Universidad Zaragoza

Proyecto Fin de Carrera
Ingeniería en Informática

Sistema de extracción de información semántica de la DBpedia

Autor: Guillermo Esteban Pérez

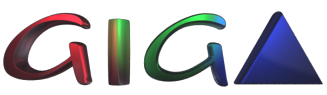
Director: Carlos Bobed Lisbona

Ponente: Francisco José Serón Arbeloa

Septiembre de 2012

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Departamento de Informática e Ingeniería de Sistemas Universidad Zaragoza



Sistema de Extracción de Información Semántica de la DBPedia

RESUMEN

Hoy en día, nos podemos encontrar cada vez con más información en la Web. Los usuarios, con la llamada Web 2.0, se han vuelto también proveedores de información y cada vez son más los datos disponibles en Internet. La Web Semántica dota toda ésta información de semántica y relaciones, de manera que éstos recursos, hasta ahora sólo consumibles por seres humanos, pueden ser entendidos y tratados por máquinas.

Para poder dar formato semántico a los recursos presentes en la Web se usan ontologías, que definen de una manera exhaustiva y rigurosa el modelo conceptual de uno o varios dominios dados. Éstas son utilizadas para etiquetar distintos tipos de recursos de manera que los contenidos pasen a ser procesables por los computadores, pasando a formar parte de esta forma de la Web Semántica, donde los recursos se encuentran estructurados según dichas ontologías.

La aparición de estos datos estructurados abre un nuevo camino para las técnicas de *Information Retrieval* (IR). En este momento, estas técnicas básicamente se realizan como búsquedas sintácticas y probabilísticas, buscando exclusivamente por lexemas y realzando el peso de los resultados más populares en cada búsqueda. El objetivo de este proyecto ha sido el diseño e implementación de una solución de búsqueda híbrida basada en keywords que utiliza la semántica de los recursos para enfocar la búsqueda y posibilitar la búsqueda sobre datos estructurados. Llamamos a este tipo de búsqueda híbrida ya que utiliza técnicas sintácticas y semánticas. Esto se ha implementado sobre un servicio web con los métodos necesarios para poder realizar búsquedas sobre el dominio definido por el usuario. El sistema permite al usuario realizar búsquedas de este tipo sin necesidad de disponer del conocimiento sobre lenguajes de consulta formales que este tipo de búsqueda normalmente requieren.

Para realizar tales búsquedas ha sido necesario realizar el aprendizaje de diversas tecnologías hasta ahora no utilizadas durante la carrera. Entre estas tecnologías destacan primero los lenguajes de modelado semántico propuestos por el W3C: OWL y RDF. OWL es un lenguaje para la representación de ontologías basado en Lógicas Descriptivas (Description Logics, DL), mientras que RDF es un lenguaje de modelado de conocimiento con menor expresividad destinado a describir recursos de acuerdo a vocabularios u ontologías externas. El lenguaje RDF ha sido tomado como estándar por la iniciativa Linked Data para la publicación de datos. Dicha iniciativa propone la publicación de datos de manera que éstos queden vinculados semánticamente entre sí.

Después de estudiar la relación entre RDF y OWL y su correcto uso, se estudió el lenguaje de consultas SPARQL, sucesor semántico de SQL y lenguaje estándar

del W3C para la consulta sobre RDF. Las consultas SPARQL en este proyecto se realizan sobre uno de los puntos de acceso a la DBPedia. Dicho proyecto se dedica a la extracción automática de información semántica de la Wikipedia y actualmente es un referente por la cantidad de datos de los que dispone.

Aunque DBPedia dispone de una cantidad de datos considerable, estos están etiquetados según varias ontologías, proporcionando cada una una visión distinta de los contenidos o, como ocurre a menudo, la misma visión doblemente etiquetada o con sutiles diferencias; el estudio de este problema para poner en valor la cantidad de datos que proporciona la iniciativa Linked Data también forma parte del proyecto con objeto de conseguir una recuperación de datos más útil de datos.

