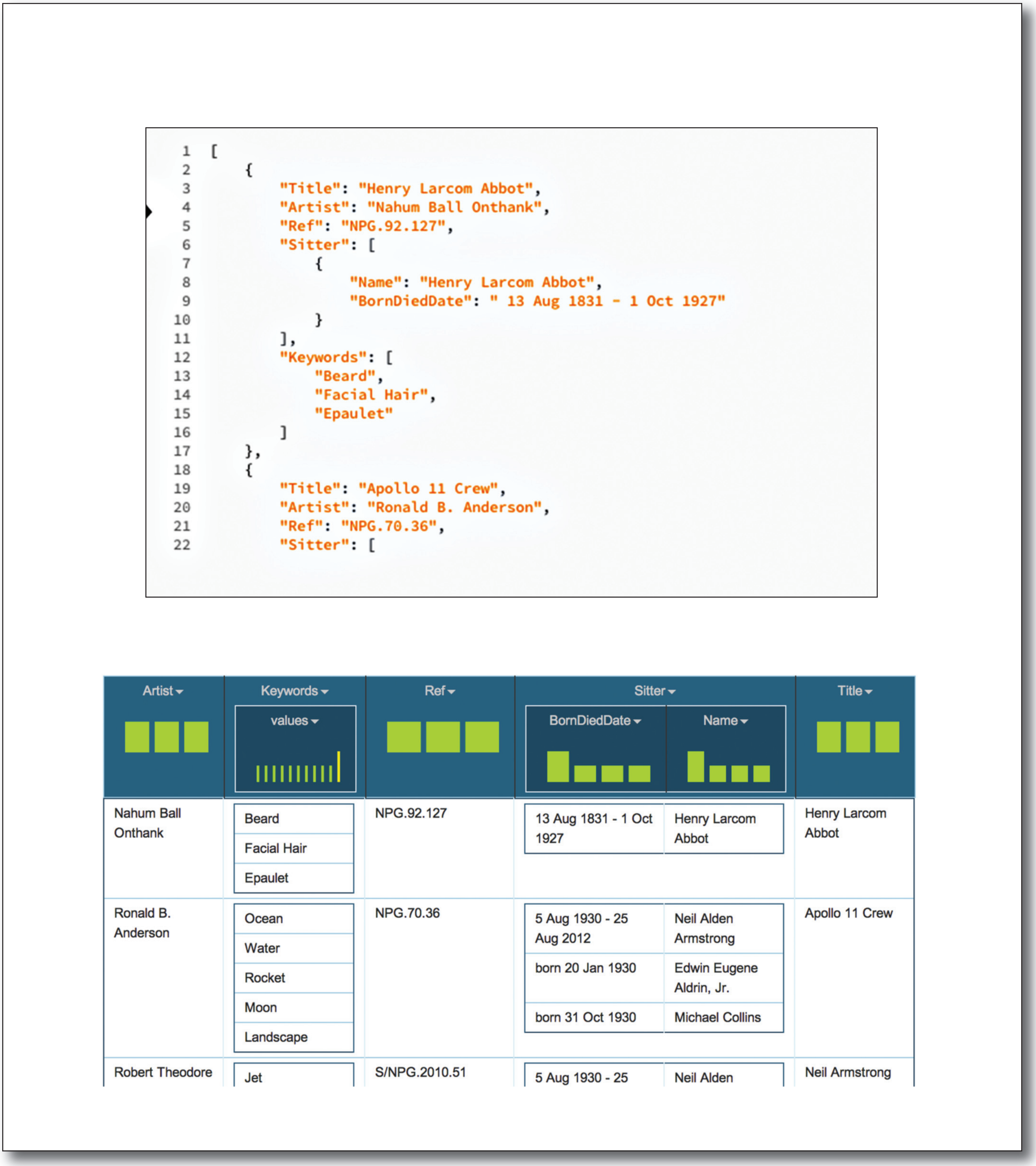
Karma addresses the data size challenge by only importing subsets of large data sets. In interactive mode, when users use Karma to clean and model the data, Karma imports a sample of the data so that users can work interactively. Once users define a model, they can store it on disk and apply it to large data sources in batch mode. When Karma operates in batch mode, it repeatedly loads subsets of the data into memory, converts each subset to its nested relational representation, applies the operations defined in the model to the loaded subset, and outputs the corresponding results.

Cleaning

Several data cleaning problems arise when integrating data across sources. First, there are often noisy data, missing values, and inconsistencies that need to be identified and fixed. Second, the data in different sources is often represented in different and incompatible ways. Figure 4 illustrates both types of problems with the data from the Crystal Bridges Museum. The spreadsheet has missing values and inconsistent representations for years (for example, “ca. 1902,” “1936-1937”). Also artist names in the first column are represented as “last-name, first-name,” while other data sets represent names as “first-name last-name.” In order to integrate these data we need to resolve the inconsistencies.

Karma helps the user find noisy, missing, or inconsistent data by performing an analysis of the data distribution in each column. This is shown in figure 4 with a bar chart at the top of each column. Each bar represents the frequency of a particular value for up to 10 values. If there are more than 10 different values, the remaining values are shown in a yellow bar at the right of the bar chart. A white bar shows the frequency of null values in the column; a red bar shows the frequency of outliers (values whose type is different from that of the majority of values). For example, one can see that the column for “End Date” has a number of null values given the large white bar. And one can see the column for “Dated” has a number of outliers (values that cannot be parsed as dates) given the large red bar. This analysis helps users to quickly identify potential problems in a new data set.

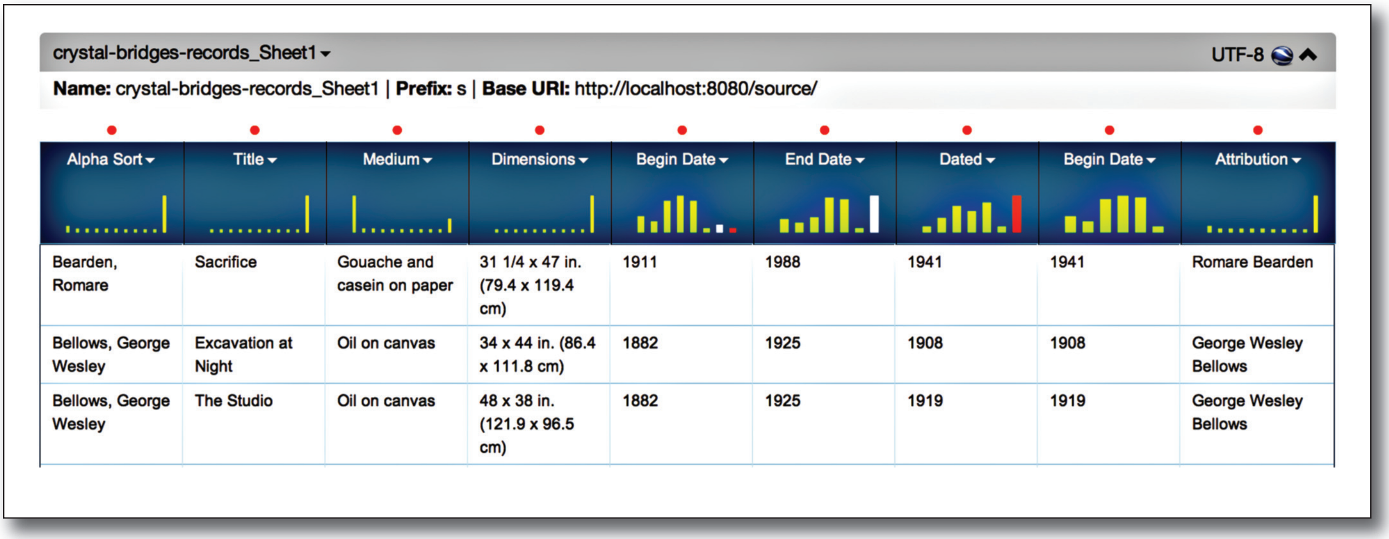
In order to address the problem of data that is in the wrong format, we developed an approach that allows a user to quickly transform data by example (Wu, Szekely, and Knoblock 2014). In our approach, the user selects a column to transform and then just provides examples of the transformed data for individual rows. The system learns a transformation program from the examples, applies it to the rest of the data, and shows the results to the user.



```
[
    {
        "Title": "Henry Larcom Abbot",
        "Artist": "Nahum Ball Onthank",
        "Ref": "NPG.92.127",
        "Sitter": [
            {
                "Name": "Henry Larcom Abbot",
                "BornDiedDate": " 13 Aug 1831 - 1 Oct 1927"
            }
        ],
        "Keywords": [
            "Beard",
            "Facial Hair",
            "Epaulet"
        ]
    },
    {
        "Title": "Apollo 11 Crew",
        "Artist": "Ronald B. Anderson",
        "Ref": "NPG.70.36",
        "Sitter": [
```

