Ontologías 2: Representación estándar (formal y lógica) de las ontologías

Juan Luis Castro

Objetivo docente 1

- Comprender como se representa mediante un lenguaje formal estándar una ontología (RDF y RDF Schema):
  - Conocer el modelo Objeto-Atributo-Valor, o equivalentemente Sujeto-Predicado-Objeto (y su representación mediante el estándar RDF) para representar formalmente afirmaciones básicas.
  - Conocer la extensión de RDF a RDFS (RDF Schema) para representar las propiedades intrínsecas de una ontología (incluyendo formas estándares para representar “Clase”, “subclase”, “Individuo”, “Propiedad”, “Dominio”, “Rango”)

Objetivo docente 2

- Comprender como se representa a nivel lógico una ontología (lenguaje OWL):
  - Que conceptos derivados se pueden nombrar
  - Que relaciones se pueden utilizar en los axiomas
  - Como se puede expresar formalmente un axioma
  - Que preguntas (formales) se pueden plantear para que un razonador responda a partir del conocimiento representado en la ontología

Objetivo docente 3

- Comprender como funciona a nivel operativo una ontología, entendiendo como cada concepto y axioma se traduce en simples operaciones con las ternas que codifican el conocimiento

Recordatorio de sesión anterior 1/3

- Las ontologías introducen los conceptos y predicados básicos sobre un dominio:
  - Los conceptos se traducen en clases o individuos. Ejemplo: el concepto profesor, alumno o asignatura serian clases útiles para describir conocimiento sobre unos estudios, que tendrían profesores, alumnos o asignaturas concretos como individuos.
  - Los predicados se traducen en relaciones entre los individuos de una clase origen o dominio y los individuos de una clase destino o rango. Ejemplo: La relación es_profesor_de es un predicado que relaciona individuos de la clase profesor (dominio) con individuos de la clase asignatura (rango)

Recordatorio de sesión anterior 2/3

- Los conceptos pueden tener entre ellos una relación jerárquica de subclase. Ejemplo: asignatura_optativa sería una subclase de asignatura.
- En algunos predicados es destino o rango es un tipo de dato estándar (predicados de datos). Ejemplo: El valor del predicado nombre tomará como valores una cadena de caracteres.

Recordatorio de sesión anterior 3/3

- Las clase e individuos junto con sus propiedades de datos y su jerarquía tienen una estructura similar a un frame
- Una primera ampliación sobre los frames es que además hay propiedades que pueden tomar valor en clases definidas dentro de la ontología
- Otra gran ampliación es que las ontologías permiten incluir AXIOMAS, que son condiciones que deben verificarse y que permitirán integrar conocimiento a muy alto nivel de forma muy simple.

¿Como representar ontologías formalmente?

- Mediante unos Lenguajes (Estándares) para la definición de ontologías
  - RDF
  - RDF Schema
  - OWL

Estándares básicos

- UNICODE estándar que proporciona el medio por el cual codificar un texto en cualquier forma e idioma
  - IRI international resource identifier cadena caracteres que identifica inequívocamente un recurso (servicio, página, documento, etc.) físico o abstracto
    - identifica el recurso, pero no tiene por que localizar su ubicación(URL)

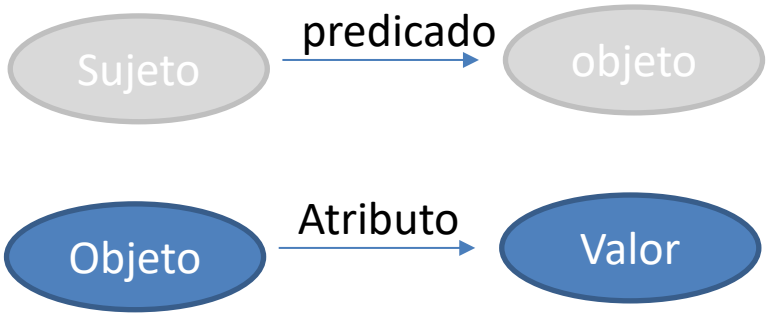
¡CADA OBJETO TENDRÁ UN IRI QUE LO IDENTIFIQUE!