Por las especificaciones de un proyecto paralelo, el cual requería de este PFC como punto para recuperar información, y con los conocimientos adquiridos durante la fase de aprendizaje, se ha decidido crear un servicio web con diversos métodos que permitan la realización de estas búsquedas de manera desacoplada y general, de manera que el sistema desarrollado no sólo sirviese como punto de acceso para el proyecto paralelo y en el dominio particular especificado (en este caso, el dominio de la mecánica de fluidos), sino que este pudiese ser usado en cualquier otro dominio y en otras herramientas con diversos propósitos.

Agradecimientos

Quiero agradecer a madre su apoyo incondicional durante el transcurso de la carrera y de este proyecto; ponerme sobre aviso o tranquilizarme en los momentos adecuados ha sido de gran ayuda y sé que ha sufrido tanto si no más que yo durante este ciclo de mi vida. También a mi hermana, primos y tíos, que me han ayudado con lo que ha estado en su mano. Se termina un período de mi vida, que espero recordar con ilusión, y empieza otro, no sé si peor o mejor, pero distinto sin duda.

También me gustaría dar las gracias a todos los amigos que he podido hacer durante la carrera, tanto los que siguen en Zaragoza y en contacto como los que no, todos ellos me han ayudado de una y otra manera. Entre todos hemos sobrellevado los mejores y peores momentos de los últimos años, no puedo describir lo mucho que me ha ayudado que estuviesen siempre ahí.

Por supuesto, quiero también agradecer a Paco Serón, por ofrecerme la oportunidad de trabajar en un proyecto tan interesante como este, y a Carlos, por todo lo que me ha ayudado con el aprendizaje de las tecnologías semánticas, al igual que por su paciencia y ayuda prestada durante el proyecto; todo lo que me ha enseñado en relación al proyecto, durante la redacción del artículo que publicamos, y finalmente en el trabajo que compartimos en paralelo, ha sido de una ayuda inestimable.

Sin todos vosotros no podría estar presentando este proyecto, gracias.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El presente documento representa la memoria de trabajo del proyecto de fin de carrera "Sistema de Extracción de Información Semántica de la DBPedia", realizado por el alumno Guillermo Esteban Pérez cuyo director y ponente son, respectivamente, Carlos Bobed Lisbona y Francisco José Serón Arbeloa. Este proyecto se ha realizado dentro del marco del Departamento de Informática e Ingeniería de Sistemas de la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza, concretamente dentro del Grupo de Informática Gráfica Avanzada.

Cada vez es más la información estructurada disponible en Internet, sin embargo, el volumen de los datos a tratar, y el hecho de que todos los recursos no estén perfectamente etiquetados y enlazados, hacen necesarias nuevas técnicas de búsqueda que exploten la semántica y sean eficientes. En este proyecto se ha desarrollado una aproximación híbrida basada en keywords que utiliza la semántica de los recursos para enfocar la búsqueda y posibilitar este tipo de consultas sobre datos estructurados. Se ha implementado sobre un servicio web con los métodos necesarios para poder realizar búsquedas sobre el dominio definido por el usuario. Dicho servicio explota la información presente en la DBPedia a través de su punto de acceso público para 1) realizar búsquedas eficientes de palabras clave sobre un dominio específico, y 2) posteriormente, refinar las búsquedas realizadas por el usuario.

El sistema desarrollado se ha realizado bajo un contexto de investigación. Los resultados del proyecto han sido publicados en la conferencia KES 2012 [16], dicho trabajo ha sido realizado por el Director de este PFC Carlos Bobed, el alumno y autor del mismo proyecto Guillermo Esteban, y el Profesor Titular de la Universidad de Zaragoza Eduardo Mena. El artículo ha sido aceptado y será presentado en San Sebastián del 10 al 12 de Setiembre de 2012.

