**GENERADOR SEMÁNTICO**

16

**Recuperación de Información**

**Sistema de Recuperación Semántico**

**Jorge Andrés Galindo 679155 Javier Aranda García 679184**

Partiendo del grafo RDFS que se había diseñado para modelar nuestro sistema de recuperación semántico, se procedió a realizar un programa con la ayuda de la librería de Java Jena que permitiera crear nuestro sistema semántico. Para ello, la primera tarea era obtener la información de los metadatos XML proporcionados. Estos datos se obtuvieron mediante un parser XML que ya se utilizó en el sistema de recuperación tradicional, guardando la información en un objeto de tipo “Etiqueta” con el nombre del campo y su contenido.

Seguidamente, una vez ya se tenían parseados los documentos, se iba obteniendo la información de cada campo mediante un bucle, y se creaban las propiedades equivalentes a nuestro modelo RDFS con los valores obtenidos de los campos. Dichas propiedades se creaban de tipo “Literal” O “Resource” según fuera el caso en el modelo RDFS, y prestando especial atención al tipo de dato literal, ya que se tuvo que modificar el modelo RDFS para poder realizar las búsquedas requeridas con un mayor rango de aproximación.

Además, se decidió introducir en nuestro campo fecha, que en un principio estaba orientado tan sólo a obtener las fechas del campo “date” de los metadatos, el valor de todas aquellas fechas incluidas tanto en el campo descripción como en el campo título. La indexación de los nuevos elementos fecha en el modelo RDF era obvia, ya que se debía seguía el mismo procedimiento que para el resto de datos, pero la dificultad venía dada por saber que elemento del texto era una fecha. Para obtenerlo, se fijó un sencillo patrón que fuera recorriendo el texto de los campos de los que se quería obtener la fecha e identificara una serie de 4 dígitos seguidos, que en la mayoría de los casos son años de fechas.

Teniendo ya las propiedades creadas, junto con los literales necesarios para completar el modelo RDF, simplemente con la ayuda de JENA se iban añadiendo a la instancia de tipo “Model” que representaba el grafo RDF semántico. Al final de todo el proceso, se escribía el modelo en un fichero indicado como parámetro en formato XML, que en procedimientos posteriores se utilizaría para realizar las consultas en lenguaje SPARQL.

**SKOS**

Teniendo ya el modelo RDF creado con los recursos y literales necesarios, el siguiente paso era crear el modelo terminológico SKOS. Este modelo se creaba en formato XML y consistía en una serie de etiquetas que identificaban un término en concreto, pudiendo añadir alternativas para ese término (sinónimos). Además, el modelo terminológico SKOS permite indicar con atributos el idioma de un término, por lo que dada una palabra, se pueden obtener textos en otros idiomas sobre ese tema, algo que favorece en gran medida los resultados de las consultas.

Sabiendo el concepto y el significado de lo que se necesitaba, se procedió a diseñar el modelo SKOS con la mayor riqueza semántica posible. Para ello, se realizó un estudio de los juicios de relevancia de aquellos documentos que trataban sobre un cierto tema de las consultas y no habían sido escogidos en primeras posiciones por no contener exactamente las palabras buscadas. Una vez los teníamos localizados, analizamos posibles sinónimos de las palabras de la consulta y lo contenido en el fichero que iba sobre el tema, y los añadimos al fichero SKOS para que los reconociera como términos relacionados.

Con el fichero SKOS terminado, nuevamente con JENA se generó el grafo de los conceptos con propiedades “prefLabel” (término principal) y “altLabel” con los sinónimos de cada uno de los términos, y se iban añadiendo al mismo modelo generado para los documentos. De este modo se consiguió escribir en el fichero final nuestro grafo RDF completo, conteniendo todo el modelo terminológico SKOS que habíamos diseñado centrado especialmente en mejorar los resultados de las consultas en SPARQL.

**MODIFICACIÓN RDFS**

Dado el grafo inicial, se tuvieron que realizar modificaciones en el modelo RDFS de nuestro sistema semántico final. El campo fecha de los trabajos era de tipo literal, lo que no permitía realizar búsquedas por rangos de fechas necesarios en las consultas. Para conseguirlo, se estudiaron distintos tipos de datos literales que nos pudieran servir de ayuda como eran “dateTime”, “date”, etc. Finalmente, nos decidimos por el tipo “gYear”, ya que las consultas pedían rango de fechas, pero tan sólo de años, por lo que era el tipo más indicado para guardarlas en nuestro sistema de recuperación semántico.

