Adaptación de IMDb usando RDFlib y SparQL

Jesús A. Torrejón León
Universidad Nacional de Ingeniería
Lima, Perú
jtorrejonl@uni.pe

Jordi J. Bardales Rojas
Universidad Nacional de Ingeniería
Lima, Perú
jbardalesr@uni.pe

Walter J. Felipe Tolentino
Universidad Nacional de Ingeniería
Lima, Perú
wfelipet@uni.pe

Julio C. Yaranga Santé
Universidad Nacional de Ingeniería
Lima, Perú
julio.yaranga.s@uni.pe

***Resumen*—En este trabajo realizaremos una adaptación de IMDB a una estructura RDF/N3 para consultas y visualización, utilizando las librerías RDFlib para utilizar las funciones RDF; IMDB para extraer la información de películas, actores, directores,etc., mediante URI's; Graphviz para la visualización de los grafos y SparQL para consultas más avanzadas como listas las películas mejor valoradas de acuerdo al índicador IMDb Rating. Partiendo del sentido primitivo de la semántica del lenguaje natural que realizaremos una estrategia para vincular el sentido de las palabras con un modelamiento de triples, estructurándola para el modelamiento RDF.**

Índice de Términos— RDF, RDFlib Python, Web Semántica, [término del tema a estudiar].

I. Introducción

La semántica de un lenguaje de programación, suele referirse al mapeo de la sintaxis del lenguaje a algún formalismo que expresa el significado de ese lenguaje. Cuando nos referimos a la semántica del lenguaje natural a menudo nos referimos a comprender el significado detrás de los enunciados, y en el sentido más primitivo de esta noción semántica es el vínculo de un término en un enunciado con la entidad en el mundo a la que se refiere el término[1]. Dado que los símbolos pueden referirse a cosas en el mundo surge la pregunta ¿Cómo podemos construir modelos a partir de éstos símbolos que nos ayuden a capturar, comprender y comunicar lo que sabemos sobre las relaciones entre estas cosas? Para dicha solución adoptaremos la estrategia RDF a una base de datos de IMDB donde extraeremos información sobre películas, directores, actores, etc.

II. Estructura RDF

En este trabajo modelamos un sistema de reseñas de Películas para ello utilizaremos el formato N3 el cual es un superset de RDF que nos permitirá almacenar la estructura en archivos. Por ejemplo para una determinada película tenemos:

[1]vinculos de terminos

```
<https://www.imdb.com/title/tt9208876/> a imdb:Movie ;
    dc:title "The Falcon and the Winter Soldier" ;
    rev:hasReview <file:///home/turing/movies.
    n3#N749c0863a3084db59528198622f71997> ;
    imdb:cast "Anthony Mackie",
        "Sebastian Stan" ;
    imdb:director "Kari Skogland" ;
    imdb:year 2021
```

Que representa una serie de triplets como por ejemplo https://www.imdb.com/title/tt2084970/ rev:hasReview file:///home/turing/movies.n3#N7... ; indicando URIs (Uniform Resource Identifier) para la película, y su reseña correspondiente. Además se tiene prefijos como dc, rev e imdb los cuales se encuentran definidos al inicio del archivo según:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix imdb: <http://www.csd.abdn.ac.uk/~ggrimnes/dev/
imdb/IMDB#> .
@prefix rev: <http://purl.org/stuff/rev#> .
```

Al definirlos de esta manera el predicado rev:hasReview refiere al URI http://purl.org/stuff/rev#/hasReview con lo que se tiene una mayor grado de estandarización pues otro proyecto podría referirse al mismo predicado.

En nuestra estructuras las personas dejan reseñas y para describirlas utilizamos la estructura foaf, esta estructura es ideal para representar usuarios de redes sociales. Para la cual definimos los siguientes prefijos:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <https://www.perceive.net/
                    schemas/20031015/relationship/> .
```

Utilizando esta estructura podemos representar un usuario que tiene 3 amigos definidos por el predicado ”friendOf”:

```
<file:///home/turing/users.n3#jordi15> a foaf:Person ;
    foaf:mbox "jordi@uni.pe" ;
    foaf:name "Jordi Bardales" ;
    foaf:nick "jordi15" ;
    rel:friendOf <file:///home/turing/users.
    n3#felipeturing>,
        <file:///home/turing/users.n3#jesus15>,
        <file:///home/turing/users.n3#julio15> .
```

Las reviews han sido modeladas considerando los campos disponibles en el prefijo rev

```
<file:///home/turing/movies.n3#Nc97... > a rev:Review ;
    dc:date "2021-08-21T00:00:00"^^xsd:dateTime ;
    rev:maxRating 5 ;
    rev:minRating 0 ;
    rev:rating 1 ;
    rev:reviewer <file:///home/turing/users.n3#jesus15> ;
    rev:text "No me gusto" .
```

Todas estas propiedades serán utilizadas es durante las consultas de SparQL

III. Descripción de recursos usando RDF

En Web Semántica un recurso es algo disponible en la Web para ser utilizado o procesado, por ejemplo, texto, una imagen, una página web, un archivo, etc. RDF nos permite describir estos recursos de manera estandarizada de tal forma que puedan representar un conocimiento y sean distribuidas en internet gracias a las URI's.