Artist	Keywords	Ref	Sitter		Title
	values		BornDiedDate	Name	
Nahum Ball Onthank	Beard Facial Hair Epaulet	NPG.92.127	13 Aug 1831 - 1 Oct 1927	Henry Larcom Abbot	Henry Larcom Abbot
Ronald B. Anderson	Ocean Water Rocket Moon Landscape	NPG.70.36	5 Aug 1930 - 25 Aug 2012 born 20 Jan 1930 born 31 Oct 1930	Neil Alden Armstrong Edwin Eugene Aldrin, Jr. Michael Collins	Apollo 11 Crew
Robert Theodore	Jet	S/NPG.2010.51	5 Aug 1930 - 25	Neil Alden	Neil Armstrong

Figure 3. Raw JSON Data (top) and Data Imported in Karma (bottom) for the National Portrait Gallery.



crystal-bridges-records_Sheet1 UTF-8

Name: crystal-bridges-records_Sheet1 | **Prefix:** s | **Base URI:** http://localhost:8080/source/

Alpha Sort	Title	Medium	Dimensions	Begin Date	End Date	Dated	Begin Date	Attribution
Bearden, Romare	Sacrifice	Gouache and casein on paper	31 1/4 x 47 in. (79.4 x 119.4 cm)	1911	1988	1941	1941	Romare Bearden
Bellows, George Wesley	Excavation at Night	Oil on canvas	34 x 44 in. (86.4 x 111.8 cm)	1882	1925	1908	1908	George Wesley Bellows
Bellows, George Wesley	The Studio	Oil on canvas	48 x 38 in. (121.9 x 96.5 cm)	1882	1925	1919	1919	George Wesley Bellows

Figure 4. The Crystal Bridges Data Once It Has Been Loaded Into Karma and Analyzed for Data Cleaning.

Figure 5 shows an example of this learning process. In the example, the user entered two examples with the names in the correct order and the system learned a transformation and applied it to the rest of the data. The system uses active learning with uncertainty sampling to suggest a new example for labeling, selecting the example that it is least confident about and thereby helping the user to quickly find the examples that might be transformed incorrectly. In this case, it selects an artist that has a single name, so the transformation of moving the first name first does not apply, which is correct. Karma shows the transformation on each row using color coding so that the user can quickly visualize the effects of the transformation on the data. The system also highlights several other examples on which the learned transformation does not apply. The user can then decide if these are correct or provide additional examples. If the user provides additional examples, the system would then relearn a new transformation program that covers the examples.

This approach to transforming the data allows a user to define the required transformations by simply working with the data and showing the system the desired result. By user active learning and identifying the examples on which the system is least confident, the system can quickly converge on a transformation program that works on the entire data set and the user does not need to scan all of the data. This approach to learning transformations does not require any programming knowledge. Karma also provides a script-based transformation capability similar to the Open Refine Expression Language,[2] which enables users to write their transformations in Python.

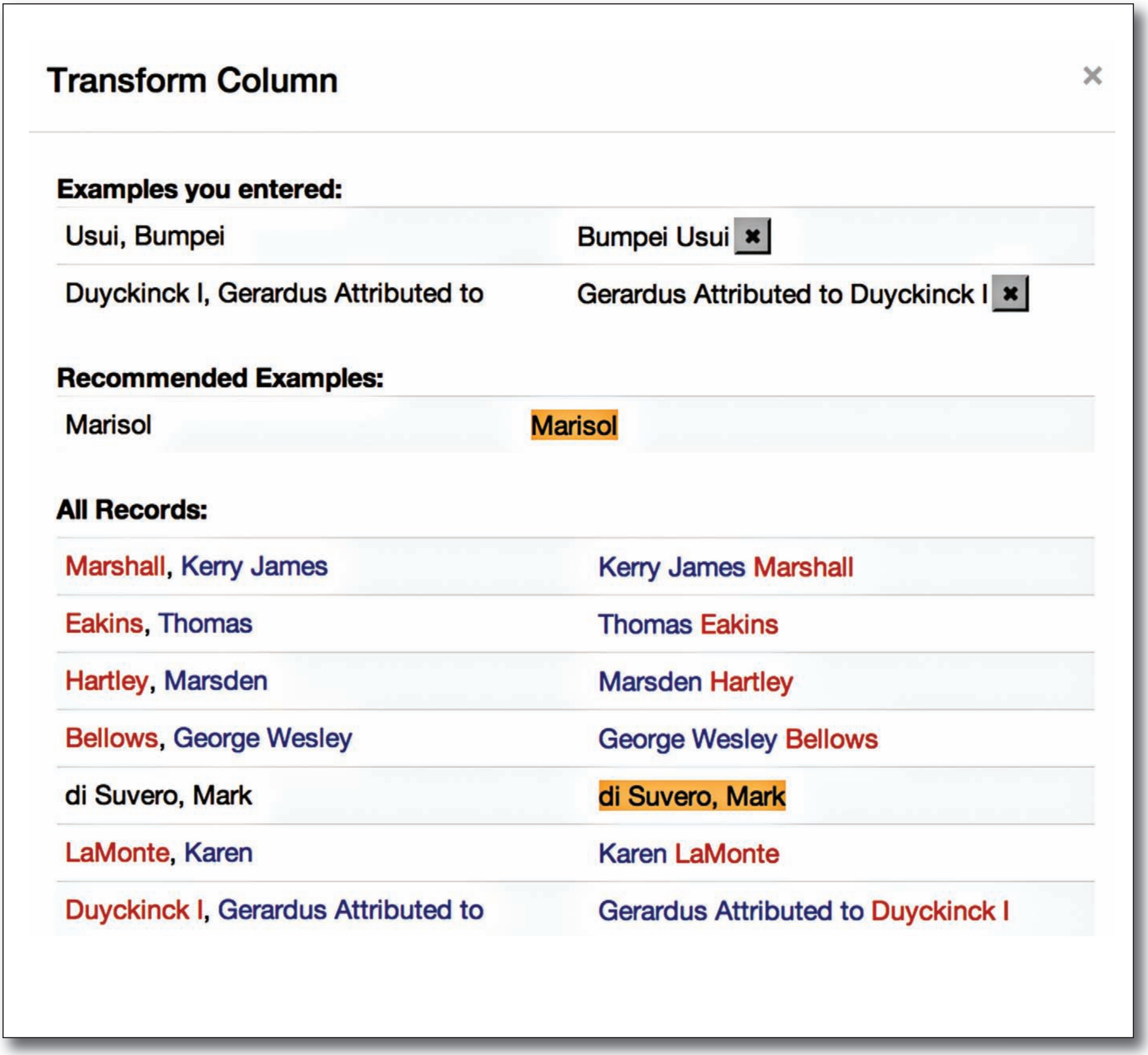
Modeling

One of the main challenges of integrating diverse data sets is to harmonize their representation. Even after we clean the data to normalize the values of individual attributes, we are left with two main problems.

