Translation of Instance Data using RDF and Structured Mapping Definitions

Mehmet Aydar and Austin Melton

Kent State University, Department of Computer Science
`{maydar,amelton}@kent.edu`
`http://www.kent.edu/cs`

Abstract. *We present a healthcare information interoperability project in development. The idea behind the system is achieving semantic interoperability of different healthcare data models, by enabling translation of instance data based on structured mapping definitions, and using RDF as a common information representation. We have developed a framework that allows the domain experts to define linkages between different data elements and utilizes the mappings to translate the data models from one format to another.*

1 Introduction

A healthcare IT environment breeds incredibly complex data ecosystems. Clinical data exist in multiple layers and forms ranging from the heterogeneous of structured, semi-structured, and unstructured data captured in enterprise-wide electronic medical record and billing systems, through a wide variety of departmental, study, and lab-based registries and databases. In many cases the data need to be retrieved from multiple independent sources. Because of the lack of interoperability between disparate data sources, retrieving data from multiple sources is extremely time consuming, wasteful and costly.

Interoperability can be achieved by adopting standards and translations between different standards. There exist many different standards in healthcare, each having its own uses and advantages. It is unrealistic to have one universal standard that fits all the use cases. Therefore an efficient way of translation between different data models is needed. Since each data model can be in a different format, adopting a common information representation model is unavoidable for the translation. In this sense, RDF (Resource Description Framework) [5] is a good choice since it is schemaless and is a commonly accepted information representation model by the semantic web community. Translation requires mappings between the sets of data elements in the different data models. Capturing the mappings between the data models in a structured format enables auto-generation of the translation code.

2 Translation of Instance Data

Figure 1 illustrates the main components of the translation framework and their interactions as proposed in this work. The red labels help show how the trans-

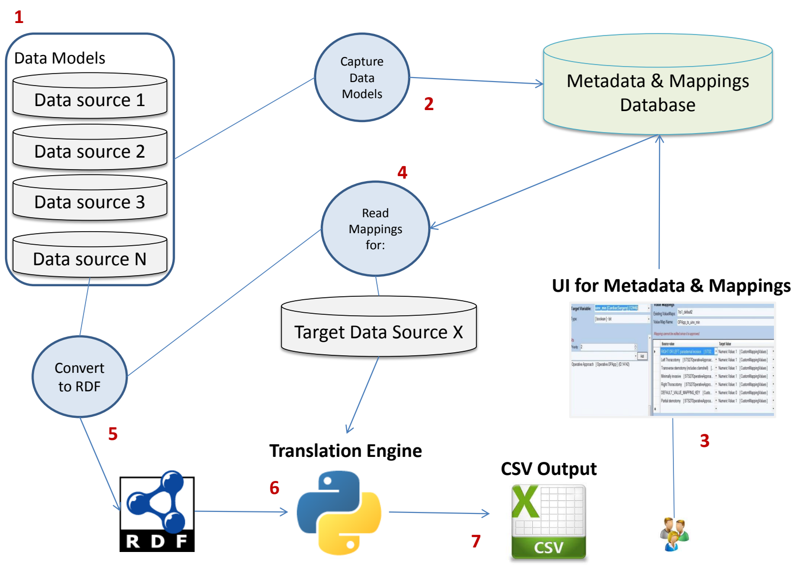

Fig. 1. Translation of instance data

lation components work together. The metadata for each data source (1) are captured and saved in the metadata repository or database (2). Further, the data managers define the linkages (mappings) between the data elements using a user-friendly interface, and these linkages are also saved in the metadata repository (3). In preparation for a translation, the defined mappings from the source data model to the target data model are retrieved (4). The source instance data are represented or lifted to RDF format (5). Then the translation engine uses the source RDF data, the target data model metadata, and the mappings from the source data model to the target data model to translate the source instance data into target data (6). The translation engine produces the results in a CSV (comma separated values) file format, and these results are conveyed to the flattened target data model (7). More details about the components are given in the following sections.

2.1 Metadata Repository & Mapping Tool

For each of the data sources, a data dictionary is captured and stored in a metadata repository [7]. In our work a data dictionary includes data model details for classes, attributes, attribute values, and their associations to each other. As an example, for a relational data model, the metadata consists of tables, table fields, allowed field value sets for the fields, as well as the relationships between different metadata elements such as parent-child relationships.

The metadata repository also includes a mapping schema. The goal is to enable storing the mappings between different data models in a structured format.