Para una representación visual de los triples en N3 se utilizará un grafo dirigido, en el cual el nodo es el sujeto, la arista el predicado y el objeto, que puede ser un literal u otro sujeto anidado esta representado por otro nodo, como se observa en la figura 1 y 2.

```
<file:///home/jordi/users.n3#jesus> a foaf:Person ;
    foaf:mbox "jtorrejon@uni.pe" ;
    foaf:name "Jesus Torrejon" ;
    foaf:nick "jesus" ;
    rel:friendOf <file:///home/jordi/users.n3#katherine> .

<file:///home/jordi/users.n3#katherine> a foaf:Person ;
    foaf:mbox "kathe@uni.pe" ;
    foaf:name "Katherine" ;
    foaf:nick "katherine" ;
    rel:friendOf <file:///home/jordi/users.n3#jesus> .
```

Figura 1. Triple en N3

En el presente informe el recurso va a describir una red social en el cual los usuarios podrán hacer comentarios de una películas y recomendarle a sus amigos. Se tiene

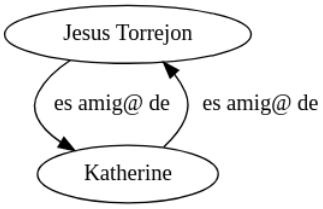

Figura 2. Representación en el grafo

para ello dos recursos, uno para los usuarios y el otro para las películas y revisiones.

En este ejemplo (figura 3) se utiliza el archivo el cual tiene las relaciones de amistades.

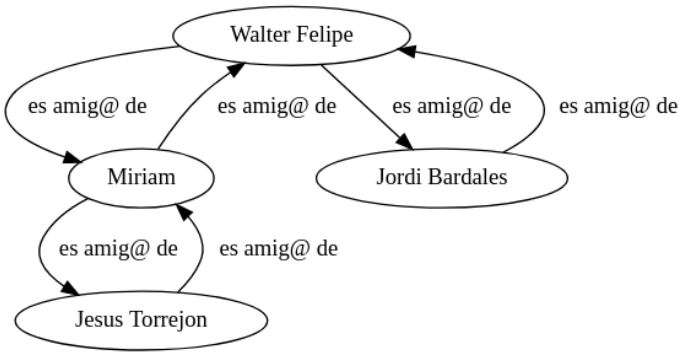

Figura 3. Grafo de amistades

En este grafo (figura 4) se muestra la información del archivo donde se almacenan las películas con sus revisiones.

Figura 4. Películas dirigidas por Cristofer Nolan

En este último ejemplo (figura 5) se combina la información de ambos recursos. Esto es posible gracias a los URI's los cuales nos permiten identificar a un triple de forma única. Primero se obtiene el nick de usuario en su archivo, luego se buscan todas las revisiones asociadas al usuario en el archivo de películas.

III-A. Funcionalidades del programa

1. Agregar nuevo usuario (nombre, nick y correo).
2. Listar los nicks de los usuarios.
3. Consultar si un usuario está en **user.n3**.
4. Agregar nueva amistad entre dos usuarios.
5. Mostrar datos de un usuario por nick.
6. Listar todas las amistades de **user.n3**.

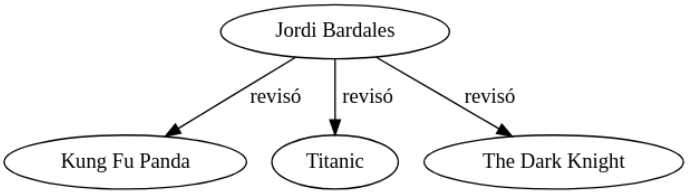

Figura 5. Películas revisadas por Jordi Bardales

7. Listar las amistades de un usuario por nick.
8. Agregar datos de la tienda (Cine).
9. Agregar una nueva película (nombre, fecha, directores y actores principales).
10. Registrar una revisión (valoración, comentario y fecha de revisión) a una película por parte de un usuario.
11. Consultar si una película está en **movies.n3**.
12. Recomedar las películas vistas por las amistades de un usuario.
13. Mostrar datos de una película por url.
14. Listar películas mejor valoradas de acuerdo a la valoración de IMDb.
15. Mostrar películas revisadas por un usuario.

