Representación del Conocimiento y Web Semántica

SI (Sistemas Inteligentes) 2014/15

Integrantes:

- Gabriela Guadalupe Martínez Puente
- Karen Elizabeth Torres Delgado
- Angel Santiago Jaime Zavala

Profesor: Victor Manuel Darriba Bilbao

Objetivo

Realizar un pequeño ejemplo que explote las posibilidades de la web semántica en Prolog.

Introducción

De acuerdo a la W3C, la definición de Web Semántica y de RDF, consisten de lo siguiente:

Web Semántica: La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante.

RDF (Resource Description Framework o Marco de Descripción de Recursos): RDF is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed.

RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a “triple”). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications.

This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations.

Desarrollo

1.- Se deben de importar las librerías necesarias para la práctica por medio de cargas y tuplas RDF, así como ajustar los prefijos que se utilizarán para las etiquetas del código HTML por medio de RDF:

```
/* (1.1) Carga de la librería */

:- use_module(library(semweb/rdf_db)).

/* (1.2) Carga de tuplas RDF */
:- rdf_load('/jamendo-rdf/jamendo.rdf').

/* (2.1) Habilitar uso de prefijos en URIs */
:- use_module(library(semweb/rdf_portray)).

/* (2.2) Definir los prefijos de las tuplas del ejemplo de Jamendo*/
:- rdf_register_prefix(dc,'http://purl.org/dc/elements/1.1/').
:- rdf_register_prefix(event,'http://purl.org/NET/c4dm/event.owl#').
:- rdf_register_prefix(foaf,'http://xmlns.com/foaf/0.1/').
:- rdf_register_prefix(mo, 'http://purl.org/ontology/mo/').
:- rdf_register_prefix(tags, 'http://www.holygoat.co.uk/owl/redwood/0.1/tags/').
:- rdf_register_prefix(time, 'http://www.w3.org/2006/time#').
:- rdf_register_prefix(tl, 'http://purl.org/NET/c4dm/timeline.owl#').

/* Librería y prefijos de GeoNames */
:- use_module(library(semweb/rdf_http_plugin)).
:- rdf_register_prefix(gn, 'http://www.geonames.org/ontology#').
```

2.- Para probar los predicados que más adelante se utilizarán, se realizarán pruebas con los siguientes artistas:

- Zumbido (España)
- Loon Attic (España)
- Mamut (España)
- Patient 99 (USA)
- Breeding Sonic Waves (Portugal)
- Ekat (Francia)

3.- El siguiente paso es crear un predicado por medio del cual podamos acceder a la URI del artista, tan solo ingresando el nombre de este. Esto es ideal ya que para predicados que se construyan más adelante, será más claro utilizar el nombre del artista en vez de su URI:

```
artista(Artista,ArtistaURI) :-
    rdf(ArtistaURI,foaf:name,literal(type(xsd:string,Artista))).
```

```
1 ?- artista('Patient 99',ArtistaURI).
ArtistaURI = 'http://dbtune.org/jamendo/artist/7502'.
```

4.- Se construye un predicado por medio del cual podamos obtener todos los discos que un artista posee, guardándolos en una lista:

```
discos(Artista,ListaDiscos) :-
    artista(Artista,ArtistaURI),
    findall(Disco,rdf(ArtistaURI,foaf:made,Disco),ListaDiscos).
```

```
2 ?- discos('Patient 99',ListaDiscos).
ListaDiscos = ['http://dbtune.org/jamendo/record/6597', 'http://dbtune.org/jamendo/record/7061'].
```

5.- Se construye un predicado para obtener todas las etiquetas de un disco de algún artista, y guardarlas en una lista:

```
etiquetas(DiscoURI,ListaEtiquetas) :-
    findall(Etiqueta,rdf(DiscoURI,tags:taggedWithTag,Etiqueta),ListaEtiquetas).
```

```
4 ?- etiquetas('http://dbtune.org/jamendo/record/7061',ListaEtiquetas).
ListaEtiquetas = ['http://dbtune.org/jamendo/tag/electro'].
```