  - XML meta-lenguaje extensible de etiquetas usado para el intercambio de datos en la web
    - uso de etiquetas con significado intuitivo para humanos, pero no para las máquinas
    - XML estandariza formato no significados
      - nombre de las etiquetas XML no ofrece semántica por si mismo

Representación básica basada en ternas

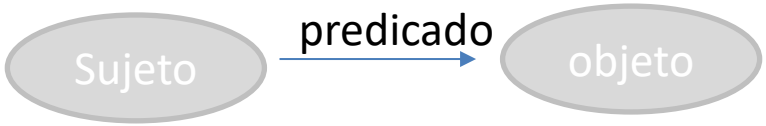
- Clásicamente, una afirmación básica podemos representarla en forma de terna:
  - Objeto-Atributo-Valor: Ingeniería del Conocimiento se imparte en tercero → (Ingenieria_del_Conocimiento, Curso, 3)
  - Análogamente, podemos hablar de la estructura Sujeto-Predicado-Objeto (Juan_Luis_Casto,Profesor,Ingenieria_del_Conocimiento)

  (son dos formas equivalentes, la idea es representar el conocimiento como una terna donde el segundo componente es el predicado que relaciona la primera y la tercera)



Representación de conceptos y relaciones (RDF)

- RDF (Resource Definition Format) estándar W3C para describir recursos (cualquier concepto que tenga una URI) en la web utilizando una representación basada en ternas:
  - Formato común para describir información que pueda ser leída y entendida por una aplicación informática.
  - Permite representar conceptos y relaciones mediante un conjunto de tripletas.
    - tripleta: describe propiedades de un recurso identificado por una IRI:
      - representa un documento o parte de él, o de una colección de documentos, un objeto, e
      - Propiedad: es siempre una IRI (predefinida y con significado preestablecido)
    - Cada tripleta combina: un recurso (Sujeto), una propiedad (Predicado) y un valor para la propiedad (Objeto)



Representando en formato rdf

Revisar la entrada de la Wikipedia sobre este estándar:
https://es.wikipedia.org/wiki/Resource_Description_Framework

Como veréis el formato es una terna por línea con el formato:

IRI IRI VALOR

Dónde

- el primer IRI es el identificador del sujeto,
- el segundo IRI es el identificador del predicado,
- El VALOR es libre, podría un dato de cierto tipo, un IRI,…

Uso de RDF (ejemplo simple)

Supongamos que tenemos ternas de la forma

IRI(ProfesorX) IRI(es_profesor_de) IRI(AsignaturaY)

Y de la forma

IRI(AsignaturaZ) IRI(pertenece_a) IRI(RamaV)

Ahora, encontrar los profesores de una Rama sería una simple operación con esas ternas:

1. Obtener el conjunto A formado por las primeras componentes de las ternas donde la segunda componente sea IRI(pertenece_a) y la tercera sea el IRI de la rama en cuestión
2. Obtener el conjunto respuesta como la primera componente de las ternas donde la segunda componente sea IRI(es_profesor_de) y la tercera componente pertenezca a A

Uso del estándar RDF

- RDF permite usar vocabularios semánticos definidos por expertos para describir recursos:
  - Dublin Core: descripción de recursos digitales (páginas HTML,libros, etc) [http://dublincore.org/]
  - FOAF (friendof a friend): ontología para descripción de personas[http://www.foaf-project.org/]
- Representable en forma de documentos XML [serialización RDF/XML]
- Posibilidad de usar lenguajes de consulta sobre tripletas RDF
  - SPARQL (http://www.w3.org/TR/rdf-sparql-query/): sintaxis tipo SQL sobre bases de datos de tripletas
  - RDF. DBPEDIA (http://wiki.dbpedia.org/): versión estructurada (tuplasRDF) de la Wikipedia.

RDF Schema (idea)

- Si usamos RDF para representar el conocimiento de una ontología, habrá predicados que siempre estarán:
  - Que un objeto X sea una clase
  - Que un objeto X sea una propiedad
  - Que un objeto X sea un individuo de una clase C
  - Que una clase C1 sea una subclase de otra clase C2
  - Que una propiedad P tenga como dominio una clase C
  - Que una propiedad P tenga como rango una clase C
  - Que una propiedad P1 sea subpropiedad de otra P2
- RDF Schema fija un nombre estándar para cada una de esos predicados primarios que estarán en cualquier ontología.
- De esta forma, una máquina comprenderá el conocimiento representado en esa ontología, pues la semántica de esos predicados ya están definidos por RDF Schema