Otra de las cosas que cambiamos fue la organización. En un principio, la vimos como una propiedad del trabajo, pero nos dimos cuenta de que era más adecuado definirlo como una clase en sí misma, ya que modelaba mejor el sistema. Además, añadimos propiedades como el nombre a la organización y las personas, que en el grafo inicial habíamos omitido y era necesario. Finalmente, se añadieron etiquetas textuales a cada una de las propiedades para definirlas e indicar que representan.

**CONSULTAS**

Una vez teníamos le grafo RDF final junto con el modelo terminológico SKOS, se procedió a realizar las consultas. Dichas consultas se realizarían en lenguaje SPARQL sobre nuestro modelo y se guardarían en un fichero de salida que se pasaría por parámetro. Las consultas a realizar eran 5 y venían fijadas por el fichero de necesidades de información que ya se había utilizado en el sistema de recuperación tradicional, salvo que en este caso no se leían en lenguaje natural y se traducían lo mejor posible mediante java a los campos de nuestro modelo.

En el sistema de recuperación semántico, las consultas se escribían a mano en el fichero de necesidades que se pasarían al Java, y debían estar ya en lenguaje SPARQL. Por ello, se construyeron directamente de la manera que creíamos que mejor se adaptaba a la expuesta en lenguaje natural, obteniendo las palabras clave de la consulta, y aplicándolas a los campos apropiados. Además, obtuvimos las fechas y realizamos filtros por rangos facilitados por el cambio del tipo fecha comentado en el apartado anterior.

Un ejemplo de consulta con filtro de fechas sería el siguiente:

// Ejemplo de consulta.

**BUSCADOR**

De igual modo que el programa para construir el grafo RDF, se ha diseñado e implementado otro capaz de realizar las búsquedas mediante la ayuda de Jena. El programa recibe como argumentos el modelo RDFS, el grafo semántico RDF, las necesidades de información con el formato especificado y en lenguaje SPARQL y finalmente la ruta del fichero de salida donde se guardarán los resultados. Este programa simplemente recorre las líneas del fichero de necesidades y realiza las búsquedas sobre el grafo.

En primer lugar, sabiendo que se parte de la idea de ir realizando las consultas una a una sobre el modelo, se debe cargar dicho modelo RDF en el programa. Para ello, se hace uso de la clase “FileManager” que permite cargar un modelo desde un fichero a un objeto de la clase “Model” como el que se ha tratado en el programa para generar el grafo. Este objeto, junto con el del tipo “Query” van a permitir realizar las consultas de las necesidades requeridas.

El objeto “Query” nombrado anteriormente se genera con el texto en SPARQL de la consulta que leemos del fichero, y se le pasa junto con el modelo al objeto “QueryExecution” que realiza la consulta y devuelve un iterador con los resultados. Los textos obtenidos se van guardando con el formato especificado en el fichero de salida y se repite el mismo proceso para todas las consultas, consiguiendo de este modo aplicar el lenguaje SPARQL de nuestras consultas sobre el modelo diseñado por nosotros.

**RESULTADOS**

CONSULTA 1

En la primera consulta se pretenden buscar aquellos textos o trabajos sobre información geológica, concretamente en España, obteniendo los siguientes resutlados.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Como se puede ver,

CONSULTA 2

En la segunda consulta, se pide información sobre enfermedades infecciosas, pero preferiblemente entre los años 1980 y 1990, lo que da como resultado:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

Los textos obtenidos

CONSULTA 3

La tercera de las consultas pretende obtener información de la colección relacionada con algoritmos de funciones geométricas y, además, si pueden no tener implementación mejor. Se consiguen los siguientes resultados:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

CONSULTA 4

Esta consulta busca principalmente los trabajos sobre robótica a partir del año 2005 incluido y se consiguen los siguientes documentos en las primeras posiciones:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

CONSULTA 5

Finalmente, la última consulta trata sobre la inteligencia artificial, buscando los proyectos de los últimos 5 años (2012-2017). De la consulta se extraen los siguientes resultados:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

**EVALUACIÓN**

Igual que para el sistema de recuperación tradicional, se ha aplicado el mismo programa a los resultados obtenidos para el sistema semántico, consiguiendo datos como la precisión, recall, MAP… Estos datos se han comparado con los del sistema tradicional y se han expuesto conclusiones sobre cuales eran mejores y los principales motivos que han desencadenado esos datos.