6.- El siguiente predicado consiste en obtener todas las etiquetas de un artista, utilizando la unión de conjuntos, para así al final obtener una lista de todas las etiquetas únicas de un artista. Se utiliza la definición recursiva con un caso base, para de esa manera recorrer la lista de discos:

```
etiquetas_discos([],[]).
etiquetas_discos([Car|Cdr],ListaEtiquetas) :-
    etiquetas(Car,EtiquetasCar),
    etiquetas_discos(Cdr,EtiquetasCdr),
    union(EtiquetasCar,EtiquetasCdr,ListaEtiquetas).
```

```
5 ?- etiquetas_discos( ['http://dbtune.org/jamendo/record/6597', 'http://dbtune.org/jamendo/record/7061'], ListaEtiquetas).
ListaEtiquetas = ['http://dbtune.org/jamendo/tag/electro'].
```

7.- Se construye un predicado con el cual se obtienen todas las etiquetas de todos los discos de un artista, y se guardan en una lista final, sin repetirse, y catalogando así a algún artista a estas etiquetas:

```
etiquetas_artista(Artista,ListaEtiquetas) :-
    discos(Artista,ListaDiscos),
    etiquetas_discos(ListaDiscos,ListaEtiquetas).
```

```
6 ?- etiquetas_artista('Zumbido',ListaEtiquetas).
ListaEtiquetas = ['http://dbtune.org/jamendo/tag/experimental', 'http://dbtune.org/jamendo/tag/galego',
do/tag/punkrock', 'http://dbtune.org/jamendo/tag/rock'].
```

8.- El siguiente predicado es para llevar a cabo el cálculo del coeficiente de Jaccard, que es un estadístico que mide la similitud, disimilitud o distancias que existen entre dos estaciones de muestreo. Este predicado ya está facilitado por el profesor, por medio de un ejercicio visto en clase:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

```
coeficiente_jaccard(Lista1,Lista2,Valor) :-
    intersection(Lista1,Lista2,Interseccion),
    length(Interseccion,A),
    union(Lista1,Lista2,Union),
    length(Union,B),
    Valor is float(A) / float(B).
```

```
11 ?- coeficiente_jaccard(['http://dbtune.org/jamendo/tag/acid', 'http://dbtune.org/jamend
/jamendo/tag/techno', 'http://dbtune.org/jamendo/tag/tekno', 'http://dbtune.org/jamendo/ta
Valor = 0.16666666666666666.
```

9.- Se construye también un predicado para calcular el coeficiente de Dice, también conocido por otros nombres tales como el índice de Sørensen, coeficiente de Dice, y que es un estadístico utilizado para comparar la similitud de dos muestras:

$$QS = \frac{2C}{A+B} = \frac{2|A \cap B|}{|A| + |B|}$$

```
coeficiente_dice(Lista1,Lista2,Valor) :-
    intersection(Lista1,Lista2,Interseccion),
    length(Interseccion,A),
    length(Lista1,S1),
    length(Lista2,S2),
    Valor is float(2.0 * A) / float(S1 + S2).
```

```
12 ?- coeficiente_dice(['http://dbtune.org/jamendo/tag/acid', 'http://dbtune.org/jamendo/tag/
mendo/tag/techno', 'http://dbtune.org/jamendo/tag/tekno', 'http://dbtune.org/jamendo/tag/trar
Valor = 0.2857142857142857.
```

10.- El siguiente predicado tiene como intención el elegir entre que coeficiente utilizar a la hora de que querer obtener la similitud entre dos muestras:

```
usar_coeficiente(Coeficiente,Lista1,Lista2,Valor) :-
    (Coeficiente == 'Jaccard' -> coeficiente_jaccard(Lista1,Lista2,Valor); coeficiente_dice(Lista1,Lista2,Valor)).
```

```
13 ?- usar_coeficiente('Jaccard', ['http://dbtune.org/jamendo/tag/acid', 'http:/
tune.org/jamendo/tag/techno', 'http://dbtune.org/jamendo/tag/tekno', 'http://dbt
Valor = 0.16666666666666666.
```

11.- El siguiente predicado es uno de los que son requeridos en la práctica, por medio de este predicado se puede determinar la similitud de dos artistas por medio de la comparación de las etiquetas de ambos artistas con el uso del coeficiente Jaccard o Dice.

```
artista_similar(Artista1,Artista2,Coeficiente,Similitud) :-
    etiquetas_artista(Artista1,ListaEtiquetas1),
    etiquetas_artista(Artista2,ListaEtiquetas2),
    usar_coeficiente(Coeficiente,ListaEtiquetas1,ListaEtiquetas2,Similitud).
```

```
14 ?- artista_similar('Zumbido','Breeding Sonic Waves','Jaccard',Similitud).
Similitud = 0.14285714285714285.
```

