Modeling and Exploiting Knowledge through Semantic Representations

ALAN CHAPPELL
National Security Directorate, Human Centered Analytics Team Lead
Discovery 2020 Short Course on Semantic Data Analysis
Technologies developed to help computers work *as though* they understand semantics
- Exploit meaning, rather than only data

Ontology
- Formal representation of concepts and relationships between them
- Defines a vocabulary

Knowledge Base
- Collection of entities and relationships that conform to an ontology
- May contain instances drawn from multiple ontologies

Tools may exploit ontologies, knowledge bases, or both
Core Semantic Technology Standards

- **Query:** SPARQL
- **Ontology:** OWL
- **Taxonomy:** RDFS
- **Data Interchange:** RDF
- **Syntax:** XML (Turtle); Namespace

- Why standards? Interoperability!
- Not like languages. Used in an integrated manner.
- Layers in the “stack” build on lower layers. Augmenting not replacing.
- When working at a level, use technologies below.
Semantic Technology Standards

- **Syntax:**
  - XML (Turtle);
  - Namespace

- **Data Interchange:**
  - RDF

- **Taxonomy:**
  - RDFS

- **Ontology:**
  - OWL

- **Query:**
  - SPARQL

- **Finding Triples**

- **Triples**

- **Expressive Semantics**

- **Domain Types and Relationships**

- **Physical Form**

May 20, 2013
Entity-Relationship Diagrams

Intelligence community has long used Entity-Relationship diagrams

- Analyst Notebook/Palantir

- E-R diagrams are collections of **triples**.

- Except, the meaning is provided by the reader.
Resource Description Framework (RDF)

▶ Language for representing information about resource in the WWW.
  ▶ Generalize: If you can give it a URI, you can talk about it.
  ▶ Description is in the form of properties and values.

▶ Those parts are called the Subject, Predicate, and Object.
▶ Combined, the **three** make a Statement.
▶ RDF is all about **triples**.
But URIs are still just strings. What is new?

How does a word have “meaning”?
- Because we agree on the meaning when that particular string is used.

URI is (by definition) unique

If we use the same URI, we are talking about the same thing.
- Applies to both resources and properties.

Reuse provides meaning.
Namespaces

- Namespaces – distinguish my vocabulary from yours
  - URI gives container for encapsulating
    - http://doe.gov/energy/Nuclear
    - http://doe.gov/energy/Coal
  - Prefix gives shorthand
    - doe = http://doe.gov/energy
    - doe:Nuclear
    - doe:Coal
  - Different container, different resource
    - ge:Nuclear
Serializing

- RDF designed to serialize to XML
  - Files or streaming
  - Original standard because of web origins
  - But it is verbose, bloated, and difficult to read

- Other RDF-centric formats developed by communities
  - More compact
  - Easier to read

- Important questions
  - Do your tools support the format?
  - Where are the triples?
Where are the Triples?

**RDF/XML**

```xml
<ObjectProperty rdf:about="&s;hasSkill">
  <rdfs:domain rdf:resource="&s;Person"/>
  <rdfs:range rdf:resource="&s;Skill"/>
</ObjectProperty>
```

**OWL/XML**

```xml
<Declaration>
  <ObjectProperty IRI="#hasSkill"/>
</Declaration>
<ObjectPropertyDomain>
  <ObjectProperty IRI="#hasSkill"/>
  <Class IRI="#Person"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
  <ObjectProperty IRI="#hasSkill"/>
  <Class IRI="#Skill"/>
</ObjectPropertyRange>
```

**Turtle**

```
s:hasSkill rdf:type owl:ObjectProperty ;
  rdfs:domain s:Person ;
  rdfs:range s:Skill .
```
Literals serialized as strings, but may be other types.

- Record types so tools know how to process
- RDF shorthand for literal data types
- Optional language tag on Strings

```
s:arc rdf:type s:Person ;
s:age "50"^^xsd:integer ;
s:birthdate "1962-06-14"^^xsd:date ;
s:comment "Suspicious character"@en .
```
RDF oddities

- **Blank Nodes**
  - Group properties in related set without URI for the concept
  - Only locally unique!

**Example SPARQL query:**

```sparql
s:arc s:address _:alanaddress .
   _:alanaddress s:street "1100 Dexter Ave" ;
   s:city "Seattle" ;
   s:state "WA" ;
   s:zipcode "98109" .
```

**Example RDF graph:**

- `http://www.../arc`
- `http://www.../address`
- `"1100 Dexter Ave"`
- `"WA"
- `"Seattle"
- `"98109"`

**Example RDF triple:**

```
s:arc s:address [ s:street "1100 Dexter Ave" ;
   s:city "Seattle" ;
   s:state "WA" ;
   s:zipcode "98109" ] .
```
RDF provides a limited vocabulary.

How do we define our own?

