SISTEMA DE RECUPERACIÓN SEMÁNTICO

Recuperación de Información – Francisco Ferraz y Guillermo Cruz

Resumen

En el desarrollo de las prácticas y el trabajo se ha implementado un sistema de recuperación semántico utilizando RDFS, SKOS, OWL, Jena y SPARQL y se ha comparado con el sistema de recuperación tradicional de la anterior entrega.

Desarrollo

1. RDFS

Partiendo del modelo RDFS que se creó para la tutoría de diciembre, se han mejorado algunos detalles:

1. Se asignaron dos subclases a la clase Documento, TFG y TFM. De esta manera el tipo de un recurso que representa a un trabajo será TFG o TFM, y por herencia se deduce que es un Documento.
2. Se creó una clase Persona.
3. Se guardaron los subjects como Concepts de SKOS.
4. Se añadieron las propiedades de Organización e Idioma.

2. SKOS

Se ha creado un fichero *skos.xml* en el que se desarrollan los distintos temas más relevantes para la creación del grafo RDF según las necesidades de información con las que se opera. En el modelo terminológico está el termino principal, que sería el que aparece directamente en la necesidad de información, y después se han declarado otros términos similares con la propiedad skos:broader cuyo objeto es el término principal, de esta manera se declaran como hijos de éste y por medio de inferencia posteriormente se pueden deducir nuevos temas en los documentos.

3. SemanticGenerator

En primer lugar, al igual que en la primera parte de la asignatura para el sistema tradicional, se han obtenido todos los atributos de los ficheros XML que representan a los documentos gracias a la clase NodeList y el método getElementsByTagName(). De esta manera, una vez que ya se tiene toda la información de los documentos, como el creador, la descripción, fecha, etc., se ha procedido a crear el grafo que representa la colección semántica.

En primer lugar, se han creado los recursos de tipo TFG o TFM, que representan un recurso de tipo Documento, cuya URI es del tipo

http://www.trabajos.fake/trabajos/coleccion/<id documento>. A continuación, se han ido añadiendo las propiedades de dicho recurso: título, identificador, descripción , fecha, 'publisher' e idioma. Para añadir los creadores del documento, primero se crea un recurso del tipo Persona, al que se le añade la propiedad del nombre, y posteriormente se añade este recurso como creador al documento.

Para añadir los temas o 'subjects' correspondientes a cada documento, y que además pudieran ser utiles para realizar las consultas más eficientemente, se ha procesado el modelo terminológico skos.xml. El proceso ha consistido en recuperar todos los términos que aparecen en el modelo, por medio de su etiqueta 'prefLabel', y a continuación mirar documento a documento si ese término se encontraba en el campo 'subject' o en el campo 'description' de dicho documento. Si así era, se añadía al recurso que representaba el documento la propiedad 'subject' con ese término como objeto. De esta manera, al realizar la consulta, habrá que mirar dicha propiedad para devolver los documentos adecuados. Cabe destacar que a los campos 'subject' y 'description' se les hacía un filtrado de caracteres especiales antes de buscar si el término se encontraba en ellos, quitando todas las tildes y pasando todo a minúsculas.

Por último, a esta colección generada, se le añade por medio del método createUnion() de la clase ModelFactory el modelo owl, para que posteriormente se pueda realizar la inferencia ya que es en dicho modelo donde se encuentran las reglas.

4. SemanticSearcher

Para realizar las búsquedas en SPARQL se ha implementado un módulo que hace uso de las librerías *ModelFactory* de Jena para trabajar con el grafo RDF, y un *HashMap* para almacenar las necesidades de información.

Se itera sobre este mapa para realizar cada una de las consultas sobre el grafo RDF y almacenar los documentos resultantes en un fichero de texto. El formato de almacenamiento es, como se indica en el enunciado, del estilo:

02-2 oai_zaguan.unizar.es_5460.xml

02-2 oai_zaguan.unizar.es_6453.xml

…

5. Consultas SPARQL

Se han creado 5 consultas SPARQL, una por cada necesidad de información, que hacen referencia a los elementos que componen nuestro modelo RDF. En ellas se han intentado recuperar todos los documentos que puedan ser importantes para las consultas, sin dejar de lado las distintas condiciones requeridas en cada necesidad de información. Además, se han añadido índices textuales que apuntan a la descripción

de los documentos para clasificarlos en función de su relevancia. Se utiliza la misma técnica que en la práctica 6, una ordenación en base a queries de texto.

Un ejemplo de filtro por rangos utilizado para conseguir los documentos en el rango de fechas solicitado es el siguiente:

```
SELECT distinct ?doc WHERE
{
    { ?doc mt:fecha ?fecha  }
    . FILTER ((xsd:integer(?fecha) >= 20110101)
              &&
             (xsd:integer(?fecha) <= 20190101))
}
```

Imagen 1. Filtro para fechas

Donde se puede observar que las fechas se han almacenado con el formato yyyymmdd.

6. Evaluación

Se ha aplicado el mismo programa utilizado para el sistema tradicional a los resultados obtenidos para el sistema semántico, consiguiendo datos como la precisión, recall, MAP, etc. Los datos obtenidos se han comparado con los del sistema tradicional y se ha discutido sobre cuáles eran mejores y los motivos principales de sus diferencias.

En la evaluación del sistema, se han conseguido unas medidas ligeramente inferiores al sistema tradicional, lo que puede ser consecuencia de no haber realizado el ranking de los documentos adecuadamente, puesto que se opina que las medidas deberían ser bastante mejores, ya que en el sistema semántico la información puede almacenarse de una manera más eficiente gracias al uso de tesauros y conseguir buenos resultados con la inferencia, además de escribir las consultas SPARQL a mano y ceñirlas lo máximo posible a la necesidad de información en vez de procesarlas como en el sistema tradicional.

A continuación se muestran las medidas principales de cada consulta:

```
INFORMATION_NEED    01-2
precision   0.280
recall  0.400
F1  0.329
prec@10 0.300
average_precision   0.378
```

Imagen 2. Resultados necesidad 1

De la consulta sobre la necesidad 2 no se ha conseguido devolver ningún resultado.

```
INFORMATION_NEED    06-1
precision   0.160
recall  0.308
F1  0.211
prec@10 0.100
average_precision   0.284
```

Imagen 3. Resultados necesidad 3

```
INFORMATION_NEED    11-4
precision   0.020
recall  0.022
F1  0.021
prec@10 0.100
average_precision   0.143
```

Imagen 4. Resultados necesidad 4

```
INFORMATION_NEED    15-4
precision   0.380
recall  0.317
F1  0.345
prec@10 0.100
average_precision   0.314
```

Imagen 5. Resultados necesidad 5

```
TOTAL
precision   0.168
recall  0.209
F1  0.181
prec@10 0.120
MAP 0.224
```

Imagen 6. Resultados sistema