Demonstration: A Design Pattern for Representing Idiosyncratic Ontologies in Industrial Applications

Anne Hunt [1]

[1] *Realtor.com, Austin, Texas*

Abstract

Applications that intelligently handle data from multiple sources often need to manage and present incoming data that is structured by source system ontologies that are widely different from each other. Hereafter, these are referred to as idiosyncratic ontologies. In many cases, idiosyncratic ontologies are mutually inconsistent. Because of the nature of the data and the importance of these applications within such domains as health information, research data, and financial data, it is necessary to both provide a coherent experience for end users and to maintain a computationally accessible provenance for the source data. This is a demonstration of a design pattern that has been effectively re-used across multiple domains.

Keywords

Idiosyncratic ontologies, representation, industry

1. Introduction

This demonstration illustrates a repeatable pattern for managing data structured by idiosyncratic ontologies. We especially focus on singular referential terms. This pattern is currently used to support important software applications at several companies, where data sources have differing and inconsistent ontologies whose original provenance must be maintained while the data is "made sense of" for users of the software systems.

In application systems design, as in philosophy of language, trouble arises when there is no one-to-one correspondence between a term and the intended referent. And the situation is exacerbated when there is no general, community-wide agreement on the semantics of the terms. This type of situation arises in the real world more often than one might expect. For one common example, suppose that Google Home's underlying data representation is such that it has data structures representing two living room lamps in my home when, in the real world, there is only one. A more serious example occurs when an electronic health record (EHR) has a representation of two people called "Jane Doe" attending a specific medical practice when there is in fact only one such person. And these situations aren't merely bugs in the data that should be corrected, for example by merging the records. To see that this is the case, suppose someone has two identities that they need maintained for reasons of privacy, and some helpful programmer "fixes" the problem by merging records from each identity.

This is a demonstration of one way to properly represent data sourced from a variety of messy and inconsistent ontologies, such that the applications built on this data behave according to these requirements:

- End users are given a coherent experience
- Inferences such as identity resolution can be undone when defeasible

FOIS 2021 Demonstrations, held at FOIS 2021 - 12th International Conference on Formal Ontology in Information Systems, September 13-17, 2021, Bolzano, Italy

EMAIL: annejudehunt@gmail.com (A. Hunt)
ORCID: 0000-0001-9805-3538 (A. Hunt)

- The original data must be maintained, and the provenance of inferences must be available
- New data can be added to the system and data can be removed from the system
- End users see the data according to appropriate permissions (e.g. private identities are maintained in appropriate contexts)

2. Cases

At Castlight Health, a public company, we ran into the problem of idiosyncratic ontologies with data received from hundreds of other companies (usually insurance companies) daily. This data represented medical providers – doctors, practices, clinics, hospitals, and other medical facilities – as well as networks of providers. For example, a single record could represent a facility, such as a clinic, or provider group such as a partnership of doctors with related specialties. The name of one of these facilities could be almost identical to that of an individual doctor. Where a doctor's name was, for example, Dr. John Smith, the name of his clinic might be "The Doctor John Smith Practice."

Castlight Health's application was a consumer-facing product that helps consumers pick in-network healthcare providers. The magnitude of the data we received was on the order of millions of records per week. While the data was labeled as to which of these types of named entities was represented – whether it was a practice or a doctor, for example – that labeling was not reliable in all cases. Worse, many of the data providers had inconsistent representations – for example, one insurer's data held that "Mary Washington Hospital" was the same as "Mary Washington Center", while another insurer's data held they were two different providers, and a third held that there was yet another provider, "M. Washington Hospital Center" that was not identical to either of the other two. The data was accompanied by National Provider Identifiers (NPI) which are meant to be unique, but in reality are not.

The following table illustrates the type of data received.