RDFS adds new vocabulary for defining:
- classes of resources and
- properties of those classes

It is still RDF!
- The standard is just providing URIs representing concepts.
- We reuse those URIs to exploit the meaning.
RDFS provides mechanisms to describe properties that characterize classes

- What classes participate in the property.
  - Domain = source, Range = target
- Refinements of properties.

```
s:Person
  rdfs:domain s:Person
  rdfs:range rdf:Property

s:Female
  rdfs:domain s:Female
  rdfs:range rdf:Property

s:Person
  rdf:type rdf:Property

s:Female
  rdf:type rdf:Property

s:hasParent
  rdfs:domain s:Person
  rdfs:range s:Person
  rdfs:subPropertyOf rdf:Property
  rdf:type rdf:Property

s:hasMother
  rdfs:domain s:Female
  rdfs:range s:Female
  rdfs:subPropertyOf rdf:Property
  rdf:type rdf:Property
```
RDFS Extras

- rdfs:comment
  - Property who’s value is a string the (typically) describes the authors intent or meaning.

- rdfs:seeAlso
  - Property that can point to a related concept
Beyond simply hierarchy
- Need richer descriptions of classes and relationships
- Support formal reasoning so computer can do some of the work

OWL sub-languages (profiles, fragments, species)
- Subsets of the OWL capability that trade off expressiveness and reasoning time
- Choose based on need of problem being solved.
Class Descriptions

- Refine representations of classes
  - owl:disjointWith – instances can NOT be both

- New ways of defining Classes
  - owl:oneOf – enumeration
  - owl:unionOf – class defined by the combination of other classes [also intersection and complement]
Property Descriptions

- Easier ways to define properties
  - owl:ObjectProperty – Object is a Resource
  - owl:DatatypeProperty – Object is a combination Literal and associated xsd data type.

- Logical descriptors
  - FunctionalProperty, SymmetricProperty, TransitiveProperty

- Restrictions
  - Required numbers – owl:maxCardinality, owl:minCardinality
    - Parent MUST have at least one Child [minCardinality 1]
  - Local range restriction – owl:allValuesFrom, owl:someValuesFrom
    - Animal hasOffspring Animal
    - Elephant hasOffspring Calf [allValuesFrom Calf]
    - Human hasOffspring Child [allValuesFrom Child]
owl:Thing
- Class of everything. Always the highest level class.

owl:sameAs
- Two instances are the same real-world thing
- Anything that is true for one is also true for the other.

owl:imports
- Read in external ontology so you can exploit the classes and properties
- Supports modularization
Building Ontologies

- It is all RDF.
- Meaning is in use of common, agreed upon URIs.
- Each layer in the stack adds formalism and richness.
- Designed to support reasoning.
- But what about that Query box?
- Time to get our hands dirty!
Protégé – Ontology editor and viewer
Name your Ontology

 Ontology ID

Please specify the ontology IRI.

The ontology IRI is used to identify the ontology in the context of the world wide web. It is recommended that you set the ontology IRI to be the URL where the latest version of the ontology will be published. If you use a version IRI, then it is recommended that you set the version IRI to be the URL where this version of the ontology will be published.

Ontology IRI

http://www.yourOrg.gov/example/people.OWL

Version IRI

http://www.semanticweb.org/ontologies/2013/4/Ontology1367557013857.OWL

☐ Enable Version IRI

Go Back  Continue  Cancel
Give a local place to store your Ontology
Choose a default persistence format

Ontology Format

Please select the format in which the ontology will be saved (by default).

Note that the Manchester OWL Syntax does not support all OWL constructs (e.g., GO's and annotations of undeclared entities) and the Latex format cannot be reloaded.

Turtle
You have an empty Ontology
Add a namespace prefix
Switch to Classes tab
Add a subClass to Thing called Person
Now add the class Organization
Make Person and Organization disjoint
Switch to Data Properties tab
Create the name Property
Add Person and Organization to Domain

[Image of a computer interface showing the process of adding a person or organization to a domain.]
Add age to Person with type nonNegativeInteger
Switch to Object Properties tab
Add the employed_by property
Add the domain and range
Add the inverse property employs
Now save your Ontology

► File … Save
Now let’s look at the file

Find the triples!
Issues in Modeling

- **Multiple Inheritance**
  - Powerful modeling technique
  - Many potential pitfalls
  - Some design patterns explicitly avoid use

- **Do you need to create another ontology?**
  - Reuse **CAN** save time and money

- **Different philosophies on ontology scale**
  - **UBER Ontology**
    - Large ontology to address many domains and needs
    - More likely to be able to share data
    - Can be complex and hard to understand
  - **Everybody builds their own ontology**
    - Semantics and scale to your problem
    - Sharing data requires **alignment** to other Ontologies

May 20, 2013
Working with Instances

- Modeling performed in the Ontology vocabulary to describe items of interest.
- It is the items that are the real focus, not the Ontology.
- Tools to work with instances
  - Protégé – may be too low level for domain users
  - Higher-level (domain-specific) tools
    - AKEA – Analyst-Driven Knowledge Enhancement and Analysis
    - Create, Edit, Visualize, Explore, Query
Working with Instances
Issues in Working with Instances

▶ Entity Disambiguation
  - When data comes from different sources, how do we know when the instances are actually the same
    - People don’t use URIs!

▶ Storing Instances
  - Triple store is common
  - Current approaches exploit Quad store
  - New-ish technologies
    - Performance questions but improving quickly