Pizza Ontology

"a review of core concepts for building a pizza ontology"

presentation material based on:


presented by:

Atif Khan

http://www.infotrellis.com/
Outline

- Disclaimer
  - I am not an ontology engineer
- Goal
  - duration ~ 30 mins
  - review some basic ontology components
    - concepts, object properties, data properties, individuals
    - classification
  - introduce Protégé ontology editor
  - share my experience building the Pizza Ontology
Core Terminology

• Ontology

  “An ontology is a formal, explicit specification of a shared conceptualization”
  
  R. Studer (1998)

• Components

  - concepts define aggregation of things
  - individuals are instances of concepts
  - properties link concepts/individuals
Core Terminology

- Triples
  - a representation of ontological components
    - using the following notation
      \[\text{Subject verb Object}\]
  - example: “a pizza has a deep pan base”

\[
\text{Pizza hasBase DeepPanBase}
\]
Why Use Ontologies

• Precision of:
  – representation/expression
  – information sharing
  – knowledge inference

“Now! That should clear up a few things around here!”

Creating a Pizza Ontology

Protégé
Version 4.2.0 (Build 284)

OWL
Web Ontology Language
http://www.w3.org/TR/owl-features/
Define Core Concepts

- Identify core concepts
  - Pizza
  - Pizza Base
  - Pizza Toppings
example1.owl
Define Core Concepts

• Unique name assumption
  
  - need to explicitly define **sameness** & **uniqueness** using
  
  • Equivalent to
  
  • Disjoint with
1. Pick root class
   - Please select the root class
   - [Diagram]

2. Enter hierarchy
   - Please enter the hierarchy that you want to create. You should use tabs to indent names!
   - Prefix: [Input]
   - Suffix: [Input]
   - Pizza
   - PizzaBase
   - PizzaTopping
   - [Diagram]

3. Make sibling classes disjoint?
   - Do you want to make sibling classes disjoint? (Recommended)
   - [Check box: Make sibling classes disjoint]
   - [Diagram]
Concept Hierarchy

Disjoint Concepts
Define Properties

- Link concepts using properties
  - a pizza has a deep pan base
    \((\text{hasBase})\)
  - a pizza has a mozzarella cheese topping
    \((\text{hasCheeseTopping})\)
  - a pizza has a tomato and cheese topping
    \((\text{hasTomatoTopping})\) and \((\text{hasCheeseTopping})\)

- Property Hierarchy
  \[
  \text{hasBase} \Rightarrow \text{hasIngredient} \Leftarrow \text{hasTopping}
  \]
Define Inverse Properties

- Inverse property
  - each object property may have a corresponding inverse property

- “a pizza has a deep pan base”
  \[ \equiv \text{a deep pan is a base of a pizza} \]

\[ (isBaseOf) \text{ is inverse of } (hasBase) \]
\[ (hasBase) \text{ is inverse of } (isBaseOf) \]
“is_a” hierarchy

inverse property
Characteristics of Properties

• OWL *primitives* to enrich relationship definitions (see §4.6)
  - functional & inverse functional
  - transitive
  - symmetric & anti-symmetric
  - reflexive & irreflexive
Property Domains & Ranges

- Definition
  - properties link individuals from the **domain** to individuals from the **range**
Property Restrictions

• Restriction \(\approx\) Anonymous Class
  
  – a restriction is a class definition that groups individuals together based on one or more object properties

• Example
  
  – class of individuals that have at least one “hasTopping” relationship to individuals that are members of MozzarellaTopping
Existential Restrictions

• Intention
  – describe “some values from” restrictions

• Example
  – a pizza must have a pizza base
Existential Restrictions

Example:

```
hasBase some PizzaBase
```
Existential Restrictions

• Implication of "hasBase some PizzaBase"
  - if something is a Pizza then it is necessary for it to have a kind of PizzaBase

Figure 4.34: A Schematic Description of a Pizza — In order for something to be a Pizza it is necessary for it to have a (at least one) PizzaBase — A Pizza is a subclass of the things that have at least one PizzaBase

(M. Horridge, 2011)
Using the Reasoner (Classifier)

• Using a reasoner we can
  – determine class inconsistencies
    • e.g. inconsistent pizza
  – discovering implicit information
    • using necessary and sufficient conditions
    • e.g. cheesy pizza
Inconsistent Pizza Topping

Annotations: ProbeInconsistentTopping

- **comment**
  - This class should be inconsistent when the ontology is classified

Description: ProbeInconsistentTopping

- Equivalent To
  - CheeseTopping
  - VegetableTopping

To use the reasoner click Reasoner->Start reasoner

example4.owl
Inconsistent Pizza Topping

example4.owl
Using the Reasoner (Classifier)

• Using a reasoner we can
  – determine class inconsistencies
    • e.g. inconsistent pizza
  – discovering implicit information
    • using necessary and sufficient conditions
    • e.g. cheesy pizza
Cheesy Pizza

Explicit & Implicit definitions

- NamedPizza and its sub-classes are explicitly defined
- Discover sub-classes of CheesyPizza

example4.owl
Universal Restrictions

• Intention
  – describe “all and only values from” restrictions

• Example
  – a “vegetarian pizza” can only have cheese or vegetable toppings
Universal Restrictions

• Run the reasoner
  - expected behaviour:
    • Soho pizza and Margherita pizza should be classified as vegetarian pizzas
  - actual behaviour
    • reasoner does not find any vegetarian pizza subclasses
Open World Assumption

• OWA – What it means:
  – missing information is not confirmation of negation
  – in other words:
    • SohoPizza and MargheritaPizza toppings must be explicitly limited to their toppings

<table>
<thead>
<tr>
<th>SohoPizza:</th>
<th>MargheritaPizza:</th>
</tr>
</thead>
<tbody>
<tr>
<td>hasTopping only ( MozzarellaTopping or TomatoTopping or OliveTopping or ParmezanTopping )</td>
<td>hasTopping only ( MozzarellaTopping or TomatoTopping )</td>
</tr>
</tbody>
</table>
Universal Restrictions

Class hierarchy (inferred):
- Thing
  - Nothing
  - Pizza
    - CheesyPizza
    - NamedPizza
    - SpicyPizza
    - VegetarianPizza
      - MargheritaPizza
      - SohoPizza
  - PizzaBase
  - PizzaTopping
  - ValuePartition

Annotations:
- Annotations: MargheritaPizza
  - comment [type: string]
    - A pizza that only has Mozzarella and Tomato toppings

Description:
- Equivalent To
- SubClass Of
  - hasTopping only (MozzarellaTopping or TomatoTopping)
  - hasTopping some MozzarellaTopping
  - hasTopping some TomatoTopping
  - NamedPizza
    - CheesyPizza
    - VegetarianPizza

SubClass Of (Anonymous Ancestor)
Working with Protégé

● Protégé is simply an ontology IDE
  - editing
  - visualization
  - validation

● not required but extremely useful for
  - managing large ontologies
  - discovering existing ontologies