RDF Schema

- RDFS (RDF Schema) lenguaje extensible que proporciona los elementos básicos para crear ontologías (vocabularios semánticos RDF).
- Permite definir clases, relaciones entre clases, restricciones sobre propiedades, etc. Utilizando los siguientes IRIs de predicados prefijados:
  - rdfs:Class declarar recursos como clases para otros recursos
  - rdfs:subClassOf definir jerarquías (relaciona clase con superclases)
  - rdfs:property definir subconjunto de recursos RDF que son propiedades
  - rdfs:subPropertyOf definir jerarquías de propiedades
  - rdfs:domain dominio de una propiedad (clase de recursos que aparecen como sujetos en las tripletas de ese predicado)
  - rdfs:range rango de una propiedad (clase de recursos que aparecen como objetos en las tripletas de ese predicado)
  - rdfs:individual para declarar un individuo de una clase
- RDF Schema define el significado de los términos usados en las tripletas RDF

Ejemplo

- Queremos introducir el concepto profesor, alumno, asignatura, y asignatura_optativa como subclase de asignatura

  IRI(profesor) rdfs:Class ““
  IRI(alumno) rdfs:Class ““
  IRI(asignatura) rdfs:Class ““
  IRI(asignatura_optativa) rdfs:subClassOf IRI(asignatura)

(fijaros que la tercera componente de las tres primeras líneas se deja vacia y lo he marcado con ““ para indicar que es la cadena vacia)

Ejemplo

- Queremos introducir la propiedad es_profesor_de que es un predicado que relaciona individuos de la clase profesor con individuos de la clase asignatura

IRI(es_profesor_de) rdfs:property “”

IRI(es_profesor_de) rdfs:domain IRI(profesor)

IRI(es_profesor_de) rdfs:range IRI(asignatura)

Ejemplo

- Ahora queremos introducir que JLC es profesor de la asignatura IC

  IRI(JLC) IRI(es_profesor_de) IRI(IC)

Fijaros que ahora el sistema, con lo introducido podría deducir por consistencia de la ontología:

IRI(JLC) rdfs:individual IRI(profesor)

IRI(IC) rdfs:individual IRI(asignatura)

Comentario

- No es usual diseñar una ontología escribiendo directamente las ternas en formato rdfs.
- Normalmente se utiliza la herramienta desarrollada en la Universidad de Stanford Protegè, que permite hacerlo mediante una herramienta gráfica :

  https://protege.stanford.edu

Introduciendo conceptos, propiedades y relaciones no básicas

- RDFS nos sirve para declarar las cosas básicas de las ontologías, y para hacer unas deducciones elementales como hemos visto
- Pero todavía nos falta ver como podemos introducir los axiomas, esas condiciones que pueden ser muy complejas.
- Para ello utilizaremos el lenguaje de la lógica descriptiva OWL.

OWL

- OWL (ontology web language) extiende RDFS para permitir la expresión de relaciones complejas entre clases RDFS, y mayor precisión en las restricciones de clases y de propiedades.
  - Derivado de la fusión de los lenguajes de ontologías DAML y OIL.
  - Permite:
    - expresar clases complejas
    - expresar relaciones entre clases
    - expresar y restringir clases (rango, dominio)
    - expresar y restringir propiedades (cardinalidad)
    - Hacer afirmaciones complejas

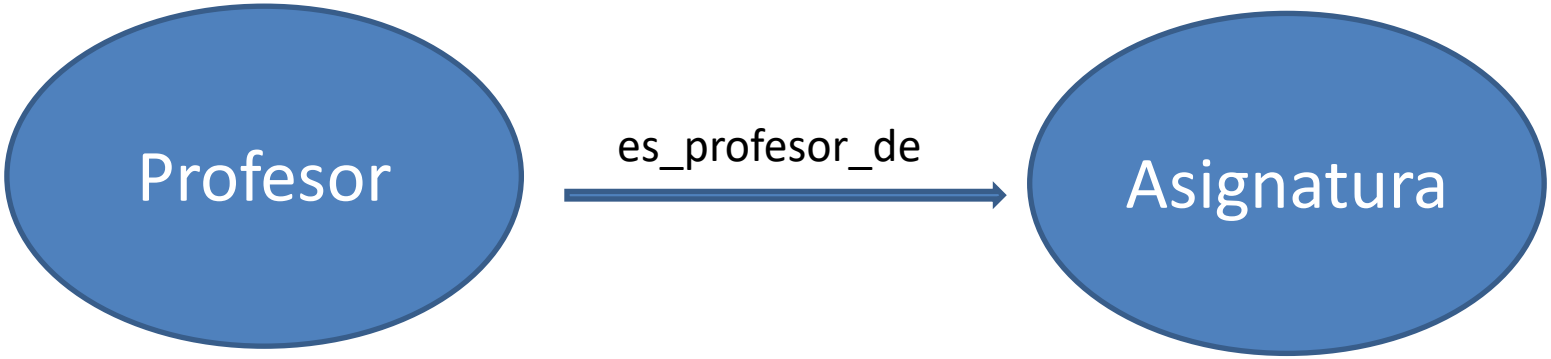
OWL

- Tres variantes/sublenguajes(menor a mayor potencia expresiva)
  - OWL-lite: versión simplificada (representación de jerarquías simples)
  - OWL-DL: incluye constructores tomados de Description Logics(DL) -Busca compromiso entre máxima expresividad y eficiencia computacional (sólo constructores decidibles de DL)
  - OWL-full: soporte completo de constructores DL