IV. Manipulación de RDF

Para la manipulación de RDF se usa principalmente la librería **RDFlib** de Python, la cual brinda ciertos métodos para la serialización y parseo entre **grafos conjuntivos**[2] de RDF y, en nuestro caso, una estructura N3, asímismo métodos para la manipulación del grafo y en consecuencia hacer inferencias más personalizadas con SparQL, que según la documentación oficial viene con RDFlib en su versión SPARQL 1.1 Query y SPARQL 1.1 Update.

Importamos la biblioteca IMDB, para principalmente retornar los datos y metadatos de una película dada su URI para luego ser adaptado a un conjunto de triples con el mismo sujeto. Por otro lado desde RDFlib importamos las funciones: *RDF*, *BNode*[3], *ConjunctiveGraph*, *URIRef*, *Literal*, *Namespace* y los espacios de nombres *FOAF*, *DC*, *IMDB*, *REL* y *REV*. Como ya hemos mencionado en la sección II, en N3, se almacena cada parte de un triple con URIs, estás se pueden enlazar con prefijos definidos en un espacio de nombres, o sencillamente usar literales como simples cadenas de texto. Para la identificación de una cadena de texto como URI se usa *URIRef*, asimismo usamos nodos en blanco para generar un código aleatorio que permite participar de forma anónima en los triples y RDF.type para que en trabajos futuros se pueda manipular los esquemas con RDFS y tener más abstracción.

[2]Los grafos conjuntivos son múltiples grafos con nombre que se diferencian por el contexto

[3]Blank Node o nodos en blanco, no hacen referencia a algún otro triple del N3, es decir son anónimos.

El objeto grafo tiene métodos que implementan : La inserción de triples con el método **add()**, las consultas de triples con el método **query()** usando el lenguaje SparQL en la cadena de argumento, la configuración de prefijos con el método **bind()**, la serialización con el método **serialize()** y el analizador sintáctico (parsing) del N3 a un objeto grafo con el método **load()**.

A continuación explicamos la implementación de las funcionalidades más importantes, que se menciona en la sección III, en métodos de nuestro programa y ver los resultados que estos muestran cuando se ejecutan. Pero antes listamos una serie de reglas para las inferencias o consultas a nuestras estructuras, las cuales son

1. Las relaciones de amistad son bidireccionales
2. Las películas pertenecen al conjunto de mejor valoradas cuando tienen como mínimo 2 revisiones.
3. La lista de películas mejor valoradas están ordenadas en forma descendente respecto al indicador IMDb Rating[4], el cual sigue la siguiente formula para dar una calificación ponderada a cada película

$$WR = \left(\frac{v}{v+m}\right) R + \left(\frac{m}{v+m}\right) C$$

   Donde WR es la calificación ponderada de la película, R la media de las valoraciones, v el número de revisiones, m la mínima cantidad de revisiones requeridas para entrar al conjunto de mejor valoradas (en nuestro caso $m = 2$) y C es la media de las valoraciones de todas las películas en el N3.
4. Las películas recomendadas a un usuario específico con aquellas que han sido revisadas por las amistades de dicho usuario.
5. Una película puede ser revisada por muchos usuarios y un usuario pueden revisar muchas películas.

Principalmente tenemos tres clases llamadas, *DoConjunctiveGraph*, *UserFactory* y *Store*, las dos últimas heredan las propiedades y atributos de la primera para instanciar cada una un grafo conjuntivo dedicado a la manipulación de usuarios (**users.n3**) y a la de películas (**movies.n3**), tal cual se representa en la figura 6.

En la primera clase *DoConjunctiveGraph* en cuyo constructor se instancia un grafo conjuntivo y se carga, si existe, el archivo con formato N3 donde se van a registrar los triples, sino se crea uno, con los prefijos generales *DC* y *FOAF*, en alguna ruta indicada. Además implementa el método **save()** que serializa el objeto graph en el archivo N3 indicado. Luego si se desea contar el número de triples del grafo se llama al método **__len__()**.

El la clase *UserFactory*(*DoConjunctiveGraph*), se sobrecarga el constructor heredado para instanciar el grafo que hace referencia a **user.n3** y agregar el prefijo *REL*, para establecer semánticamente relaciones de amistad

[4]Formula de IMDb para valorar una película.

