Overview

- Types and examples of ontologies
- Representation formats
- Ontology Learning in Practice
Types and examples of ontologies
Types and examples of ontologies

- **Upper level ontologies**
  - Basic Formal Ontology (BFO)
  - Cyc
  - Dublin core
  - Friend of a Friend (FOAF)
- **Domain ontologies**
  - Biomedicine: Unified Medical Language System (UMLS)
  - Finance: eXtensible Business Reporting Language (XBRL)
- **Application ontologies**
  - Clinical documentation: SNOMED CT
- **Lexical or terminological ontologies**
  - WordNet, Simple Knowledge Organization System (SKOS)
  - EuroVoc, IATE
Upper level ontologies

Captures concepts, relations, and axioms that apply across multiple domains, such as the Basic Formal Ontology (BFO) applied to an emotion ontology below.

Domain Ontologies

Represents concepts, relations, and axioms that are specific to a domain of discourse.

Application ontologies

Engineered for a specific use or application scope that is specified by testable use cases, such as clinical documentation with SNOMED CT depicted below.
Lexical and terminological ontologies

Ontology that consists of terminological entries or synonym sets and lexico-semantic relations, as exemplified with SKOS below.

Representation formats
Representation formats

Semantic Web stack of technologies:

- User interface and applications
- Trust
- Proof
- Unifying Logic
  - Querying: SPARQL
  - Ontologies: OWL
  - Rules: RIF/SWRL
  - Taxonomies: RDFS
  - Data interchange: RDF
  - Syntax: XML
  - Identifiers: URI
  - Character Set: UNICODE

Source: https://commons.wikimedia.org/wiki/File:Semantic_web_stack.svg
Dagmar Gromann, 25 January 2019  Semantic Computing
Resource Description Framework (RDF)

**Goal:** Provide a structure (framework) to describe things (resources). It consists of three basic elements that allow us to model simple ontologies:

- Resources - things being described
- Properties - relations between things
- Classes - abstract concepts used to group things

**Structure = RDF triples:** `<Subject> <Predicate> <Object>`

- `<SemanticComputing> <hasLecturer> <DagmarGromann>`

All things are uniquely identified with a Uniform Resource Identifier (URI):

http://www.foaf.com/Person#DagmarGromann
RDF example (informal)

Alice

is a friend of

BOB

is interested in

The Mona Lisa

is a

Person

is born on

14 July 1990

was created by

Leonardo Da Vinci

is about

La Joconde à Washington

Source:

https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/

Dagmar Gromann, 25 January 2019  Semantic Computing
Web Ontology Language (OWL) Basics

- **Axioms**: basic statements that an OWL ontology expresses, e.g. "it is raining", "every man is mortal" - asserted to be true)

- **Entities**: elements used to refer to real-world objects, e.g.
  
  Class: Course, Student: Mary

- **Expression**: combinations of entities to form complex descriptions from basic ones, e.g.

  Entities need to be declared to be of a specific type, e.g.
  
  Individual: SemanticComputingObjectProperty: belongsTo
  Class: Course
OWL modeling

• Class Expressions:
  – Conjunction (and): \( C \cap D \)
  – Disjunction (or): \( C \cup D \)
  – Negation (not): \( \neg C \)

• Property Expressions:
  – Quantifier: \( \exists r.C \) (existential; some), \( \forall r.C \) (universal; all)
  – Cardinality: \( \geq n \ r.C \) (min), \( \leq n \ r.C \) (max)

• Class Axioms:
  – Subclass: Student \( \sqsubseteq \) Person (Student is_a Person)
  – Equivalence: \( C \equiv D \) (C sameAs D)
  – Disjointness: DisjointClasses(Boy, Girl) or boy \( \cap \) girl \( \sqsubseteq \bot \)

• Property Axioms: same as for class + transitive, symmetric, reflexive, functional, inverse
Web Ontology Language (OWL)

Model complex ontologies. Different types of syntaxes:

- **Turtle notation:** `Student rdfs:subClassOf Person`
- **Manchester syntax:**
  ```
  Class: Student
  SubClassOf: Person
  ```
- **Description Logic (DL) syntax:** `Student ⊑ Person`
- **First Order Logic (FOL) syntax:** `∀x. Student(x) → Person(x)`
- **OWL/XML:**

```xml
<?xml version="1.0"?>
<!DOCTYPE Ontology [ 
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" > 
 ]>
<owlx:Ontology owlx:name="http://www.example.org/person"
  xmlns:owlx="http://www.w3.org/2003/05/owl-xml">
  <SubClassOf>
    <Class URI="&example;Student"/>
    <Class URI="&example;Person"/>
  </SubClassOf>
</owlx:Ontology>
```
Prefixes; Ontology imports
Declaration( NamedIndividual( :John ) )
Declaration( NamedIndividual( :Mary ) )
Declaration( Class( :Person ) )
Declaration( Class( :Woman ) )
Declaration( Class( :Man ) )
Declaration( ObjectProperty( :hasWife ) )
Declaration( ObjectProperty( :hasSpouse ) )
Declaration( DataProperty( :hasAge ) )

ObjectPropertyDomain( :hasWife :Man )
ObjectPropertyRange( :hasWife :Woman )

SubClassOf( :Woman :Person )
SubClassOf( :Man :Person )
EquivalentClasses( :Person :Human )
DisjointClasses( :Woman :Man )

SubObjectPropertyOf( :hasWife :hasSpouse )

ObjectPropertyAssertion( :hasWife :John :Mary )
DataPropertyAssertion( :hasAge :John "51"^^xsd:integer )

Source: https://www.w3.org/2007/OWL/wiki/Primer#Appendix:_The_Complete_Sample_Ontology
OWL vs. RDF

- RDF: describes simple facts in form of subject-predicate-object triples (plus schema that let’s you specify type)
- OWL: adds semantics to properties and classes and allows you to make statements about two things at a time (samAs, transitivity, etc.)
Ontology learning in practice
# Ontology learning tasks

Bees are insects that produce honey. They have six legs. Bees live only in beehives - or just hives. Maya and Flip are bees. Maya, in particular, is a notable bee. Maya and Flip are friends.