Este PFC está a su vez relacionado con el proyecto de fin de carrera "Interface de Usuario Multimodal Asistido con Agente Virtual", realizado por Daniel Martínez Millán. Ambos proyectos se enmarcan dentro del proyecto Alfa III GAVIOTA [27] de la Unión Europea, y forman juntos una herramienta web que permite obtener información relacionada con la mecánica utilizando una interfaz. El trabajo en esta

herramienta se dividió en dos partes, la primera, realizada por Daniel Martínez en su PFC, se encarga de la interfaz de la herramienta utilizando agentes virtuales. La segunda, realizada en este proyecto, se encarga de proporcionar los datos semánticos.

1.1. Motivación

La Web ha hecho que una gran y cada vez mayor cantidad de información esté disponible a sus usuarios. La aparición de nuevos paradigmas, como la llamada Web 2.0, ha hecho que la información presente sea aún mayor y cada vez crezca más rápido, ya que los mismos usuarios de esta Web se han convertido en suministradores de información. Tal cantidad de información es en muchas ocasiones difícil de tratar por los propios usuarios, ya que está orientado al consumo humano, y las máquinas no pueden aliviar de manera eficiente la carga de trabajo que supone navegar por tal cantidad de información. En este punto es donde aparece la Web Semántica [2], que propone hacer procesables los contenidos de la Web para poder procesarlos automáticamente con computadores; por tal motivo propone añadir información semántica a los recursos disponibles. Para poder aprovechar mejor la información disponible, la iniciativa Linked Data [1] recomienda una serie de prácticas para etiquetar y enlazar recursos estructurados con otros relacionados. De esta manera las máquinas pueden entender qué tipo de recurso están analizando y pueden tratar su información, analizarla y vincularla con otros recursos de la Web, liberando así esa carga de trabajo del usuario.

Para poder etiquetar apropiadamente los recursos existentes se hace uso de las llamadas ontologías, con las que podemos definir de manera exhaustiva, rigurosa, formal y explícita el esquema conceptual de diversos ámbitos de conocimiento. Agrupadas alrededor de la Web Semántica existen un conjunto de tecnologías que nos permiten realizar estas descripciones, tales tecnologías son promovidas y apoyadas por el W3C [1] y son consideradas estándar para su publicación como datos estructurados. Iniciativas como Linked Data ya han hecho pública una gran cantidad de información semántica siguiendo estos estándares.

Ahora ha llegado el momento de poder hacer uso de toda esta información, extraerla y analizarla con agentes informáticos de manera que el usuario le pueda dar una forma y uso a todos estos datos. Desafortunadamente, no solo no es realista suponer que todos los recursos presentes en la Web estén apropiadamente etiquetados y enlazados, sino que los propios usuarios no están familiarizados con los lenguajes de consultas formales que se requieren para recuperar este tipo de información. Actualmente los usuarios están acostumbrados a métodos de búsqueda que se basan en conjuntos de palabras clave, con complejos algoritmos de *ranking* para obtener los resultados más propicios para las palabras clave elegidas.Por ejemplo, Google ofrece resultados relativamente acertados (siempre según el conjunto de palabras

[1] *World Wide Web Consortium*, http://www.w3.org

clave usadas) a lo que el usuario espera de ellos, sin embargo, el proceso usado por este y otros motores de búsqueda populares no tienen en cuenta la semántica de las palabras usadas, solo se basa en su sintaxis y ordena los resultados con métodos probabilísticos, con lo cual el significado final de las palabras no está presente y los resultados pueden llegar a desvirtuarse; por otro lado, resultados no tan populares pueden ser ocultados. Para poder dar soluciones que tengan en cuenta este aspecto es necesario utilizar datos estructurados, y para conseguir que esta información pueda ser de utilidad a los usuarios hay que adaptar los mecanismos de consulta progresivamente y, en la medida de lo posible, hacer el cambio transparente los usuarios, con lo que se han de proponer soluciones que hagan uso de la información semántica disponible sin renegar de los métodos sintácticos que actualmente están en uso y son más familiares al usuario.