Nomenclature differences. Data sets from different providers often use different names to refer to attributes that have the same meaning. For example, the Crystal Bridges spreadsheet has the artist names under an “AlphaSort” column, the National Portrait Gallery has the artist names in the “Artist” attribute, and the Smithsonian database uses “ConsitutentId” to identify artists and has the names in a separate table. We need to unify the different nomenclatures that have a common meaning.

Format and structure differences. As illustrated in the museum use-case, different data sets come in different formats, relational, text-delimited, JSON, XML, and others. Furthermore, even the values of individual attributes are represented in different structures. For example, the “keywords” attribute in the Crystal Bridges spreadsheet consists of a comma-separated list of values; in the National Portrait Gallery JSON representation, the keywords come in a nested JSON object; and in the Smithsonian data set, the keywords are represented in separate rows in a “keywords” table. We need to convert all these into a common representation.

In Karma, we address these differences by modeling all data sets with respect to a common ontology. Modeling is the process of specifying how the different fields of a data set (columns in a database or spreadsheet, attributes in a JSON object, elements in



Transform Column

Examples you entered:

Usui, Bumpei	Bumpei Usui ✖
Duyckinck I, Gerardus Attributed to	Gerardus Attributed to Duyckinck I ✖

Recommended Examples:

Marisol	Marisol

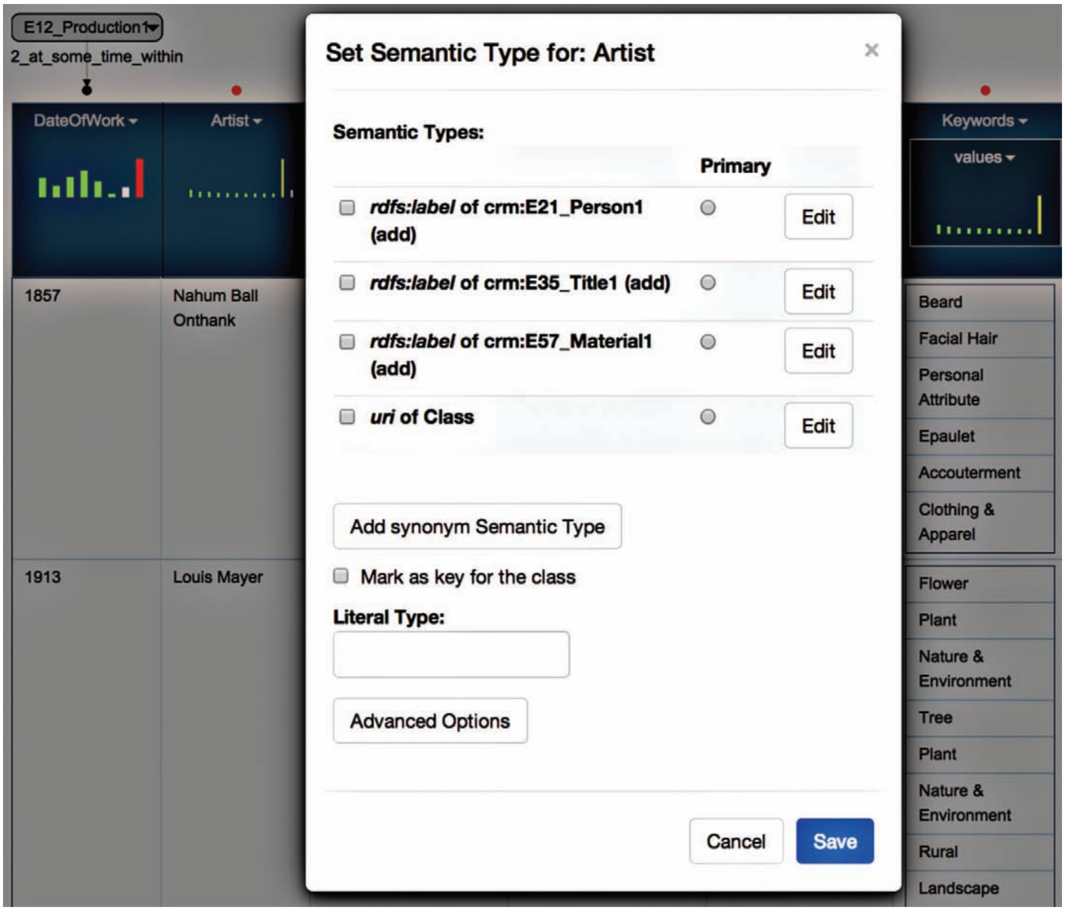
All Records:

Marshall, Kerry James	Kerry James Marshall
Eakins, Thomas	Thomas Eakins
Hartley, Marsden	Marsden Hartley
Bellows, George Wesley	George Wesley Bellows
di Suvero, Mark	di Suvero, Mark
LaMonte, Karen	Karen LaMonte
Duyckinck I, Gerardus Attributed to	Gerardus Attributed to Duyckinck I

Figure 5. Transformation Window Showing the Examples Entered, the Next Example to Consider, and the Results on the Remainder of the Data.

an XML document) map to classes and properties in an ontology. In addition, it is necessary to specify how the different fields are related. For example, it is not enough to specify that a field contains the "name" of "Person," where "name" is a property and "Person" is a class in an ontology. This person could be the creator of an artwork or a person depicted in a painting, so it is necessary to also specify how this person is related to the "Artwork" class.

Modeling can be conceptually and technically difficult. For example, in the museum use case, we chose to model the data using the CIDOC CRM ontology,[3] an ISO standard ontology for modeling cultural heritage data, and the SKOS ontology for modeling knowledge organization systems.[4] These ontologies combined have 93 classes and 305 properties, so it is conceptually challenging to select the appropriate classes and properties to map the fields in a data set



Set Semantic Type for: Artist

Semantic Types:

	Primary	
□ *rdfs:label* of crm:E21_Person1 (add)	○	Edit
□ *rdfs:label* of crm:E35_Title1 (add)	○	Edit
□ *rdfs:label* of crm:E57_Material1 (add)	○	Edit
□ *uri* of Class	○	Edit

Figure 6. Karma Suggests Semantic Types for the Artist Column After the User Clicks the Red Dot.