<table>
<thead>
<tr>
<th>Axiom</th>
<th>$\text{Bee} \sqsubseteq \text{Insect} \sqcap \exists \text{produce.Honey} $</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation</td>
<td>$\text{produce}(\text{Bee}, \text{Honey})$</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>$\text{is}_a(\text{Bee}, \text{Insect})$</td>
</tr>
<tr>
<td>Concept</td>
<td>Beehive</td>
</tr>
<tr>
<td>Synonym</td>
<td>{beehive, hive}</td>
</tr>
<tr>
<td>Term</td>
<td>bee, beehive, hive, honey, ...</td>
</tr>
</tbody>
</table>

Ontology learning approaches


- **LearningDL**: Ma, Y. and Distel, F. (2013). Learning formal definitions for SNOMED CT from text. In Conference on Artificial Intelligence in Medicine in Europe, pp. 73-77. Springer


Ontology learning approaches

Approaches until 2018:

• heavy use of NLP toolkits and corpora
• strong relying on hand-crafted rules and patterns
• targeting different source and target languages

Shared challenges:

• axiom learning challenging - mostly lightweight ontologies
• high cost of maintenance and evolution
One idea to learn formal axioms has been to benefit from intensional definitions (as opposed to extensional, object-oriented definitions):

- **Definiendum**: concept being defined (e.g. "a bee")
- **Definitor**: usually a verb introducing the definition (e.g. "is")
- **Definiens**: the genus phrase (e.g. "an insect")
- **Differentiae**: characterizations with respect to genus (e.g. "that produces honey")
Transforming NL to DL

All extralogical symbols are taken directly from the sentence:

bee ⊑ insect ⊎ Ʌ ⊏ produces ⊌ honey
A bee is an insect that produces honey

Major challenges:

• Dataset NL to DL
• NMT architecture


Dagmar Gromann, 25 January 2019
Semantic Computing
Dataset

We need many, good examples:

• Bees are insects that produce honey
• A bee is also an insect that produces honey
• Every bee is an insect that produces honey
• A cow is a mammal that eats grass

Dataset needs to:

• cover MANY syntactic variations of identical semantic contents
• cover many domains
• has annotated <sentence, axiom> pairs

No such dataset is currently available!
A NP is a NP that VB NP
C₀ ⊑ C₁ ⋈ ∃R₀.C₂

Templates: structural regularities beyond meaning.

Dagmar Gromann, 25 January 2019
Dataset Templates

\[ V \]

Context-Free Grammar

\[ \text{random!} \]

\[ \text{Every } C_0 \ R_0 \text{ at least } NUM \ C_1 \]
\[ C_0 \supseteq \geq NUM \ R_0 . C_1 \]

Every innocent exile craves at least 100 towers.

NMT processes

Train two processes: copy or emit

Architecture: GRU-based


Dagmar Gromann, 25 January 2019 Semantic Computing
Evaluation metrics

Avg. Per-Formula Acc. \[ FA(\hat{\mathcal{F}}, \mathcal{F}) = \frac{CF}{M} = \sum_{k=1}^{M} \begin{cases} 1, & \text{if } f^k = \hat{f}^k \\ 0, & \text{otherwise} \end{cases} \] fully automated

Avg. Edit Distance \[ ED(\hat{\mathcal{F}}, \mathcal{F}) = \sum_{k=1}^{M} \frac{\delta(f^k, \hat{f}^k)}{M} \] semi-automated

Avg. Per-Token Acc. \[ TA(\hat{\mathcal{F}}, \mathcal{F}) = \frac{\sum_{k=1}^{M} \sum_{j=1}^{T_{fk}} \begin{cases} 1, & \text{if } f_{j}^k = \hat{f}_{j}^k \\ 0, & \text{otherwise} \end{cases}}{\sum_{k=1}^{M} T_{fk}} \] quick control

<table>
<thead>
<tr>
<th>training set size</th>
<th>FA</th>
<th>ED</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>0.61</td>
<td>2.48</td>
<td>0.92</td>
</tr>
<tr>
<td>5000</td>
<td>0.84</td>
<td>0.60</td>
<td>0.98</td>
</tr>
<tr>
<td>10000</td>
<td><strong>0.89</strong></td>
<td>0.47</td>
<td><strong>0.99</strong></td>
</tr>
<tr>
<td>20000</td>
<td>0.81</td>
<td><strong>0.46</strong></td>
<td>0.98</td>
</tr>
</tbody>
</table>
Future Directions

- Create a more varied dataset
- Test different architectures
- Combine neural architecture with knowledge representation approaches
Review of Lecture 13

- Which different types of ontologies do you know?
- Can you give a specific example for those types?
- How can ontologies be represented? What is the difference between RDF and OWL?
- What do non-neural ontology learning approaches have in common? What are main challenges?
- How does the NMT approach work? How can it be evaluated and how well did it perform?
- What is needed to perform ontology learning with deep learning?
- What could potential future directions be?