Dominio y rango de las propiedades

- Las propiedades ligan individuos de un dominio a individuos de un rango

Dominio —Propiedad→ Rango



Constructores OWL
(permiten nombrar clases no simples)

itersectionof:	$C_1 \cap \ldots \cap C_n$	Doctor ∩ Mujer
unionof:	$C_1 \cup \ldots \cup C_n$	Doctor U Abogado
complementof:	¬ C	¬ Hombre
oneof	$\{ x_1 \} \cup \ldots \cup \{ x_n \}$	{Juan} U {Maria}
AllValuesFrom	∀P.C	∀es_profesor_de.CSI
SomeValuesFrom	∃P.C	∃matriculado.CSI
maxCardinality	≤n.P	≤1.tieneHijo
minCardinality	≥n.P	≥2.tieneHijo

∀es_profesor_de.CSI → "la clase de los profesores que tienen toda la docencia en la rama CSI, o no tienen docencia"

∃matriculado.CSI → "la clase de los estudiantes matriculados en alguna asignatura de CSI"

≤1.tieneHijo → "la clase de las personas que tienen 1 hijo o menos"

¡Siempre se restringe a la clase dominio de la propiedad!

Propiedades derivadas

- Propiedad inversa

  inverseof P^{-1} $es\_profesor\_de^{-1}$

  sería la propiedad es_impartida_por

  $\forall es\_profesor\_de^{-1}$.Decsai → "clase de las asignaturas con todos los profesores de DECSAI"

- Metapropiedad Cardinal

  numberoff |C| |profesor|

Relaciones y Axiomas OWL

subClassof $C_1 \subseteq C_2$ Humanos $\subseteq$ Animales $\cap$ Bipedos

EquivalentClass $C_1 \equiv C_2$ Doctor $\equiv$ Tesis.{Si}

DisjointWith $C_1 \subseteq \neg C_2$ Matriculado.{TFG} $\subseteq \neg \exists$Matriculado.tercero

SameIndividualAs $\{x_1\} \equiv \{x_2\}$ {JLC} = {JuanLuisCastro}

DifferentFront $\{x_1\} \subseteq \neg \{x_2\}$ {IC} $\subseteq \neg$ {TSI}

subPropertyOf $P_1 \subseteq P_2$ tiene_hija $\subseteq$ tiene_hijos

equivalentProperty $P_1 \equiv P_2$ coste $\equiv$ precio

inverseof $P_1 \equiv P_2^{-1}$ padre_de $\equiv$ hijo_de $^{-1}$

transitiveProperty $P^+ \subseteq P$ antecesor $^+ \subseteq$ antecesor

functionalProperty $T \leq 1.P$ $T \leq 1$.tieneMadre

inversefunctionalProperty $T \leq 1.P^{-1}$ $T \leq 1$.tieneDNI^{-1}

Propiedades simétricas

- P es simétrica: $P = P^{-}$

es_hermano_de es simétrica es_hermano_de = $es\_hermano\_de^{-1}$

porque si A es_hermano_de B entonces B es_hermano de A

Si declaramos una propiedad P como simétrica, para cada terna

S P O

el sistema añadirá la terna

O P S

Propiedades inversas: $P_1 \equiv P_2^{-1}$

es_padre_de inversa es_hijo_de es_padre_de $\equiv$ es_hijo_de^{-1}

porque si A es_padre_de B entonces B es_hijo_de A

Si declaramos una propiedad P1 y P2 como inversas, para cada terna

S P1 O

el sistema añadirá (deducirá)

O P2 S

Propiedades transitivas: P^+ C P

es_ancestro_de es transitiva

es_ancestro_de$^+$ C es_ancestro_de

porque si A es_ancestro_de B y B es_ancestro_de C, entonces A es_ancestro_de C

Si declaramos una propiedad P como transitiva, para cada par de ternas

S P O

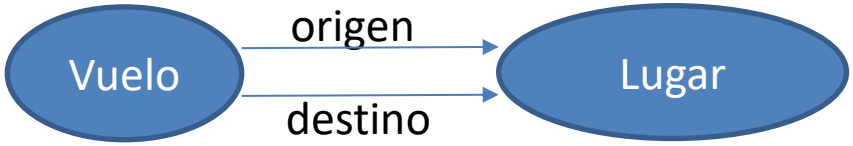
O P O1

el sistema añadirá (deducirá) la terna

S P O1

Propiedades funcionales: T ≤ 1.P

Cada individuo del dominio esta relacionado por esa relación con uno y solo uno del rango



