Generative AI and Symbolic Knowledge Representations
LLMs, Knowledge and Reasoning
4

Damir Cavar & Billy Dickson
ESSLLI 2024

August 2024

Agenda

- WordNet and NLTK
- OWL and Ontologies
- DBpedia, YAGO, etc.

WordNet and NLTK

- See Notebook on GitHub:
  - https://github.com/dcavar/python-tutorial-for-ipython/blob/master/notebooks/WordNet%20and%20NLTK.ipynb

NLP and Graph Infrastructure

- Knowledge Graphs as RESTful Microservices
  - YAGO integrated in Apache Jena with TDB, SPARQL interface, Lucene index
  - ConceptNet using remote API
  - Microsoft Concept Graph via interface to MongoDB
  - DBpedia using remote API, possible setup as for YAGO
  - SPARQL-based n-hop search and string-similarity search (mutli-lingual)

- Generated Graphs
  - Neo4J using Cypher, GQL
  - Stardog using SPARQL
  - Fluree
  - Open format based on abstract graph class

Lexicon vs. Ontology

- Lexicon is flat:
  - Part-of-speech information
  - example usage
  - meanings
  - list of related terms
  - Thesaurus might offer relational information that extend the information found in common dictionaries

Ontology

- Encodes knowledge about some domain
- Describes concepts and relationships between concepts
- W3C standard ontology language:
  - OWL
  - W3C OWL working group

OWL

- Specific ontology language
  - description of concepts
  - operators: e.g. intersection, union, negation
  - using logic and reasoners

OWL

- Consists of
  - Individuals (in Protégé: Instances)
  - Properties (in Protégé: Slots)
  - Classes (in Protégé: also Classes)

OWL Individuals

- Objects in the domain that we describe, model or are interested in
- No Unique Name Assumption (UNA)
  - various names might refer to the same individual: *Barack Obama*, *the President of the United States* (if uttered during a respective time interval), *President Obama*, etc.
  - In OWL: this must be explicitly stated that individuals are the same or different (otherwise they *might* be same or different)
- Individuals also: Instances (of Classes)

OWL Properties

- Binary relations on individuals
  - link two individuals together
  - E.g. *livesIn* links *Damir* to *Ann Arbor*
- Can have inverses: *hasOwner* and *isOwnedBy*
- Can have a single value: *functional*
- Can be *transitive* and *symmetric*
  - if some property for A and B, and B and C, then also for A and C
  - A *isSibling* B would be symmetric, but not *isBrother*

OWL Classes

- Sets that contain Individuals
- Described formally: the precise requirements for membership of the class
- May be organized into superclass-subclass hierarchies: Taxonomies
    - Subclasses specialize their superclass: are subsumed by
        - *All cats are animals*, or *All members of the class Cat are members of the class Animal*, *Being a Cat implies that you are an Animal*
- ***Concept*** often used instead of ***Class***, Classes are concrete representations of concepts

1

Protégé

- Web site...
  - download Version 5.5 Beta (October 2018)
  - documentation
  - CO-ODE web site
  - Protégé OWL tutorial (The University of Manchester)

Protégé

- Intro based on the University of Manchester tutorial v1.3 tutorial

OWL

- OWL Properties represent relationships
    - Object properties
        - link individual to individual
    - Datatype properties
    - Annotation property
        - Add information, e.g. metadata, to classes

OWL

- Select ‘Object Properties’ tab: Use the ‘Add Object Property’ button to create a new Object property.
- Name the property to *hasIngredient* using the ‘Property Name Dialog’ that pops up
- Naming convention:
  - Property names start with a lower case character
  - There are no spaces in property names