En este sentido el proyecto realizado implementa una aproximación híbrida para realizar búsquedas utilizando métodos sintácticos y semánticos, estas búsquedas se realizan sobre un repositorio de datos llamado DBPedia (Sección 2.2.2), y no exigen al usuario conocimientos sobre lenguajes de consulta formales, pidiendo simplemente el conocido conjunto de palabras clave como entrada y ofreciendo una navegación por los resultados que proporciona más información relacionada semánticamente con estos.

1.2. Objetivo y alcance

Los objetivos de este proyecto son:

- Desarrollo de un sistema que permita realizar búsquedas semánticas sobre el punto de acceso público de la DBPedia. Este sistema debe cumplir con los siguientes puntos:
  - Debe ser capaz de buscar conocimiento a partir de palabras clave.
  - Aprovechando la semántica de los datos disponibles, debe poder realizar búsquedas de refinado, obteniendo datos semánticamente relacionados con resultados de búsquedas anteriores.
  - El dominio de las búsquedas debe ser adaptable.
  - La interfaz ofrecida para su uso en distintas herramientas debe abstraer en la medida de lo posible de los aspectos semánticos más técnicos con objeto de mejorar su usabilidad.
- El estudio y aprendizaje de las tecnologías relacionadas con la Web Semántica para su aplicación en la búsqueda de datos estructurados.

- El estudio de los datos presentes en la DBPedia y el uso que hacen de diversas ontologías con iguales semánticas con objeto de conseguir una recuperación de datos más útil.

- Implementación de una utilidad con interfaz gráfica para poder dar uso a la aproximación ofrecida por el sistema.

1.3. Contenido de la memoria

La presente memoria del proyecto realizado se encuentra dividida en varios capítulos y apéndices:

- **Capítulo 1: Introducción**. Este capítulo describe el ámbito del proyecto y su motivación, así como los objetivos marcados y el marco temporal de su realización.

- **Capítulo 2: Contexto Tecnológico**. Durante este capítulo se listan el conjunto de tecnologías utilizadas para la realización de este proyecto. Lenguajes, herramientas y utilidades relacionadas con su uso, definición de ontologías, acceso a datos e implementación.

- **Capítulo 3: Estado del arte**. El capítulo 3 contiene el estudio del arte realizado, donde se listan y analizan algunas de las soluciones propuestas en artículos de investigación.

- **Capítulo 4: Desarrollo de la solución**. En este capítulo se detallarán los diversos aspectos de la solución obtenida e implementada. Dentro se incluye un extracto del análisis y diseño de la solución y lo relativo a la arquitectura del sistema y su funcionamiento.

- **Capítulo 5: Conclusiones**. Este último capítulo detallará las conclusiones generales y personales obtenidas en la finalización del proyecto, así como una sección sobre trabajos a realizar para futuras mejoras.

- **Apéndice A: Análisis y diseño de la aproximación**. Este anexo contiene el documento de Especificación de Requisitos Software, junto con el diagrama de clases, casos de uso y trazas de evento diseñados antes de la realización del proyecto.

- **Apéndice B: Vocabularios, lenguajes y herramientas**. Durante este capítulo se detallan todas las tecnologías utilizadas en el proyecto. Algunas de las secciones de este apéndice expanden descripciones de capítulos de la memoria con información más específica de la implementación realizada.

- **Apéndice C: Ontología Dominio y construcción de consultas**. En este anexo se expone como ejemplo de Ontología Dominio la utilizada para para el proyecto Alfa III GAVIOTA. También se incluye los pasos para el proceso de creación de esta ontología y cómo el sistema hace uso de ella para construir las consultas.
- **Apéndice D: Manual de usuario**. Este apéndice contiene el manual de usuario para la interfaz gráfica realizada.
- **Apéndice E: Artículo KES 2012**. El último apéndice incluye el artículo íntegro publicado y aceptado en la conferencia KES 2012, exponiendo los resultados de este proyecto.