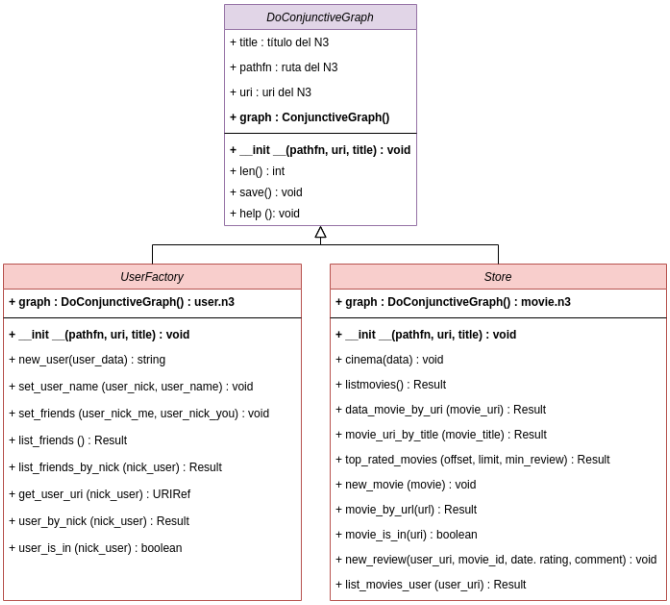

Figura 6. Diagrama de clases

entre los usuarios. Los métodos que se implementan en esta clase son esencialmente para la manipulación de **user.n3**, tales como agregar un conjunto de 4 triples para la creación de un usuario con su nick (debe ser único, para posteriores consultas personalizadas de triples), correo electrónico y nombre, validando que el nick no se encuentre registrado con la sentencia: **user triple in graph?**; los predicados que se usan en este método son : **a**, **foaf:mbox**, **foaf:name** y **foaf:nick**.

Análogamente se procede para registrar una relación de amistad, la cual es bidireccional, entre dos usuarios identificados con sus nicks, este método consta de registrar dos triples con el predicado **rel:friendOf**. Además se implementa una serie de métodos de consulta con la sentencia **SELECT ?s WHERE {?s p o}** de SparQL como argumento del método **graph.query()**, el cual retorna un *result* con los registros para luego ser iterado e imprimir el resultado. Dicha sentencia **SELECT** varía en forma en cada consulta, las cuales son: listar todos los usuarios, todas las relaciones de amistad, los amigos, amigas y la información personal completa de un cierto usuario. Además se implementa un método para, dado el nick de un usuario, retornar la URI completa, para posteriores consultas más elaboradas con SparQL en la siguiente clase.

Finalmente en la clase *Store(DoConjunctiveGraph)* se sobrecarga el constructor heredado para instanciar el grafo que hace referencia a **movies.n3** y agregar los prefijos *IMDB* y *REV*, para registrar triples de películas con atributos como director, reparto de actores, genero, año de estreno y por revisión de un usuario, la valoración y comentario. Dentro de los métodos más interesantes están aquellos que implementan el registro de una nueva revisión por parte de un usuario y la lista de películas mejor valoradas de acuerdo a la formula de valoración de IMDb.

En el primero agregamos un triple cuyo sujeto es la URI de la película revisada, el predicado **rev:hasReview** y el objeto una URI que hace referencia al sujeto de otro triple que contiene los atributos de revisión. Y en el segundo listamos en orden descendente las películas mejor valoradas de acuerdo a la formula de IMDb, véase la figura 7.

Película	Número de reviews	Valoración promedio(0-5)	IMDb Rating
The Dark Knight	5	4.4	4.2
Kung Fu Panda	2	4.5	4.1
Shrek	4	4.2	4.0
Inception	3	3.7	3.6
Madagascar: Escape 2 Africa	5	3.6	3.6
Interstellar	2	3.5	3.6
The Little Prince	3	3.3	3.4
The Lion King	4	3.2	3.4
Superman	2	3.0	3.3

Figura 7. Resultado del método *Store:top_rated_movies(.)*

V. Conclusiones

Las conclusiones que rescatamos de la elaboración de este proyecto son :

1. Se pudo adaptar IMDb a una estructura RDF/N3 consistente de triples de usuarios, peliculas y revisiones principalmente.
2. Se ha consultado las estructuras N3 con SparQL mostrando inferencias interesantes tales como listar las películas mejor valoradas por IMDb Rating o mostrar todas las relaciones de amistad entre usuarios.
3. Se utilizó el URI del nick el cual identifica de forma única al usuario en otro archivo N3 para obtener las películas que revisó, con ello se pudo combinar ambos archivos N3 y así obtener más información.

Referencias

[1] Allemang, D., & Hendler, J. (2011). Semantic Web for the Working Ontologist: Effective
[2] RDFlib 5.0.0, Official Documentation, Recuperado de : https://rdflib.readthedocs.io/en/stable/
[3] RDF 1.1 Turtle, Terse RDF Triple Language, Recuperado de : https://www.w3.org/TR/turtle/
[4] IMDbPY, Official Documentation, Recuperado de : https://imdbpy.readthedocs.io/en/latest/
[5] SparQl, SPARQL Query Language for RDF, Recuperado de : https://www.w3.org/TR/rdf-sparql-query/