12.- Este es el segundo predicado principal que se pide en la práctica, por medio de este podemos obtener una lista de artistas recomendados, tales que cuenten con un nivel mayor o igual que el del umbral proporcionado, el cual se obtiene por medio de la similitud obtenida entre el artista proporcionado como parámetro, y los artistas que se están consultando a la hora de calcular el índice con alguno de los dos coeficientes:

```
recomendar(Artista,Coeficiente,Umbral,ListaArtistas) :-
    findall(Art,
    (artista(Art,ArtistaURI),
    Art \== Artista,
    artista_similar(Artista,Art,Coeficiente,Similitud),
    Similitud >= Umbral),
    ListaArtistas).
```

```
15 ?- recomendar('Zumbido','Jaccard',0.5,ListaArtistas).
ListaArtistas = ['Illusionary World', 'the keko jones band', 'Stoke', 'VOLDENBERG',
```

13.- A partir de ahora se comenzará con la explicación de los predicados necesarios para crear una recomendación de artistas por país y por países vecinos, utilizando las relaciones proporcionadas por geoNames. El siguiente predicado obtiene y carga el país al que pertenece algún artista que tenga como ubicación una provincia o región dentro de ese país, utilizando la relación "gn:parentCountry":

```
paisParent(Artista,Pais) :-
    artista(Artista,ArtistaURI),
    rdf(ArtistaURI,foaf:based_near,GeonameURL),
    concat(GeonameURL,'about.rdf',P1),
    rdf_load(P1),
    rdf(GeonameURL,gn:parentCountry,Pais),
    concat(Pais,'about.rdf',P2),
    rdf_load(P2).
```

14.- El siguiente predicado obtiene y carga el país de un artista, en caso de que este tenga como ubicación directa a su país directamente:

```
pais(Artista,Pais) :-
    artista(Artista,ArtistaURI),
    rdf(ArtistaURI,foaf:based_near,Pais),
    concat(Pais,'about.rdf',P),
    rdf_load(P).
```

15.- Este predicado engloba los dos predicados vistos anteriormente, para determinar el país definitivo de un artista, ya sea que el artista pertenezca a una región o provincia del país, o al país directamente, con este predicado se obtiene la URI absoluta del país al que pertenece tal artista:

```
pais_artista(Artista,Pais) :-
    paisParent(Artista,Pais);
    pais(Artista,Pais).
```

```
20 ?- pais_artista('Loon Attic', Pais).
% Parsed "http://sws.geonames.org/2514254/about.rdf" in 0.02 sec; 38 triples
% Parsed "http://sws.geonames.org/2510769/about.rdf" in 0.02 sec; 216 triples
Pais = 'http://sws.geonames.org/2510769/' .
```

16.- Con este predicado determinamos si dos artistas son de un mismo país o no:

```
mismo_pais(Artista1,Artista2) :-
    pais_artista(Artista1,Pais1),
    pais_artista(Artista2,Pais2),
    Pais1 == Pais2.
```

```
21 ?- mismo_pais('Zumbido','Loon Attic').
% Parsed "http://sws.geonames.org/2510769/about.rdf" in 0.00 sec; 216 triples
% Parsed "http://sws.geonames.org/2510769/about.rdf" in 0.00 sec; 216 triples
% Parsed "http://sws.geonames.org/2514254/about.rdf" in 0.00 sec; 38 triples
% Parsed "http://sws.geonames.org/2510769/about.rdf" in 0.02 sec; 216 triples
true .
```

17.- Con este predicado obtenemos la lista de los países vecinos al país al que pertenece el artista pasado como parámetro:

```
paises_vecinos(Artista,ListaPaises) :-
    pais_artista(Artista,Pais),
    rdf(Pais,gn:neighbouringFeatures,PaisesVecinos),
    rdf_load(PaisesVecinos),
    findall(PaisVecino,rdf(PaisVecino,gn:neighbour,Pais),ListaPaises).
```

```
22 ?- paises_vecinos('Zumbido',ListaPaises).
% Parsed "http://sws.geonames.org/2510769/about.rdf" in 0.02 sec; 216 triples
% Parsed "http://sws.geonames.org/2510769/about.rdf" in 0.00 sec; 216 triples
% Parsed "http://sws.geonames.org/2510769/neighbours.rdf" in 0.02 sec; 20 triples
ListaPaises = ['http://sws.geonames.org/3017382/', 'http://sws.geonames.org/2264397/', 'h
/sws.geonames.org/2411586/'].
```