OWL

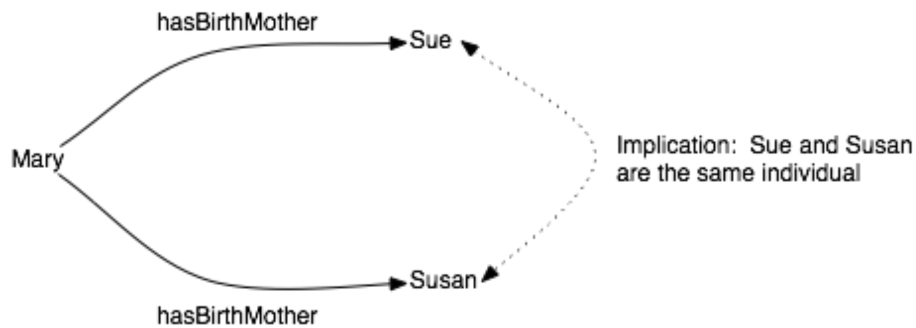
- Add two more properties as a sub-property of *hasIngredient*
  - *hasTopping*
  - *hasBase*

OWL

- Properties: binary relations
  - hasChild
  - hasParent
  - hasBirthMother
  - isBirthMotherOf
  - hasAncestor
  - hasSibling
  - ...
- Make implications possible

OWL

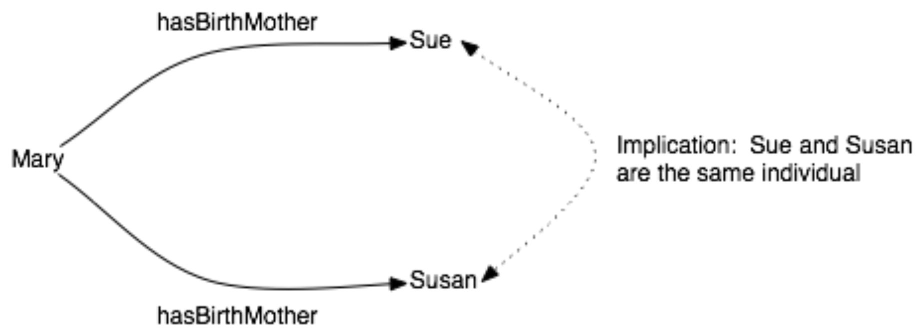
- Properties: Functional Properties



- There can only be one individual related to another individual via a functional property!

OWL

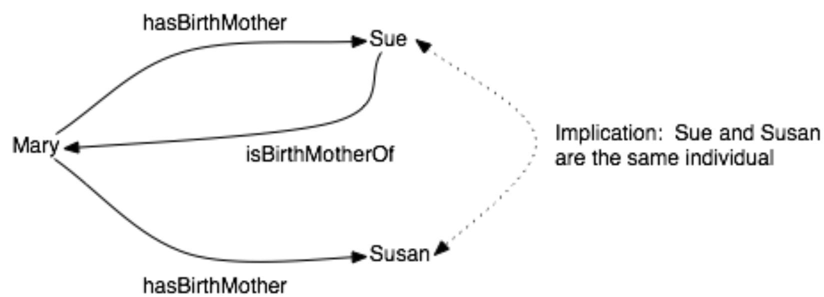
- Properties: Functional Properties



- No implication, if explicit assertion that Sue and Susan are different individuals
  - Instead: logical inconsistency

OWL

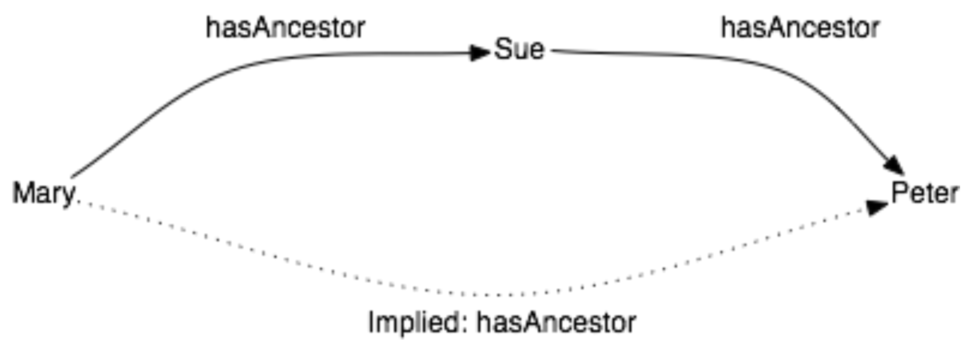
- Properties: Inverse Functional Properties