NPI	Name	Location	Provider Type
1234	Stanford Hospital	45 Middlefield, Palo Alto, CA 22345	Hospital
1234	Stanford Radiation Center	47 Middlefield, Palo Alto, CA 22345	Hospital
3456	Dr. Jane Smith	123 Chaucer Street, Palo Alto, CA 22344	Practitioner
3456	Dr. Jane Smith Practice	123 Chaucer Street, Palo Alto, CA 22344	Group
3456	Dr. John Doe	123 Chaucer Street, Palo Alto, CA 22344	Practitioner
7899	Dr. J. Smith	123 Chaucer Street, Palo Alto, CA 22344	Practitioner

Before we began our identity resolution update, Castlight's algorithms and other data sources indicated that any entities sharing a single NPI were one and the same, and that is the information that we presented to consumers. Once we dug into the data by hand in specific cases (such as illustrated above, we could see that NPIs were not unique. We also discovered that the best understanding of the identity of the entities changed over time and with additions to our data sources. While a human can easily see that Dr. Smith, her practice, and her co-worker Dr. Doe are three unique entities, and that Dr. J. Smith is probably the same person as Dr. Jane Smith, in order to infer this at scale we had to deploy a combination of machine learning and rule-based inference.

In a separate company, we ran into a very similar problem with incoming data, in this case for both medical practitioners and patients. At Medici, a health technology startup, we received patient data from multiple providers' electronic health records. Medici's application was a doctor-facing and a consumer-facing product that allowed doctors and patients to consult remotely and share data in a

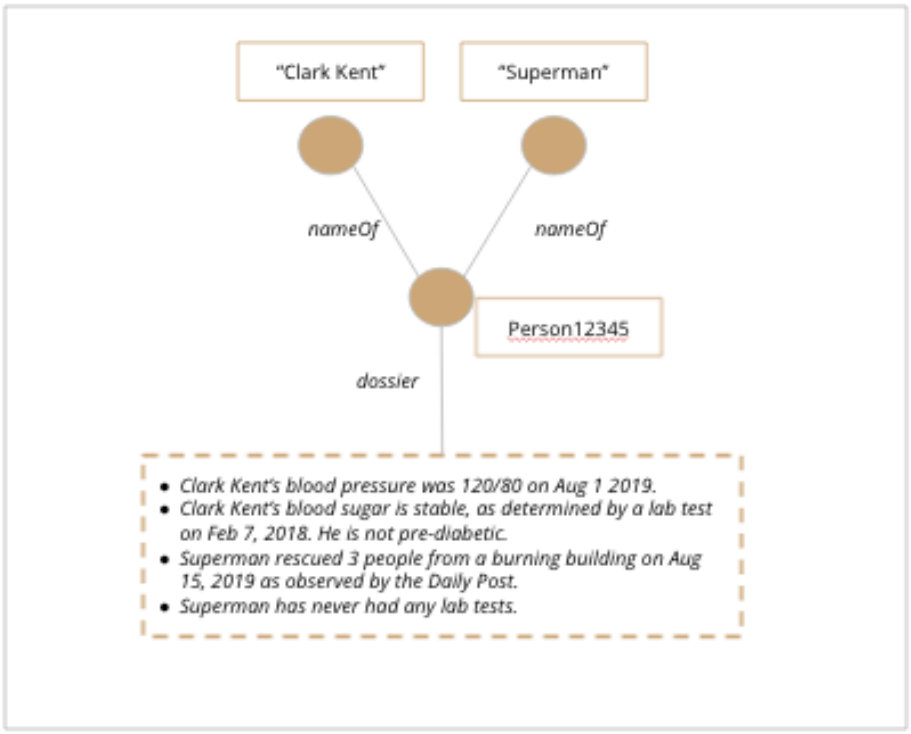
manner compliant with applicable laws and regulations. Medici's system conducted some amount of identity resolution on data about both patients and doctors, so that data from various systems could be stitched together for users to view. For reasons similar to those at Castlight, with the addition of complexities around consumer data, data from different systems about the same individual could not simply be merged.

3. Design Pattern

To describe the design pattern used in these and other cases, I'll use the example of an EHR (electronic health record). An EHR can be thought of as something like a dossier of data about a person. EHR systems are largely *for the purpose of* storing and managing information about real people.