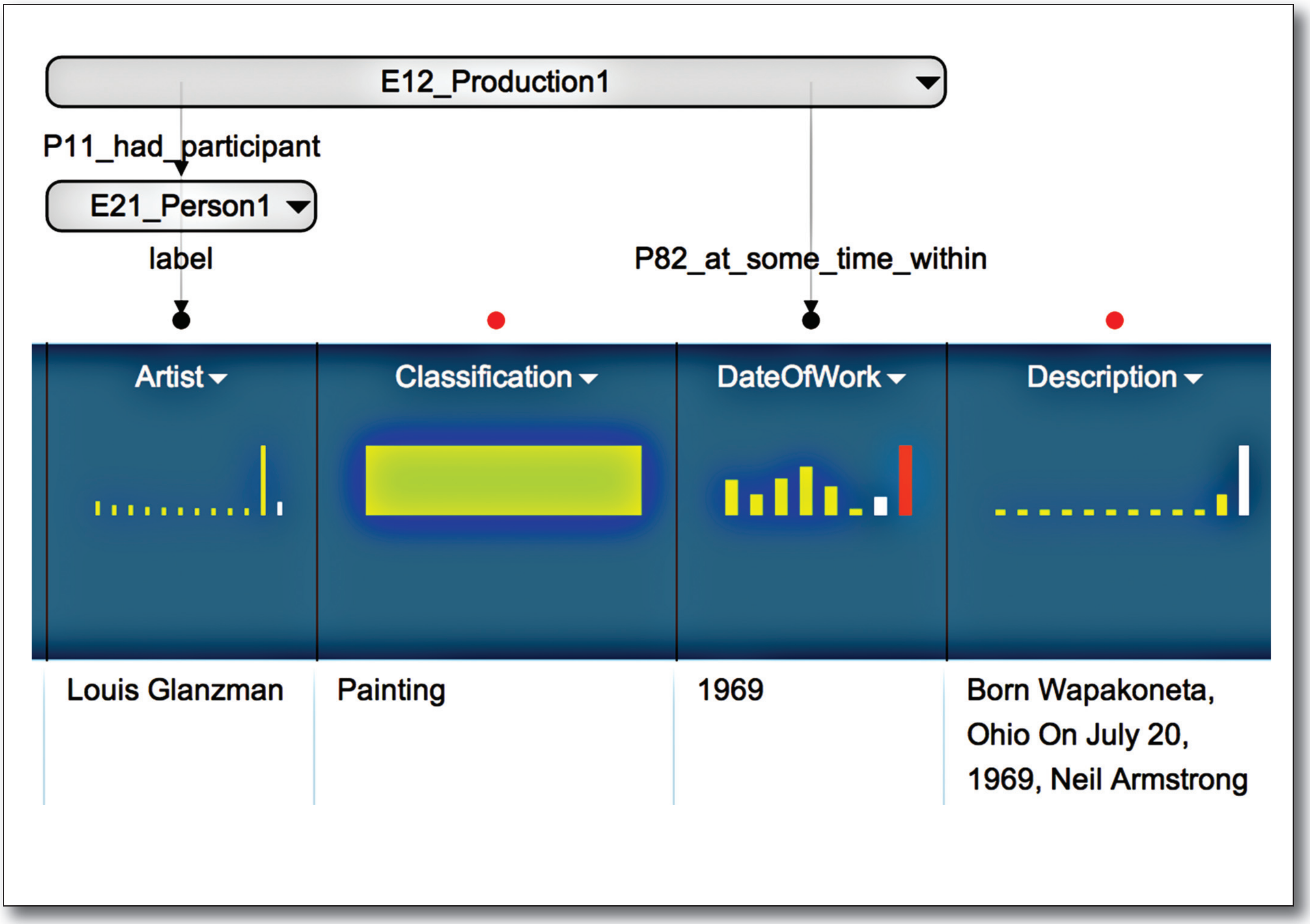
and to specify their relationships. It is also technically challenging to specify the mappings because different representations use different technologies to refer to fields (that is, SQL queries, Xpath, XSLT, or JSONPath) and then one needs to specify how the values in these fields map to classes and properties in an ontology.

The goal in Karma is to enable data-savvy users (for example, spreadsheet users) to model sources, shielding them from the complexities of the underlying technologies (for example, SQL, SPARQL, graph patterns, XSLT, XPath, and so on). Karma addresses this goal by automating significant parts of the process by providing a visual interface where users see the Karma-proposed mappings and can adjust them if necessary, and by enabling users to work with example data rather than schemas and ontologies. Karma is ontology agnostic, so users can choose the ontologies that are most appropriate for their data sets.

The Karma approach to map data to ontologies involves two interleaved steps: one, assignment of semantic types to data columns, and two, specification of the relationships between the semantic types.

A semantic type can be either a class or a pair consisting of a data property and the domain of the data property. For example, if a field contains the URIs for people, then its semantic type can be the class Person (called E21_Person in the CRM ontology). Figure 6 shows several examples of semantic types that Karma suggests for the "Artist" field in the National Portrait Gallery data set. The first suggestion for the "Artist" field is the semantic type defined by the "rdfs:label" property and the "crm: E21_Person" class. This suggestion is the correct one for this field.

Karma uses a conditional random field (CRF) model to learn the assignment of semantic types to data fields based on prior user-provided assignments (Goel, Knoblock, and Lerman 2012). The CRF uses a



Artist	Classification	DateOfWork	Description
Louis Glanzman	Painting	1969	Born Wapakoneta, Ohio On July 20, 1969, Neil Armstrong

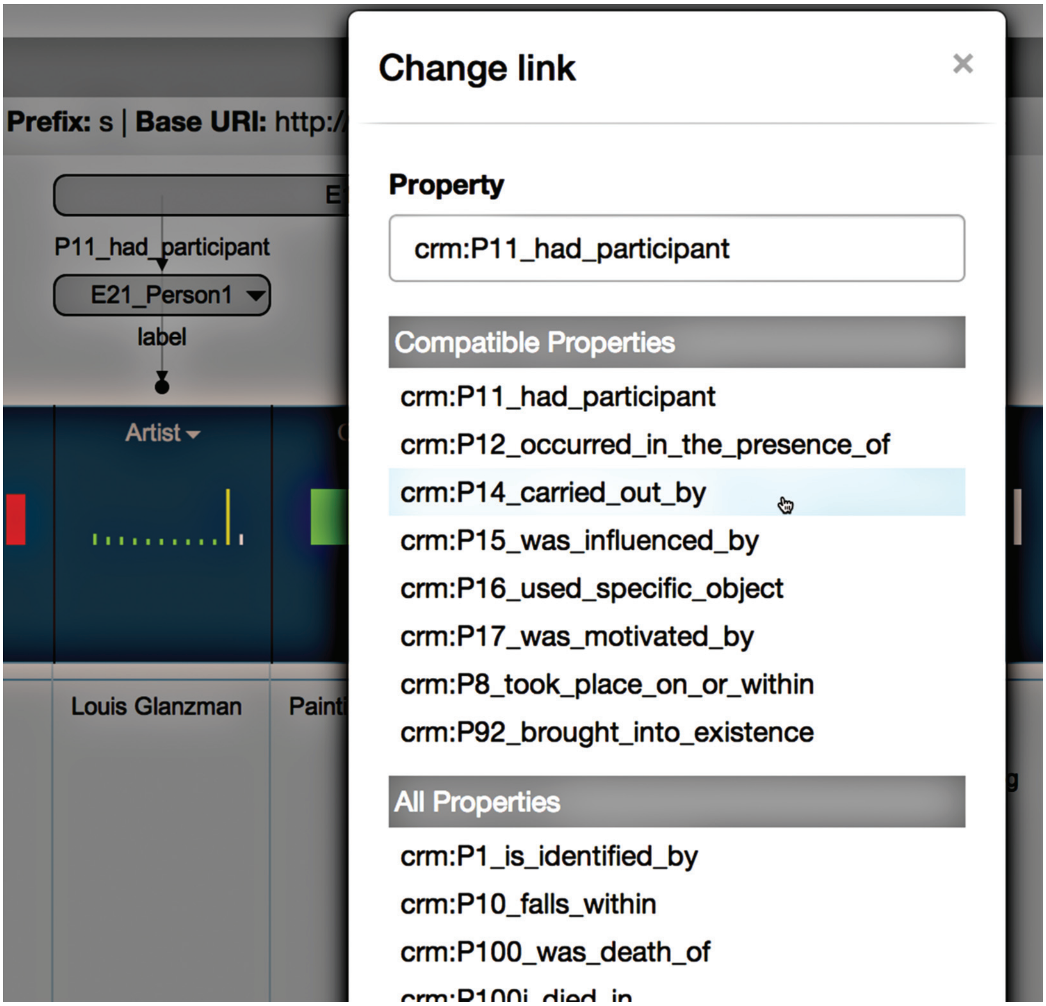
Figure 7. Karma Automatically Suggests the Had_Participant Relationship Between Production and Person.