1.4. Marco temporal del proyecto

En la Figura 1.1 se puede ver la planificación para el desarrollo del proyecto. En la Tabla 1.1 se pueden observar los tiempos requeridos para cada una de las tareas en el proyecto. Calculando las horas trabajadas al día a lo largo del proyecto obtenemos aproximadamente 5 meses y medio dedicados a jornada completa. Período de tiempo que se ha alargado debido a que los últimos 3 meses se estuvo trabajando a tiempo parcial en paralelo para un proyecto del GIGA (Grupo de Informática Gráfica Avanzada).

Durante el transcurso de este PFC también se realizó la publicación de los resultados de este proyecto en la conferencia KES 2012 anteriormente mencionada. Las horas dedicadas a este artículo no se han incluido en los tiempos estimados.

Tarea	Tiempo
Estudio Previo	23h
Documentación	44h
Análisis y Diseño	135h
Implementación de la aproximación	240h
Pruebas de la aproximación	60h
Implementación de la utilidad	100h
Pruebas de la utilidad	24h
Elaboración de la memoria	100h
Total	**726h**

Tabla 1.1: Tiempos requeridos

Nombre	Fecha de inicio	Fecha de fin
Estudio Previo	3/10/11	21/10/11
Documentación	3/10/11	11/11/11
Planificación	7/11/11	11/11/11
Análisis y Diseño	14/11/11	23/12/11
Estudio del Estado del Arte	14/11/11	21/11/11
Estudio de la DBPedia	21/11/11	9/12/11
Especificación de los Requisitos	12/12/11	23/12/11
Casos de Uso	19/12/11	23/12/11
Implementación de la Aproximación	26/12/11	16/03/12
Pruebas de la Aproximación	1/03/12	22/03/12
Implementación de la utilidad	26/03/12	27/04/12
Pruebas de la utilidad	26/04/12	4/05/12
Integración	7/05/12	11/05/12
Elaboración de la Memoria	14/05/12	8/06/12

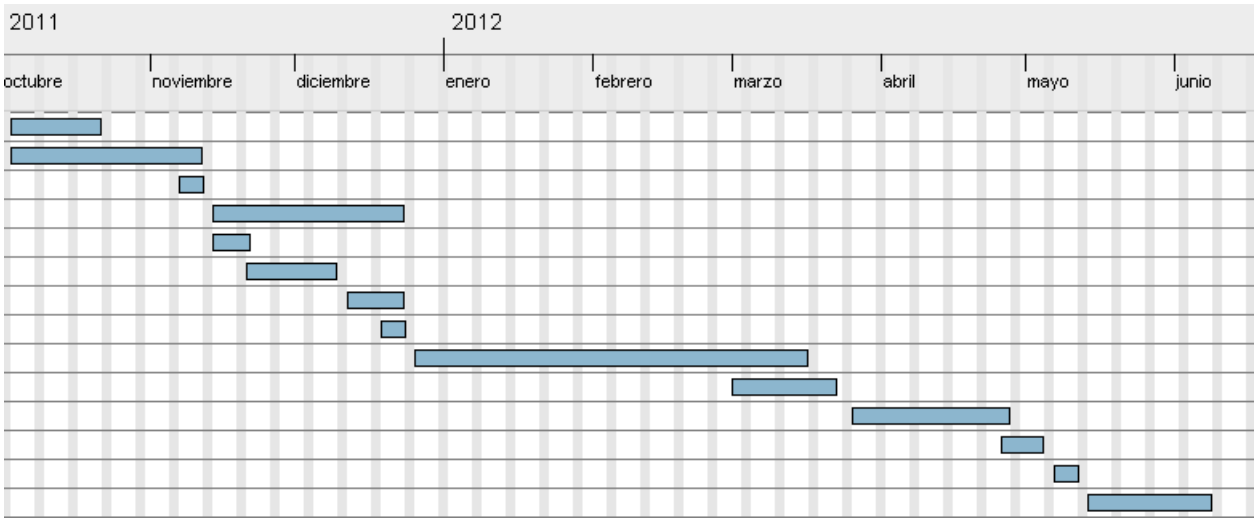


Figura 1.1: Diagrama de Gantt del desarrollo del proyecto

Capítulo 2

Contexto tecnológico

En este capítulo se detallan las diferentes tecnologías que han sido utilizadas durante el desarrollo del proyecto. En primera instancia se explican algunas de tecnologías subyacentes bajo la Web Semántica, tanto en la definición de ontologías como en la recuperación de datos estructurados, de esta manera se puede entender mejor la solución obtenida. Seguidamente, se describirán algunas de las herramientas utilizadas tanto para la recuperación de datos como para la propia implementación de la solución diseñada.