- *isBirthMotherOf* is an inverse property of *hasBirthMother*
- *hasBirthMother* is functional
- Thus: *isBirthMotherOf* is inverse functional

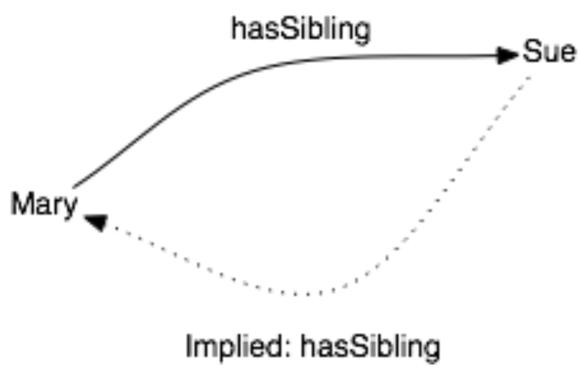
OWL

- Property: Transitive
- If a property X is transitive, then from A - X - B and B - X - C we can imply that A - X - C



OWL

- Property: Symmetric
  - If a property X is symmetric, then A - X - B implies also B - X - A

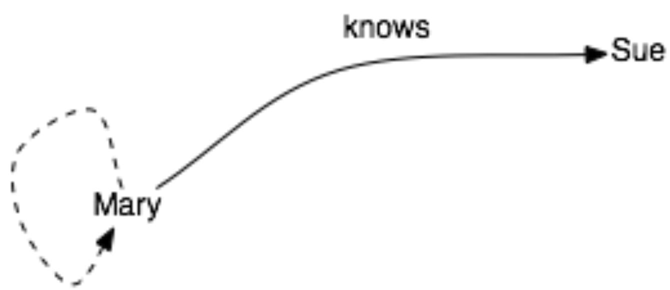


OWL

- Property: Asymmetric
  - E.g. *isChildOf*

OWL

- Property: Reflexive
  - E.g. *knows*: the related individuals can be the same



OWL

- Property: Irreflexive
  - Relates individual A to individual B, where A and B cannot be the same.

OWL

- Properties: Domain and Range may be specified
  - A property with a domain and a range links individuals from the domain (set of individuals) to individuals in the range (set of individuals)
  - These are not constraints to be checked

OWL

- Domain-Range example for properties:
  - For the Pizza ontology
    - Property *hasTopping* has the
      - domain: *Pizza*
      - range: *PizzaTopping*

OWL

- Domain-Range example for properties:
  - For the Pizza ontology
    - Property *isToppingOf* has the
      - domain: *PizzaTopping*
      - range: *Pizza*

OWL

- Domain and Range for inference:
    - If *hasTopping* has a Domain *Pizza*, and a Range *PizzaTopping*
        - individuals left of *hasTopping* are inferred to belong to the class of *Pizza*
        - individuals right of *hasTopping* are inferred to belong to the class *PizzaTopping*
    - even if these individuals have not been asserted to belong to the respective classes.

OWL

- Domain and Range of properties:
  - If a property has an inverse property, and if the domain and range are specified for it, Protégé automatically specifies the domain and range also for the inverse property

OWL

- Datatype properties:
  - e.g. an individual from the class *Person* has a property *hasAge* that is relating it to a data value of an integer type
  - e.g. a individual from the class *Pizza* has a property *hasPrice* that is relating it to a data value of a float type

OWL

- Restrictions in OWL
  - Properties describe binary relationships
  - Datatype properties describe relationships between individuals and data values
  - Object properties describe relations between two individuals

OWL

- Individual relations might be:
  - *Mark hasSibling Peter*
- The class of all individuals that have some *hasSibling* relationship
  - We describe a class of individuals via a relationship that these individuals participate in
- → Restrictions

OWL

- Restrictions
  - describe a class of individuals based on the relationships that members of the class participate in
  - it is a kind of class (like named classes are)
    - Restrictions describe ***anonymous*** classes