wide variety of features of the individual tokens, including both constant values and properties of the data, in a field to learn a model that can discriminate each class from the other classes in a domain. Karma uses the CRF model to automatically suggest semantic types for unassigned data fields. When the desired semantic type is not among the suggested types, users can edit any of the suggestions. Karma provides convenient autocompletion type-in fields to quickly enter the properties and classes, and also provides an ontology browser to find the appropriate property and class. Karma automatically retrains the CRF model after these manual assignments.

Figure 7 shows a screenshot of the state of the model after the user accepts the suggested semantic type for the "Artist" column (the user had previously specified the semantic type for the "DateOfWork" field). The screen shows the "E21_Person/label" semantic type that the user just selected, and in addition proposes the relationship "P11_had_participant" between the previously defined semantic type and the new one.

The relationships between semantic types are specified using paths of object properties. Given the ontologies and the assigned semantic types, Karma creates a graph that defines the space of all possible mappings between the data source and the ontologies. The nodes in this graph represent classes in the ontology, and the edges represent properties. Karma then computes the minimal tree that connects all the semantic types (Taheriyan et al. 2012). This tree corresponds to the most concise model that relates all the columns in a data source, and it is a good starting point for refining the model. Sometimes, multiple minimal trees exist, or the correct interpretation of the data is defined by a nonminimal tree. For these cases, Karma provides an easy-to-use GUI (figure 8) to let users select the edge in the graph that specifies the desired relationship. The menu contains only those properties whose domains and ranges are compatible with the classes to be connected. Karma then computes a new minimal tree that incorporates the user-specified relationships.

Karma improves the accuracy of its automatic



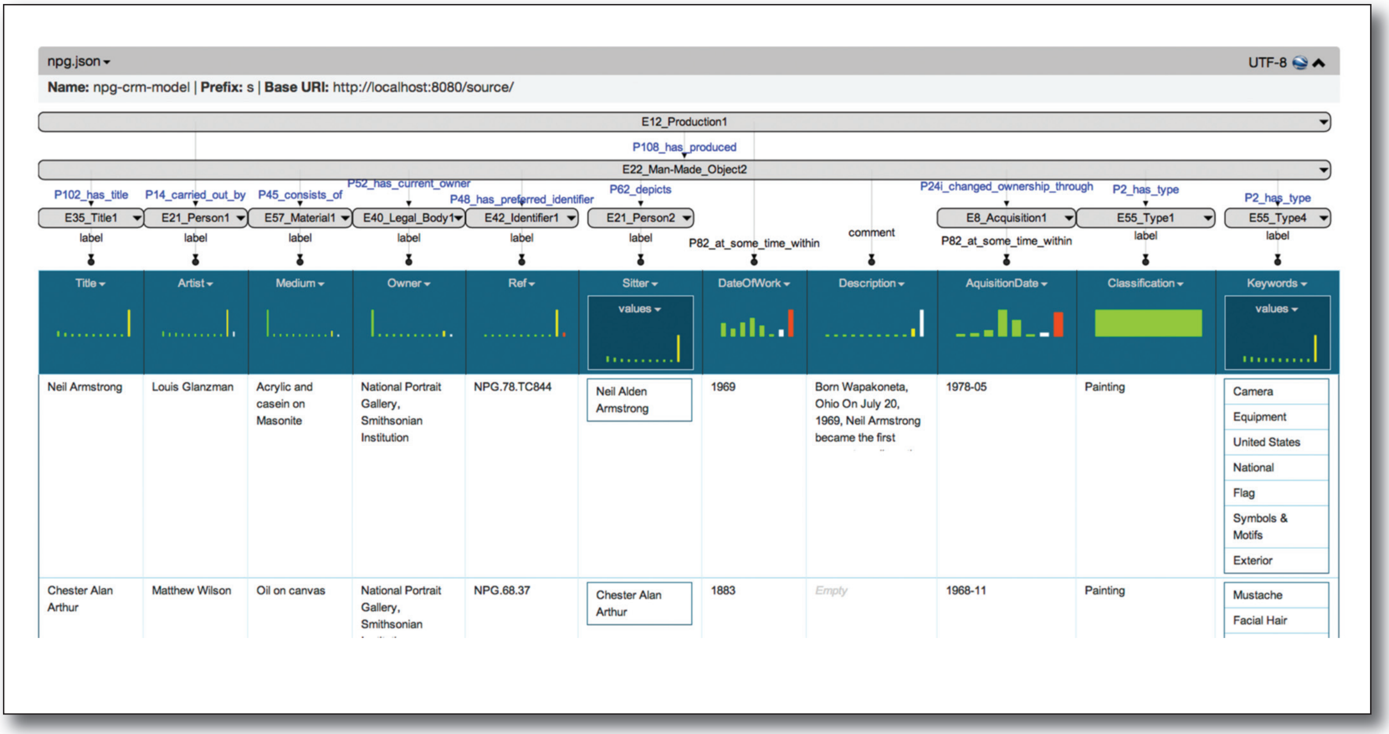
Property: crm:P11_had_participant

Figure 8. The User Clicked on Had_Participant to View a Menu Showing All Properties That Can Connect Production to Person.

modeling by learning from previously defined models for other sources in the same domain (Taheriyan et al. 2013). The system does this by learning coherent substructures of models that include both semantic types and the relationships among them. For example, in the museum domain, when a data set contains information about two people (two people columns) it will propose a model where one person is the creator and the other one is the sitter. It does this because the creator/sitter situation is more common than multiple creators.

Karma's model learning greatly reduces the effort needed to create new models and makes it practical to create models for a large number of sources. In one evaluation (Knoblock et al. 2012) we used Karma to replicate the models for 10 sources that had been specified using 41 rules in a rule-based language. Using Karma it was possible to build the same models in a short amount of time and required on average 0.75 user actions per rule. The user actions are specification of semantic types (figure 6) or selection of links (figure 8). In a more recent evaluation (Taheriyan et al. 2013), Karma's model-learning component can further reduce the effort by 58 percent in a geospatial domain and 30 percent in the museum domain.