2.1. Ontologías

Para entender el sistema, a continuación se explicará qué es una ontología y se expondrán brevemente algunas de las tecnologías que se usan para su definición.

2.1.1. Definición de ontología

El concepto de ontología es defino de manera breve como la "especificación explícita de una conceptualización" por Tom R. Gruber [10]. Una conceptualización es una abstracción, un modelo conceptual, vista simplificada del mundo que queremos representar.

Esta definición fue revisada y ampliada, añadiendo las nociones de compartida/consensuada y formal, por Studer et al. [17]: "Una ontología es la especificación explícita y formal de una conceptualización compartida. Conceptualización se refiere a un modelo abstracto de algunos fenómenos del mundo, teniendo identificados los conceptos más relevantes del fenómeno. Explícito significa que los tipos de conceptos utilizados y las restricciones en su uso están explícitamente definidos. Formal se refiere al hecho de que la ontología debe ser procesable por máquinas. Compartida

es la idea que de la ontología refleja conocimiento consensuado, esto es, que no es privado a algunos individuos, sino aceptado por un grupo".

Para clarificar el uso y papel que juegan las ontologías, se incluye un extracto de la tesis [11]:

> Mientras los metadatos (mecanismo para etiquetar, catalogar, describir y clasificar los recursos presentes en la Web) sirven para la estructuración de la información, las ontologías hacen posible una semántica para construirlos. Las ontologías son acuerdos, marcos comunes, no sólo para almacenar la información, sino también para poder buscarla y recuperarla. Las ontologías definen los términos y las relaciones básicas para la compresión de distintas áreas de conocimiento. También definen las reglas para poder combinar los términos para definir las extensiones de este tipo de vocabulario controlado, esto significa, que estas ontologías contienen información adicional de cómo realizar deducciones sobre los datos, es decir, cómo establecer axiomas que podrán, a su vez, aplicarse en los diferentes dominios que trate el conocimiento almacenado.

Así pues, las ontologías nos permiten definir esquemas comunes para representar conocimiento de un ámbito. Los datos representados haciendo usos de ontologías pueden ser tratados por máquinas, permitiendo el intercambio de datos en la World Wide Web y facilitando la comunicación entre personas.

Los elementos básicos que componen una ontología son los siguientes:

Clases (conceptos) Son las ideas básicas que se intentan formalizar. Cada clase define una categoría, una agrupación de individuos que tienen un marco y elementos comunes entre ellos por los cuales pueden ser reconocidos, diferenciados y clasificados.

Propiedades (roles) Son las características de los conceptos o clases, permiten expresar los atributos de los mismos y establecer relaciones con elementos de otras clases.

Instancias Son los objetos específicos que están determinados por una clase, es decir, los objetos de una clase.

2.1.2. OWL

OWL (*Web Ontology Language*) [13] es un lenguaje para la Web Semántica diseñado a fin de representar conocimiento semántico sobre cosas, grupos de cosas y las relaciones entre ellas. Se basa en XML Schema [19] y RDF Schema [18] y añade más vocabulario para poder definir y enriquecer clases y propiedades, esto facilita la descripción formal del significado de la terminología usada en los documentos.

OWL forma parte del conjunto de tecnologías creadas y apoyadas por el W3C para la Web Semántica.