OWL Relations

- the class of individuals that have at least one *hasSibling* relationship
- the class of individuals that have at least one *hasSibling* relationship to members of *Man* – i.e. things that have at least one sibling that is a man
- the class of individuals that only have *hasSibling* relationships to individuals that are *Women* – i.e. things that only have siblings that are women (sisters)
- the class of individuals that have more than three *hasSibling* relationships
- the class of individuals that have at least one *hasTopping* relationship to individuals that are members of *MozzarellaTopping* – i.e. the class of things that have at least one kind of mozzarella topping
- the class of individuals that only have *hasTopping* relationships to members of *VegetableTopping* – i.e. the class of individuals that only have toppings that are vegetable toppings

OWL Restrictions

- Quantifier Restrictions
  - Existential Restrictions
  - Universal Restrictions
- Cardinality Restrictions
- hasValue Restrictions

OWL Restrictions

- Existential restrictions
  - e.g. the class of individuals that have at least one (or some) hasTopping relationship to members of MozzarellaTopping
- Universal restrictions
  - e.g. the class of individuals that only have relationships along this property to individuals of a specified class: the class of individuals that only have hasTopping relationships to members of VegetableTopping

OWL Restrictions

- Examples:
  - Make MargheritaPizza
  - Make AmericanPizza
    - by cloning MargheritaPizza:
      - select MargheritaPizza
      - in Menu select Edit and Duplicate Selected Class

OWL

- Make NamedPizza subclasses disjoint
  - Select for example MargheritaPizza
  - Menu > Edit > Make primitive siblings disjoint

OWL Classes

- Classes defined via necessary conditions
  - Primitive Classes
  - Example:
    - CheesyPizza: is necessarily a Pizza and has necessarily some CheeseTopping
  - For some random individual that satisfies these necessary conditions, e.g. that it is a type of Pizza and has some CheeseTopping we cannot a priori conclude that it is a CheesePizza.

Conditions

- Necessary but not sufficient:
  - Being a mammal is necessary but not sufficient for being a human.
- Sufficient but not necessary:
  - For a number q to be rational (P) is sufficient but not necessary to q‘s being a real number (Q) (since there are real numbers which are not rational).

OWL

- Explicit definition of necessary and sufficient conditions:
  - In Protégé 4:
    - Necessary conditions are Superclasses
    - Necessary and sufficient conditions are called Equivalent classes
- Classes with at least one set of necessary and sufficient conditions are called **Defined Classes**

Protégé

- To convert a class from Primitive class to Defined class, or necessary conditions to necessary and sufficient conditions
  - select the class
  - in Menu select Convert to defined class

OWL Universal Restriction

- Express that MargheritaPizza has MozzarellaTopping and TomatoTopping, and not other topping
  - Superclass restriction:
    - add hasTopping some MozzarellaTopping
    - add hasTopping some TomatoTopping
    - add hasTopping only (MozzarellaTopping or TomatoTopping)

OWL Universal Restriction

- It would seem to be enough to say only:
  - add hasTopping only (MozzarellaTopping or TomatoTopping)
- But:
  - this would logically include also Pizza without any topping at all

OWL

- Covering axioms:
  - Adding Spiciness to the Pizza Ontology
  - Create a class ValuePartition
  - Create a sub-class to it SpicinessValuePartition
  - Create the sub-classes Mild, Medium and Hot to it, and make them disjoint
  - Create a property hasSpiciness
  - Make this property functional
  - Set the range to SpicinessValuePartition

OWL

- Create an covering axiom
  - Select SpicinessValuePartition
  - Add to Equivalent classes:
    - Hot or Medium or Mild
    - Why? Set properties...

OWL

- Select JalapenoPepperTopping
- in Superclass section add hasSpiciness some Hot
- Add this to all toppings

OWL

- Create a SpicyPizza sub-class of Pizza
- In Superclass add hasTopping some (PizzaTopping and (hasSpiciness some Hot))
- Convert to defined class
- The reasoner should identify all pizzas that are spicy now as being of class SpicyPizza too

OWL

- Cardinality restriction
  - relationships determined by at least, at most, exactly number
- Example:
  - Create a sub-class InterestingPizza of Pizza
  - and then?????
    - where is the cardinality?
    - do we make this a defined class?