Figure 9 shows the model of the National Portrait



Name: npg-crm-model | **Prefix:** s | **Base URI:** http://localhost:8080/source/

Title	Artist	Medium	Owner	Ref	Sitter	DateOfWork	Description	AquisitionDate	Classification	Keywords
Neil Armstrong	Louis Glanzman	Acrylic and casein on Masonite	National Portrait Gallery, Smithsonian Institution	NPG.78.TC844	Neil Alden Armstrong	1969	Born Wapakoneta, Ohio On July 20, 1969, Neil Armstrong became the first	1978-05	Painting	Camera Equipment United States National Flag Symbols & Motifs Exterior
Chester Alan Arthur	Matthew Wilson	Oil on canvas	National Portrait Gallery, Smithsonian	NPG.68.37	Chester Alan Arthur	1883	Empty	1968-11	Painting	Mustache Facial Hair

Figure 9. The Model of the National Portrait Gallery Data Set Captures the Meaning of all Fields and Their Relationships.

Gallery data set using the CRM ontology. Artworks are modeled as "Man-Made-Objects," artists are "Persons" ("Person1"), who carried out "Production" events that resulted in the creation of "Man-Made_Objects" at specific times. The model represents the "Legal_Body" that owns the artwork and the "Acquisition" events that transferred custody. The model also describes a second "Person" ("Person2"), who is the person depicted in the artwork.

Using Karma a user can build rich semantic models like this one in a few hours, including the time to (1) reflect on the model to select the most appropriate classes and properties to represent the meaning of the underlying data, (2) normalize, combine, and break fields as necessary, and (3) make adjustments when Karma's suggestions are incorrect. Similar effort is needed to build models for the data sets of the other museums, regardless of the format, structure, and nomenclature differences.

These rich semantic models are useful for integration because they accurately capture the information that data sets have in common (for example, most museum data sets will be about "Man-Made_Objects"), and also accurately capture the differences (for example, the National Portrait Gallery data set has information about "People" "depicted" in the artworks, while the Crystal Bridges data set does not).

Integrating Data

Information integration involves two main tasks. The first, integration at the schema level, involves homogenizing differences in the schemas and nomenclature used to represent the data. The second, integration at the data level, involves identifying records in different data sets that refer to the same real-world entity. Karma focuses on the schema-level integration problem, and in recent work (Knoblock et al. 2013) we have developed an initial approach that combines schema- and data-level integration approaches in a single unified approach.

Once users model their data sets using a common domain ontology, Karma can use these models to convert the data into the schema defined by the domain ontology. For example, recall that the data from the National Portrait Gallery is represented in JSON, and the data from Crystal Bridges is represented in an Excel spreadsheet, using different names for attributes and column headers, and using different formats for values. Once the user models them using the CRM ontology, Karma can convert the data into RDF using a common set of terms. Table 1 shows sample records from the National Portrait Gallery and Crystal Bridges. They are now represented in a common schema. The two data sets have information about the title of the artwork, ownership, material, and author. The National Portrait Gallery has information about what the artworks depict, while the

National Portrait Gallery	Crystal Bridges
<http://npg.org/ob/NPG_70_36> a crm:E22_Man-Made_Object ; crm:P102_has_title [a crm:E35_Title ; rdfs:label "Apollo 11 Crew"] ;	<http://cb.org/ob/3> a crm:E22_Man-Made_Object ; crm:P102_has_title [a crm:E35_Title ; rdfs:label "Excavation At Night"] ;
crm:P24i_changed_ownership_through [a crm:E8_Acquisition crm:P29_custody_received_by npg:NationalPortraitGallery ; crm:P82_at_some_time_within 1970] ;	crm:P24i_changed_ownership_through [a crm:E8_Acquisition ; crm:P29_custody_received_by cb:Crystal_Bridges] ;
crm:P45_consists_of npg:Oilonboard ;	crm:P45_consists_of cb:Oiloncanvas ;
crm:P50_has_current_keeper npg:NationalPortraitGallery ;	crm:P50_has_current_keeper cb:Crystal_Bridges ;
crm:P62_depicts npg:EdwinEugeneAldrin_Jr, npg:MichaelCollins, npg:NeilAldenArmstrong ; crm:P2_has_type npg:Flag, npg:Moon, npg:Rocket .	
	crm:P43_has_dimension [a crm:E54_Dimension ; crm:P2_has_type <http://aac.org/dimension/height> ; crm:P91_has_unit qudt:Centimeter ; crm:P90_has_value 111.8] ; crm:P43_has_dimension [...] .
[ ] a crm:E12_Production ; crm:P108_has_produced <http://npg.org/ob/NPG_70_36> ; crm:P14_carried_out_by <http://npg.org/id/RonaldB.Anderson> ; crm:P82_at_some_time_within 1969 .	[ ] a crm:E12_Production ; crm:P108_has_produced <http://cb.org/ob/3> ; crm:P14_carried_out_by <http://cb.org/id/RonaldBAnderson> ; crm:P82_at_some_time_within 1882 .

Table 1. Sample Records from the National Portrait Gallery and Crystal Bridges.

Expressed in the RDF Turtle format and using the common domain ontology chosen for representing the museum data sets.

Crystal Bridges data set has information about the dimensions of the work.

Once the data is converted into RDF using the same ontology, it can be loaded into a triple store and easily queried using SPARQL, the RDF query language. For example, the following simple query retrieves all artworks in the database that depict Neil Armstrong.

```
Select ?artwork { ?artwork a crm:E22_Man-Made_Object ;
    crm:P62_depicts nyt:NeilArmstrong }
```

In addition to the common use case of converting heterogeneous data sets into RDF in their entirety, the Karma models support other kinds of integration capabilities.

Converting data into other formats. Similarly to how Karma uses the models to convert data to RDF, Karma can also use the models to convert data to CSV format or to load it into a relational database. Users specify the columns they want in the output in terms of the semantic types of the model. The output tables have as headers the properties that define the semantic types. In an analogous way, the models could be used to convert the data to JSON or XML, although this is not currently implemented.

Converting data on the fly. A common use case is to define models based on samples of data and then to use the models to convert the results of queries sent to multiple data sources. In this use case, data stays in the native format (for example, relational, JSON, and others). Queries, typically keyword-based queries, are sent to multiple data sources, and a run-time Karma library applies the models to the query results, converting them into the common format defined by the ontology. The advantage is that the converted data is the up-to-date data from the database.

Exporting models in R2RML. R2RML is a W3C standard for defining mappings from relational databases to RDF. R2RML uses RDF syntax to specify the mappings, so manually defining mappings is cumbersome and error prone. Karma can export models for relational databases in the R2RML standard, enabling users to employ Karma's easy-to-use graphical user interface to define R2RML mappings.

Related Work

A closely related body of work is the research in schema mapping. This research ranges from seminal work on Clio (Fagin et al. 2009), which provided a practical system and furthered the theoretical foundations of data exchange (Fagin et al. 2005) to more recent systems that support additional schema constraints (Marnette et al. 2011). Alexe et al. (2011) generate schema mappings from examples of source data tuples and the corresponding tuples over the target schema.

Our work in this article is complementary to these schema-mapping techniques. Instead of focusing on satisfying schema constraints, we analyze known source descriptions to propose mappings that capture more closely the semantics of the target source in ways that schema constraints could not disambiguate, for example, suggesting that a worksFor relationship is more likely than CEO in a given domain. Moreover, following Karma, our algorithm can incrementally refine the mappings based on user feedback and improve future predictions.