OWL no solo nos ayuda a representar el conocimiento, sino que está creado tomando como formalismo subyacente los llamados lenguajes de lógicas descriptivas (DL) [8], una familia de lenguajes con distinta expresividad que tienen una base formal detrás que permite razonar con ellos. Para este razonamiento existen los llamados razonadores DL, programas informáticos capaces de verificar la integridad del conocimiento o de hacer explícito el conocimiento implícito.

OWL tiene en cuenta los conceptos expuestos en la Subsección 2.1.1, e incluye métodos para definir formalmente clases, propiedades e instancias. Además incluye también otras funcionalidades que dotan a las ontologías descritas con este lenguaje de mayor expresividad. Entre las características de OWL, las más utilizadas durante el desarrollo del proyecto son las siguientes:

Jerarquía OWL ofrece la posibilidad de construir jerarquías de clases. Por ejemplo, podríamos definir que una Persona es una subclase de la clase Mamífero, entonces un razonador podría deducir que cualquier recurso de tipo Persona es también de tipo Mamífero. Pueden definirse jerarquías tanto en clases como en propiedades.

Thing Los lenguajes de lógica descriptiva definen dos clases especiales, *Top* y *Bottom*, que son respectivamente superclase y subclase de todas las demás en el dominio. Al ser OWL un lenguaje de lógica descriptiva, incluye estos concepto y añade las clases *Thing* y *Nothing* [1]. Siendo *Thing* superclase de todas las clases y por herencia clase de todos individuos o instancias, y *Nothing* la clase vacía. Se podría decir que *Thing* es a OWL lo que *Object* es a Java.

Propiedades OWL permite definir propiedades y, mediante jerarquías, subpropiedades de estas. Las propiedades añaden información sobre los recursos, esta información puede ser tanto una URI como valores de datos; las dos subpropiedades básicas que diferencian estos dos casos son, respectivamente, *ObjectProperty* y *DataProperty*. Además, estas propiedades disponen a su vez de varias propiedades para enriquecerlas, con la posibilidad de definir su dominio, rango, restricciones de cardinalidad y simetría entre otras. Se pueden también describir propiedades que sean transitivas. Cuando una propiedad es transitiva, si el par (x, y) es una instancia de la propiedad transitiva P, y el par (y, z) es otra instancia de la propiedad transitiva P, entonces el par (x, z) también es una instancia de P. Esta transitividad de las propiedades es especialmente relevante en el proyecto, ya que OWL combinado con razonadores DL pueden obtener nuevos hechos utilizando la transitividad de las propiedades, como se verá más adelante.

[1] *Thing* y *Nothing*, *http://www.w3.org/TR/2004/REC-owl-guide-20040210/#DefiningSimpleClasses*

Equivalencia Con OWL es posible definir dos clases o dos propiedades como equivalentes. Los razonadores entenderán esto y si tenemos, por ejemplo, una equivalencia entre las clases Coche y Automóvil, una instancia de Coche será también instancia de Automóvil. Este aspecto de OWL se ha usado al definir la Ontología Dominio, explicada en la Sección 4.2.2.

En la sección B.1.1 del apéndice B puede encontrarse una descripción más detallada del lenguaje OWL, así como en los documentos que aloja el W3C sobre este lenguaje [13].

2.1.3. RDF

RDF responde a *Resource Description Framework*, lenguaje de modelado de conocimiento tomado como estándar por el W3C para la representación de metadatos sobre recursos de la web [12]. Es el lenguaje sobre el cual se apoya OWL para incluir semántica.

Este modelo se basa en la idea de que todo es un recurso. Cada recurso está representado mediante un identificador Web, llamado URI (*Uniform Resource Identifier*) [2], y es descrito mediante propiedades simples y sus valores. El elemento principal de RDF son las tripletas: [**a R b**], que describen hechos. Cada tripleta está formada por *sujeto* (**a**), un *predicado* (**R**) y un *objeto* (**b**), esto permite la representación de hechos en forma de grafo.

Siguiendo un ejemplo para tener una idea más concreta de cómo representa estos hechos el modelo, podríamos decir que la frase "existe una persona llamada Guillermo Esteban cuyo correo electrónico es 529679@unizar.es" contiene una serie de hechos cuya representación en forma de grafo RDF sería la mostrada en la Figura 2.1.