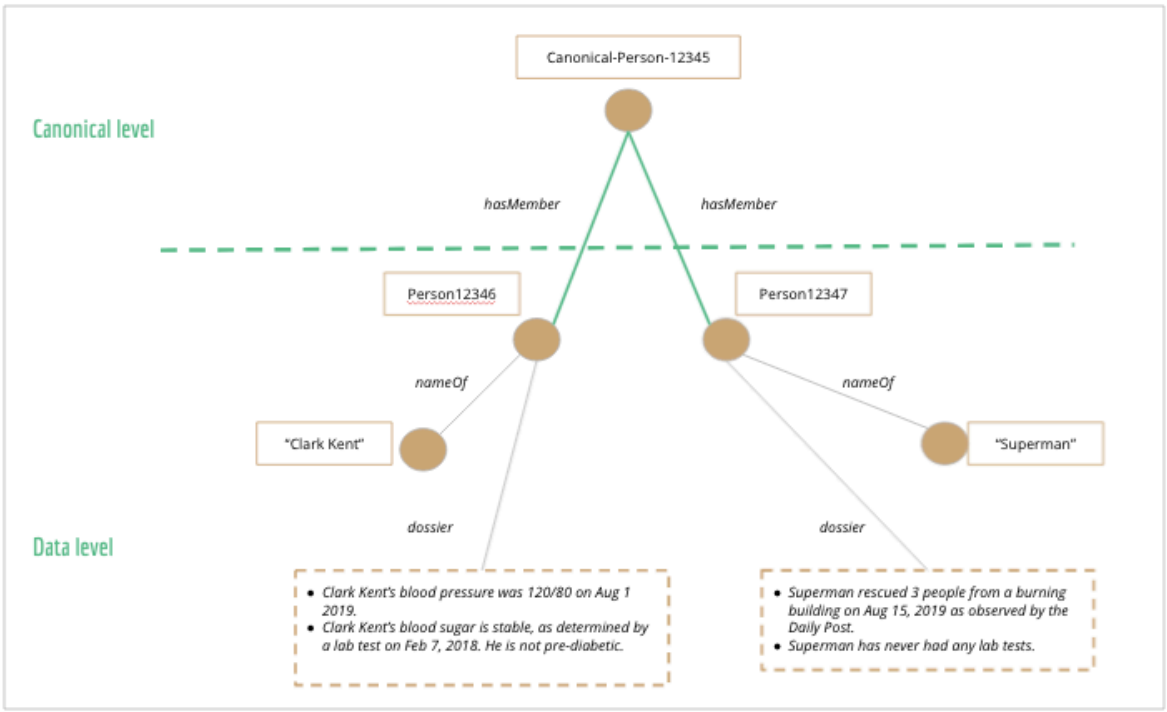
Consider the case where a patient has two names, Clark Kent and Superman, along with two associated and separate identities that he would like to keep separate for reasons of his own. Then the EHR for that person will consist of a set of facts, some of which are associated with one of the names (identities) and some that are associated with another.

The simplest approach, and the one that seems intuitively correct, would be to represent the single person and associate that entity with the two names along with the facts in the EHR. The following diagram illustrates this model:



While the model itself is relatively simple, it turns out that using this model within a software system provides no simple way to retrieve data associated with one of the identities versus the other. You would be reduced to writing a program to retrieve all of the facts, and then parsing through the natural language to find the right names (which of course can take various surface forms, including mis-spellings). It's too difficult to separate the "Clark Kent" facts from the "Superman" facts. Additionally, this sort of model doesn't allow for data provenance, for example if the Clark Kent data is sourced from one medical practice and the Superman data is sourced from another.

The approach that worked is illustrated here:



Using an algorithm that combined machine learning and inference, we were able to infer that the Clark Kent data and the Superman data pertained to a single person in the real world. We represented that real world person as what we decided to call a "canonical" entity with an associated persistent, unique identifier to be used and relied upon by downstream systems. At the data level, we represented the data received about each identity separately. This allowed us to easily and programmatically retrieve data associated with each identity separately, for purposes of maintain privacy, while also creating a correct representation (to allow us to do such things as get an accurate count of the number of patients in our system).

4. Demonstration Method

In this demonstration, we go into detail about these cases and show the ontology used to correctly capture the semantics of the data while forming a foundation for applications that met requirements around data provenance and privacy. This representation pattern is like what would be needed to capture the semantics of such statements as:

- John knows that Lois Lane believes Clark Kent can't fly.
- John knows that Lois Lane believes Superman can fly.

We will use a small ontology, developed in Protégé (https://protege.stanford.edu), along with a set of slides with illustrations of the intended model, some rejected models for comparison, and a set of first order logic formalizations. We will map the demonstration ontology to the formalizations, and describe how the pattern is easily extended to similar cases in different domains and covering different types of terms, such as classes.