Origen y destino son funcionales

Porque cada vuelo tiene uno y solo un origen, y uno y solo un destino

Si declaramos una propiedad como funcional,

- si un individuo del dominio esta relacionado mediante esa propiedad con dos elementos, el sistema deducirá que son el mismo.
- Si un individuo del dominio no esta relacionado mediante esa propiedad con ningún elemento, el sistema deducirá que hay incompletitud

Propiedades funcionales inversas: $T \leq 1.P^{-1}$

es_madre_biológica_de es funcional inversa $T \leq 1.es\_madre\_biológica\_de^{-1}$

porque cada persona tiene un y solo una madre biológica

Si declaramos una propiedad como funcional,

- si un individuo del rango esta relacionado mediante esa propiedad con dos elementos del dominio, el sistema deducirá que son la misma
- Si un individuo del rango no esta relacionado mediante esa propiedad con ningún elemento del dominio, el sistema deducirá que hay incompletitud

Ejemplos

Representar el axioma: todos los profesores que dan clase de teoría en CSI son catedráticos. Teniendo en la ontología:

Clases: - profesor, con subclase catedrático

- asignatura, con subclase CSI

Propiedades: - es_profesor_de, con subpropiedad es_profesor_de_teoría_de; Dominio=profesor y Rango=asignatura

Axioma:

$\exists$ es_profesor_de_teoría_de.CSI $\sqsubseteq$ catedrático

Ejercicio 1

Con base la ontología del congreso de las transparencias Ontologías 1, traducir a palabras el siguiente axioma :

$$\exists \text{NºArticulos}.\{n\} \equiv \leq n.\text{P-author}^{-1} \cap \geq n.\text{P-author}^{-1}$$

descomponiendo cada parte simple de la expresión

Ejercicio 2

- Crear un axioma para representar en la ontología del congreso de las transparencias Ontologías 1 la siguiente afirmación:

“Todas las presentaciones deben tener al menos uno de sus autores inscritos”

Comentario

Se han desarrollado motores que implementan de forma automática el razonamiento sobre el conocimiento de una Ontología:

- Deducen y rellenan los componentes que se pueden deducir según los axiomas indicados
- Dada un afirmación expresada en OWL, nos indica si ese axioma se verifica en la ontología

Ejemplos de Razonadores

- FaCT++

  http://owl.man.ac.uk/factplusplus/
- Pellet

  http://clarkparsia.com/pellet/download
- Racer

  http://www.racer-systems.com/products/download/index.phtml

Razonamiento con ontologías

- Clasificación automática: dado un concepto expresado en OWL, chequea si está incluido en algunas de las clases de la ontología
- Clasificación de instancias: Dada una nueva instancia, deduce si pertenece a alguna de las clases de la ontología
- Detección de redundancia: detecta si dos individuos, o dos clases son la misma
- Chequeo de consistencia
  - Disjoint: detecta si hay elementos comunes en clases que deben ser distintas
  - Restricciones: detecta si algún axioma no se verifica

Conclusiones

- Las ontologías…
  - Definen vocabulario común.
  - Crean entendimiento compartido.
  - Proveen acceso común al conocimiento.
  - Permiten la extracción de nuevo conocimiento implícito a través de razonamiento automático.
  - Permiten compartir, integrar y re-utilizar conocimiento.
  - Proveen conocimiento entendible por humanos y computadoras.

Conclusiones

Para representar conocimiento y razonar con ontologías:

1. Seleccionamos los conceptos y propiedades básicos para el dominio y el problema
2. Establecemos la jerarquías de conceptos y definimos los axiomas relativos a cada una de las propiedades (¿funcional?¿inversa funcional?¿Simétrica?¿Transitiva?)
3. Añadimos los axiomas mas complejos expresados en el lenguaje OWL
4. Rellenamos los individuos de la ontología
5. Chequeamos la consistencia con el razonador y dejamos que deduzca y relleno todo lo que pueda
6. Planteamos las tarea a desarrollar en forma de consulta estándar del lenguaje OWL (clasificar, verificar una afirmación, etc…)