18.- Este es el último predicado, con el cual creamos una lista de artistas recomendados debido a que son del mismo país o de países vecinos que el artista dado, calculando está similitud por medio del uso de los coeficiente Dice o Jaccard, de la misma manera que el predicado “artista_similar” en el punto 11:

```
recomendar_pais(Artista,Coeficiente,Umbral,ListaArtistas) :-
    paises_vecinos(Artista,ListaPaises),
    findall(Art,
    (artista(Art,ArtistaURI),
    Art \== Artista,
    artista_similar(Artista,Art,Coeficiente,Similitud),
    Similitud >= Umbral,
    (mismo_pais(Artista,Art); (pais_artista(Art,P), member(P,ListaPaises))))
    ,ListaArtistas).
```

```
24 ?- recomendar_pais('Zumbido','Dice',0.5,ListaArtistas).

% Parsed "http://sws.geonames.org/3932488/about.rdf" in 0.02 sec; 167 triples
% Parsed "http://sws.geonames.org/3932488/about.rdf" in 0.02 sec; 167 triples
% Parsed "http://sws.geonames.org/3932488/about.rdf" in 0.02 sec; 167 triples
% Parsed "http://sws.geonames.org/3932488/about.rdf" in 0.02 sec; 167 triples
% Parsed "http://sws.geonames.org/2510769/about.rdf" in 0.02 sec; 216 triples
% Parsed "http://sws.geonames.org/2510769/about.rdf" in 0.03 sec; 216 triples
% Parsed "http://sws.geonames.org/2975249/about.rdf" in 0.00 sec; 31 triples
% Parsed "http://sws.geonames.org/3017382/about.rdf" in 0.02 sec; 208 triples
% Parsed "http://sws.geonames.org/2975249/about.rdf" in 0.02 sec; 31 triples
% Parsed "http://sws.geonames.org/2975249/about.rdf" in 0.00 sec; 31 triples
% Parsed "http://sws.geonames.org/3017382/about.rdf" in 0.03 sec; 208 triples
% Parsed "http://sws.geonames.org/2975249/about.rdf" in 0.00 sec; 31 triples
ListaArtistas = ['Alquimia', 'the keko jones band', 'Stoke', 'Lithium', 'Curly\'s Revenge', 'Galère', 'DELITO Y MEDIO',
].
```

Conclusiones, Limitaciones, Posibles Problemas…

El manejo de la Web Semántica por medio de Prolog es una opción muy versátil a la hora de querer desarrollar programas que cuenten con datos bien organizados y coherentes en el ambiente de la Web. El poder manejar las etiquetas por medio de una relación y referencia única es una muestra clara de lo bien organizada que se encuentra la web, y de cómo está nos provee de varias herramientas dentro del entorno de la Web Semántica.

Una de las posibles limitaciones que veo, es el hecho de contar con grandes cantidades de datos, que, aunque a veces estos están bien organizados, son muy complicados de manipular, sobre todo si se están haciendo grandes consultas de datos. De la misma manera, la limitación más visible sale a la luz en el uso de geoNames, ya que debemos de contar con conexión a Internet para poder manejar de manera óptima todas las consultas a geoNames, aparte de que esta misma nos provee de tiempo limitado para el uso de sus servicios.

Los posibles problemas de rendimiento se presentan principalmente en el área de consultas rdf con geoNames, ya que como los datos se encuentran en Internet, se tienen que cargar constantemente los datos sobre algún país, y esto toma tiempo en cada predicado que se llega a ejecutar, perdiendo algo de rendimiento.

Y como punto final, las mejoras que se podrían hacer en el uso de los predicados que se realizaron para la aplicación podría ser en el aspecto de definiciones y recorridos recursivos para las listas, al estilo de la programación lógica, optimizando de suma manera a diferencia de los recorridos iterativos. De igual manera, evitar el uso de variables “singleton”, es una muy buena opción para mejores prácticas de código.