The mapping schema is designed so that a target field can be mapped to from multiple candidate source fields that belong to one or more candidate source data models with field-level priorities, and a value set translation from source values to target values, and a pre-defined transformation logic from source fields to the target field. A user-friendly interface is used by the data managers to view the data dictionaries, manage the term definitions and define mappings between different data elements. In addition, the interface also has a test module that lets the data managers write and execute tests to validate their mappings.

2.2 Translation Engine

The flexibility of RDF allows different schemata and models to be converted, represented and connected via RDF. In this work, the translation engine performs the translation on the RDF representation of the source data elements. The conversion from a source data format to the RDF representation differs for each data model, i.e., a different RDF data conversion routine is executed for each data format. The conversion from source instance data to the RDF representation happens on the fly during the translation process; only the specific source instance data, which is required by the mapping, is converted.

The usage of a metadata repository provides valuable benefits in data interoperability. Capturing the mapping knowledge in a structured format provides connectivity between disparate data elements, and it enables an automatic generation of the data translation code to some extent. This leads to more transparency in the transformation process; i.e., in case of any data error it helps to track the source of the problem by checking the documented data mappings.

Data value translation enables the translation of a concept term in one data code system into an equivalent concept term in another data code system. For instance, a male gender can be represented with the term “Male” in one system while it is represented with the term “M” in another system. The associations in the documented value mappings are used to discover the term conversion from “Male” to “M”. In many cases a single target data field is mapped to from more than one source data field. In this case the translation engine needs to know the priority of the source fields and/or a derivation logic to translate from multiple source fields. The priorities and the derivation logic can be defined by the data managers using the mapping interface. The translation engine is then able to generate the translation code by utilizing the defined priorities and the derivation logic.

3 Discussion

Healthcare data interoperability requires the necessary tools, an efficient roadmap and a strong commitment from relevant communities. Several studies have been suggested for this purpose. For instance, the Yosemite Project [4] suggests an ambitious roadmap for healthcare information interoperability on a global scale, by using RDF as a universal information representation and creating a hub for crowd-sourcing translation rules.

In this work we presented a framework to achieve interoperability between different healthcare data models. It tries to alleviate the lack of interoperability solutions and works fairly well on a small scale. However, our approach may need improvements for use on a larger scale. In our work, we assumed that there is already a defined data dictionary in place for each data source. This assumption may not be correct for all data sources. Also, manually defining the mappings can also require significant amounts of time due to the size of the healthcare data. As future work, we plan to work on the following improvements: (1) Creating OWL ontologies for the source data models along with a mapping ontology, instead of using a relational schema. (2) Converting the user interface of the Metadata & Mapping to a web based collaborative interface utilizing common semantic web tools such as Web-Protege [6]. (3) Having the translation engine output the translated results into RDF format and then convert them back to the target data model format as suggested in the Yosemite Project [4]. (4) Auto-generating the data dictionary from the RDF representation of a data source. (5) Auto-generating mappings between different data models using graph node similarity metrics and suggesting the auto-generated mappings to the data managers, utilizing our previous studies [1, 3, 2].

Acknowledgments Our work is implemented in the Cleveland Clinic Cardiology Application Department to support interoperability between legacy data models. The authors would like to thank the Kent State University Semantic Web Research Group (SWRG) members, Cleveland Clinic Cardiology Application Group members and Semantic Web Health Care and Life Sciences (HCLS) Interest group members for their helpful feedback.

References

1. Mehmet Aydar, Serkan Ayvaz, and Austin C Melton. Automatic weight generation and class predicate stability in rdf summary graphs. In *Workshop on Intelligent Exploration of Semantic Data (IESD2015), co-located with ISWC2015*, 2015.
2. Mehmet Aydar and Austin C Melton. Rinsmatch: a suggestion-based instance matching system in rdf graphs. In *Workshop on Ontology Matching (OM-2015), co-located with ISWC2015*, 2015.
3. Serkan Ayvaz, Mehmet Aydar, and Austin C Melton. Building summary graphs of rdf data in semantic web. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th International*. IEEE, 2015.
4. David Booth et al. Yosemite manifesto on rdf as a universal healthcare exchange language, 2013.
5. Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.
6. T Tudorache, J Vendetti, and N Noy. Web-protege: A lightweight owl ontology editor for the web. sdr. *Cit. on p*, 2008.
7. Wikipedia. Data dictionary — wikipedia, the free encyclopedia, 2015. [Online; accessed 3-July-2015].