The problem of describing semantics of data sources is at the core of data integration (Doan, Halevy, and Ives 2012) and exchange (Arenas et al. 2010). The main approach to reconcile the semantic heterogeneity among sources is to define logical mappings between the source schemas and a common target schema. Formally, these mappings are known as GLAV rules or source-to-target tuple-generating dependencies (st-tgds). The R2RML W3C recommendation (Das, Sundara, and Cyganiak 2012) captures this practice for semantic web applications. Although these mappings are declarative, defining them requires significant technical expertise, so there has been much interest in techniques that facilitate their generation. Karma can directly generate R2RML from the semiautomatically produced source model.

Carman and Knoblock (2007) also use known source descriptions to generate a GLAV mapping for an unknown target source. However, a limitation of that work is that the approach can only learn descriptions consisting of conjunctive combinations of known source descriptions. By exploring paths in the domain ontology, in addition to patterns in the known sources, we can hypothesize target mappings that are more general than previous source descriptions or their combinations.

There are a number of systems that support data transformation and cleaning. OpenRefine[5] and Potter's Wheel (Raman and Hellerstein 2001) allow the user to specify edit operations. OpenRefine is a tool for cleaning messy data. Its language supports regular expression style of string transformation and data layout transformation. Potter's Wheel defines a set of transformation operations and let users gradually build transformations by adding or undoing transformations in an interactive GUI. Lau et al. (2003) developed a system that can learn from a user's edit operations and generate a sequence of text editing programs using the version space algebra. Data Wrangler (Kandel et al. 2011) is an interactive tool for creating data transformation that uses the transformation operations defined in Potter's wheel. Besides supporting string level transformation, it also supports data layout transformation including column split, column merge, and fold and unfold. Our approach is different from these systems as it only requires users to enter the target data. In addition, these systems support structured and semistructured data, but lack support for hierarchical data (for example, XML, JSON, RDF) and they are focused on single data sets, lacking support for integrating multiple data sets. There is also an extension to Google Refine that makes it possible to publish data in RDF with respect to a domain model (Maali, Cyganiak, and Peristeras 2012), but the mapping to the domain model is defined manually.

Gulwani (2011) developed an approach to program synthesis through input and output string pairs. This method needs to generate multiple programs and evaluate these programs on all the records, which can require more processing time. Our approach improves Gulwani's work by providing recommendations. To generate a recommendation, we only need one program and its results on the records.

Discussion

In this article we described how we address the problem of big data variety in Karma. The key idea to support the integration of large numbers of heterogeneous sources is to use a domain ontology to describe each source and to automate the modeling of the individual sources. In some cases, a user may still need to refine the automatically generated models, but our goal is to simplify this process as much as possible with good visualization tools and an easy-to-use interface to refine the models. The other important contributions of this work are (1) the ability to import diverse sources of data including both relational and hierarchical sources, (2) the approach to learning transformations by example to quickly prepare data for modeling and integration, and (3) the capability to easily integrate the data across sources and publish the results in a variety of formats. In the context of large sources, all of these steps can be defined on a sample of the data sources and then executed in batch mode on very large sources.

Several important problems that Karma does not yet solve are areas of current research. First, this article focused on integrating data sources at the schema level, but there is an equally important problem of linking the data at the record level. For example, if there are artists in two museum data sets, one would like to know which records refer to the same artist, and this is not simply a matter of matching the names. The fact that we have already linked the sources at the schema level means that we know which fields should be compared across sources. We are working to integrate record linkage algorithms directly into Karma and building visualization tools that allow a user to see the results of the linking process and curate the results. When we complete this new capability, Karma will provide an end-to-end system to efficiently integrate and link large data sets, which will support the large-scale construction of high-quality linked data (Bizer, Heath, and Berners-Lee 2009).

Second, this article focused on the issue of variety and did not address the issues of volume and velocity, which are the other key dimensions of big data. With respect to volume, Karma can already create integration plans on a sample of the data and then execute those plans on large data sets. We have used Karma on databases with several millions of rows, generating RDF triples at a rate of over one million triples per second. But the next step is to be able to support cloud-based execution of the data. We are currently working on building a cloud-based execution system for Karma. We will continue to define the data processing plans as they are defined today, but then these plans will be executed on a cloud platform. In terms of velocity, we have also begun work on how to support streaming sources, which in Karma will be modeled like any other source, but the execution system will need to be extended to support the streaming data.

Acknowledgements

This research is based upon work supported in part by the National Science Foundation under Grant No. 1117913 and in part by the Smithsonian American Art Museum. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, the Smithsonian, or the U.S. government.

Notes

[1] Karma is available as open source under a free commercial license: www.isi.edu/integration/karma.

[2] openrefine.org.

[3] www.cidoc-crm.org.

[4] www.w3.org/2004/02/skos/.

[5] openrefine.org.

References

Alexe, B.; ten Cate, B.; Kolaitis, P. G.; and Tan, W. C. 2011. Designing and Refining Schema Mappings via Data Examples. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data,* 133–144. New York: Association for Computing Machinery. dx.doi.org/10.1145/1989323.1989338

Arenas, M.; Barcelo, P.; Libkin, L.; and Murlak, F. 2010. *Relational and XML Data Exchange*. San Rafael, CA: Morgan and Claypool.

Bizer, C.; Heath, T.; and Berners-Lee, T. 2009. Linked Data — The Story So Far. *International Journal on Semantic Web and Information Systems* (IJSWIS) 5(3): 1–22. dx.doi.org/10.4018/jswis.2009081901

Carman, M. J., and Knoblock, C. A. 2007. Learning Semantic Definitions of Online Information Sources. *Journal of Artificial Intelligence Research* 30(1) 1–50.

Das, S.; Sundara, S.; and Cyganiak, R. 2012. R2RML: RDB to RDF Mapping Language, W3C Recommendation 27 September 2012. Cambridge, MA: World Wide Web Consortium (W3C) (www.w3.org/TR/r2rml).

Doan, A.; Halevy, A.; and Ives, Z. 2012. *Principles of Data Integration*. San Francisco: Morgan Kaufman.

Fagin, R.; Haas, L. M.; Hernandez, M. A.; Miller, R. J.; Popa, L.; and Velegrakis, Y. 2009. Clio: Schema Mapping Creation and Data Exchange. In *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos,* 198–236. Berlin: Springer.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data Exchange: Semantics and Query Answering. *Theoretical Computer Science* 336(1): 89–124. dx.doi.org/10.1016/j.tcs.2004.10.033

Goel, A.; Knoblock, C. A.; and Lerman, K. 2012. Exploiting Structure Within Data for Accurate Labeling Using Conditional Random Fields. In *Proceedings of the 14th International Conference on Artificial Intelligence*. Athens, GA: CSREA Press.