OWL

- How would we make a FourCheesePizza?

DBpedia

- A community project
- Public access to Linked Open Data Cloud.

- Provides a Knowledge Graph derived from Wikipedia
- Query Interface to Knowledge Graph using SPARQL query language (endpoint and web interface)

DBpedia

- Website:
  - http://dbpedia.org/
- SPARQL Query Editor:
  - https://dbpedia.org/sparql

DBpedia

- Knowledge in form of a collection of RDF Triples
  - subject→predicate→object model
  - Entity→Attribute→Value model
- Use of IRIs and URIs as identifiers
- Example
  - SELECT Query with triple-pattern in the Query Body (a subject, predicate, object)
  - Limiting Query Solution projection size to 10 records

DBpedia

- SPARQL

```
SELECT *
WHERE
   {
     ?s ?p ?o
   }
LIMIT 10
```

DBpedia

- URL encoded endpoint query
  - http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=SELECT+*%0D%0AWHERE%0D%0A+++++%7B%0D%0A+++++++++%3Fs+%3Fp+%3Fo%0D%0A+++++%7D%0D%0ALIMIT+10&format=text%2Fhtml&CXML_redir_for_subjs=121&CXML_redir_for_hrefs=&timeout=30000&debug=on&run=+Run+Query+

DBpedia

- Query with and without a deep knowledge of the underlying ontology.
- Building a query by searching on language-tagged literal values that are the objects of rdfs:label properties:

```
SELECT *
WHERE
  {
    ?actor rdfs:label "Clint Eastwood"@en
  }
```

DBpedia

- URL query:
  – http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=SELECT+*%0D%0AWHERE%0D%0A+++++%7B%0D%0A++++++++%3Factor+rdfs%3Alabel+%22Clint+Eastwood%22%40en%0D%0A+++++%7D&format=text%2Fhtml&CXML_redir_for_subjs=121&CXML_redir_for_hrefs=&timeout=30000&debug=on&run=+Run+Query+

DBpedia

- Language selection and multi-linguality
  - @en is a language tag for declaring the national language component of a literal value (i.e., English)
  - This feature makes both RDF and SPARQL multilingual in nature.

- Dereferencing
  - We can click on the resulting URI for Clint Eastwood in the “actor” column to view each relation associated with his specific URI.
  - This is known as dereferencing the DBpedia Identifier (an HTTP URI) that identifies the entity literally labeled as “Clint Eastwood”.

DBpedia

- Any property listed in the Property column can also be directly added to the query for further expansion.
- Use the dbo:birthPlace property and add it into our previous SPARQL Query. Additional statements can be quickly added to the ?actor variable using “;” :

```
SELECT *
WHERE
{
 ?actor  rdfs:label  "Clint Eastwood"@en ;
         dbo:birthPlace  ?birthPlace .
}
```

DBpedia

- Adding dbo:height property:

```
SELECT *
WHERE
{
  ?actor  rdfs:label  "Clint Eastwood"@en ;
          dbo:birthPlace  ?place ;
    dbo:height ?height .
}
```

DBpedia

- Requesting birth place as a string:

```
SELECT *
WHERE
{
  ?actor  rdfs:label  "Clint Eastwood"@en ;
          dbo:birthPlace  ?place ;
                dbo:height ?height .
 ?place a            dbo:City ;
          rdfs:label     ?cityName.
}
```

DBpedia

- Select specific variables for display:

```
SELECT ?actor
        ?cityName
WHERE
{
 ?actor  rdfs:label  "Clint Eastwood"@en ;
        dbo:birthPlace  ?place ;
    dbo:height ?height .
?place    a            dbo:City ;
        rdfs:label     ?cityName.
}
```

Use cases

- Claim triggers query and validation
  - X is 1.50 meters tall
  - X was born in Moscow
  - …

- Claim to abstract NLP representation
- Abstract NLP representation to SPARQL query
- Query result to comparison or validation

- Input: text output: SPARQL

Use cases

- Enriching entity annotation
- Mention of X
  - Annotation of details about X
- Query about entity X
- Aggregation of relevant information about X