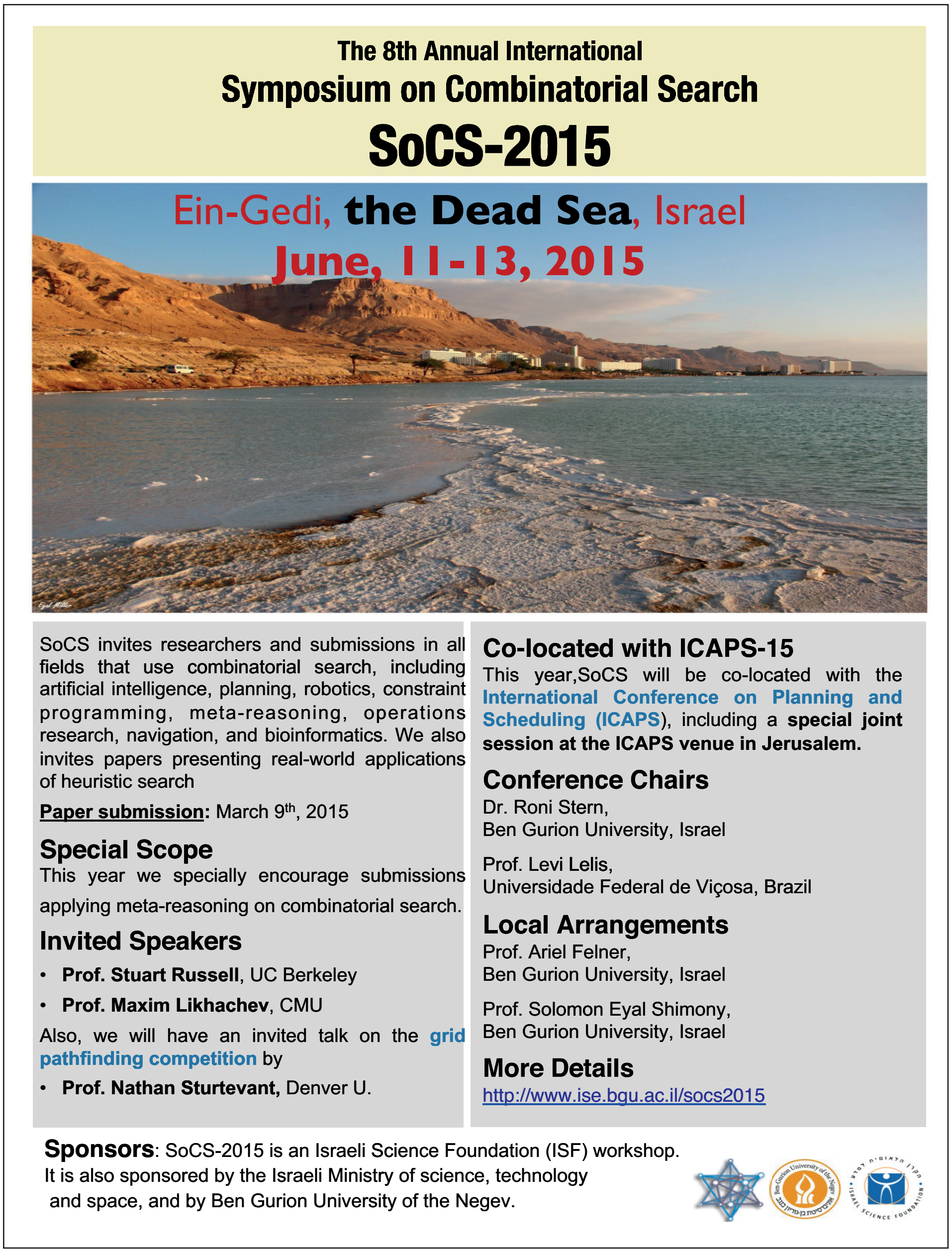
Gulwani, S. 2011. Automating String Processing in Spreadsheets Using Input-Output Examples. In *Proceedings of the 38th Symposium on Principles of Programming Languages*, 317–330. New York: Association for Computing Machinery.

Kandel, S.; Paepcke, A.; Hellerstein, J.; and Heer, J. 2011. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *Proceedings of the ACM Special Interest Group on Computer-Human Interaction* (CHI), 3363–3372. New York: Association for Computing Machinery.

Knoblock, C. A.; Szekely, P.; Ambite, J. L.; Goel, A.; Gupta, S.; Lerman, K.; Muslea, M.; Taheriyan, M.; and Mallick, P. 2012. Semi-Automatically Mapping Structured Sources into the Semantic Web. In *The Semantic Web: Research and Applications, 9th Extended Semantic Web Conferences,* Lecture Notes in Computer science, Volume 7295, 375–390. Berlin: Springer

Knoblock, C. A.; Szekely, P.; Gupta, S.; Manglik, A.; Verborgh, R.; Yang, F.; and Van de Walle, R. 2013. Publishing Data from the Smithsonian American Art Museum as Linked Open Data. In Poster and Demo *Proceedings of the 12th International Semantic Web Conference Posters & Demonstrations Track*. CEUR Workshop Proceedings Volume 1035. Aachen, Germany: RWTH Aachen University.

Lau, T.; Wolfman, S. A.; Domingos, P.; and Weld, D. S. 2003. Programming by Demonstration Using Version Space Alge-



bra. *Machine Learning* 53(1–2): 111–156. dx.doi.org/10. 1023/A:1025671410623

Maali, F.; Cyganiak, R.; and Peristeras, V. 2012. A Publishing Pipeline for Linked Government Data. In *The Semantic Web: Research and Applications,* volume 7295 of Lecture Notes in Computer Science, 778–792. Berlin: Springer.

Marnette, B.; Mecca, G.; Papotti, P.; Raunich, S.; and Santoro, D. 2011. ++Spicy: An OpenSource Tool for Second-Generation Schema Mapping and Data Exchange. In *Proceedings of the VLDB Endowment,* Volume 4, 1438–1441. New York: Association for Computing Machinery.

Raman, V., and Hellerstein, J. M. 2001. Potter's Wheel: An Interactive Data Cleaning System. In *Proceedings of 27th International Conference on Very Large Data Bases*. San Francisco: Morgan Kaufmann.

Taheriyan, M.; Knoblock, C. A.; Szekely, P.; and Ambite, J. L. 2013. A Graph-Based Approach to Learn Semantic Descriptions of Data Sources. In *The Semantic Web — ISWC 2013, Proceedings of the 12th International Semantic Web Conference*. Lecture Notes in Computer Science, Vol. 8218, 607–623. Berlin: Springer.

Wang, G.; Yang, S.; and Han, Y. 2009. Mashroom: End-User Mashup Programming Using Nested Tables. In *Proceedings of the 18th International Conference on World Wide Web,* 861–870. New York: Association for Computing Machinery. dx.doi.org/10.1145/1526709.1526825

Wu, B.; Szekely, P.; and Knoblock, C. A. 2014. Minimizing User Effort in Transforming Data by Example. In *Proceedings of the 19th International Conference on Intelligent User Interfaces*. New York: Association for Computing Machinery. dx.doi.org/10.1145/2557500.2557523

Craig Knoblock is a research professor in computer science at the University of Southern California (USC) and the director of information integration at the USC Information Sciences Institute (ISI). He received his Ph.D. from Carnegie Mellon in 1991. His research focuses on techniques related to information integration, semantic web, and linked data. He has published more than 250 journal articles, book chapters, and conference papers. Knoblock is an AAAI Fellow, a Distinguished Scientist of the ACM, and a past president and trustee of IJCAI. He and his coauthors have been recognized for the Best Research Paper at ISWC 2012 on Discovering Concept Coverings in Ontologies of Linked Data Sources and the Best In-Use Paper at ESWC 2013 on Connecting the Smithsonian American Art Museum to the Linked Data Cloud.

Pedro Szekely is a project leader and research assistant professor at the Information Sciences Institute, University of Southern California, currently working in Craig Knoblock's Information Integration Group, focusing on bioinformatics and education applications where he is working to help biologists, doctors, and educators leverage the vast amounts of data that exist in these domains to enable them to make better decisions. He has published more than 70 conference and journal papers in human-computer interaction, visualization, and multiagent systems, served as the conference chair for two leading human-computer interaction conferences, regularly serves on conference program committees, and also led several research consortiums. Szekely received his M.S. and Ph.D. degrees in computer science from Carnegie Mellon University in 1982 and 1987 and joined USC ISI in 1988.