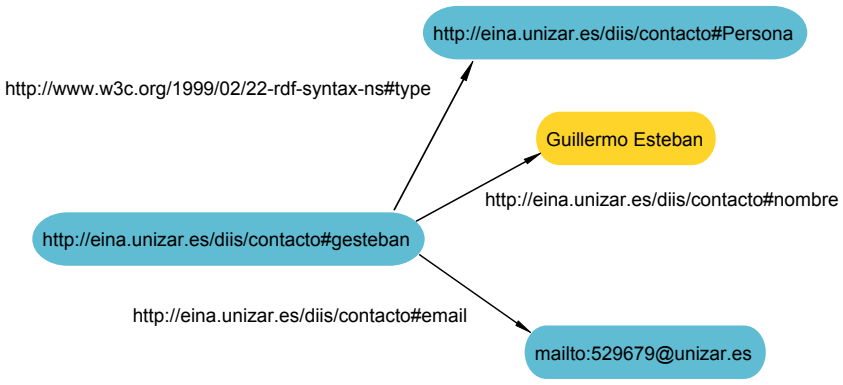


Figura 2.1: Ejemplo de grafo RDF

Como puede observarse en la Figura 2.1, el modelo RDF utiliza URIs para representar:

[2]URI, http://www.ietf.org/rfc/rfc2396.txt

- Clases: en este ejemplo, la clase del recurso sería Persona, identificado por `http://eina.unizar.es/diis/contacto#Persona` y establecido mediante la propiedad *type*, identificada por `http://www.w3c.org/1999/02/22-rdf-syntax-ns#type`.
- Individuos, o instancias: en nuestro caso, Guillermo Esteban, identificado por `http://eina.unizar.es/diis/contacto#gesteban`.
- Propiedades: por ejemplo, el nombre, propiedad identificada por `http://eina.unizar.es/diis/contacto#nombre`.
- Valores de las propiedades: en nuestro ejemplo, el correo electrónico, identificado por `mailto:529679@unizar.es`.

El modelo puede representarse en varios formatos sintácticos, siendo el ofrecido por el mismo estándar RDF y uno de los más utilizados una variación de XML llamada RDF/XML. Siguiendo el mismo ejemplo de la Figura 2.1, la representación en RDF/XML es la siguiente.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:diis="http://eina.unizar.es/diis/contacto#">
      <rdf:Description rdf:about="http://eina.unizar.es/diis/contacto#sperez">
            <rdf:type rdf:resource="http://eina.unizar.es/diis/contacto#Persona"/>
            <diis:nombre>Sergio Pérez</diis:nombre>
            <diis:email rdf:resource="mailto:sperez@unizar.es"/>
      </rdf:Description>
</rdf:RDF>
```

En la representación en RDF/XML es habitual el uso de *namespaces* para referirse a prefijos de URIs que son usadas habitualmente, de esta manera facilitamos su lectura e intercambio.

Para una descripción más profunda, referimos al documento de definición del W3C [12], en el cual se detallan el resto de características del lenguaje.

2.1.4. SKOS

SKOS (*Simple Knowledge Organization System*) [20] es un vocabulario para RDF que nos permite representar sistemas de organización del conocimiento. SKOS adopta un modelo y objetivos distintos a los de OWL, por lo que puede usarse en conjunción con este.

Con RDF y OWL se puede representar conocimiento complejo: podemos definir infinidad de clases, individuales y sus propiedades y tener representado todo ese

conocimiento en OWL para su tratamiento por máquinas. A pesar de la capacidad organizativa de OWL y RDF, tienen limitaciones a la hora de modelar y categorizar: mientras RDF y OWL modelan objetos o conceptos de un dominio, SKOS nos permite definir categorías para cada uno de ellos.

SKOS es un modelo cerrado que hace uso de RDF para definir las clases y propiedades necesarias para categorizar el conocimiento. Con SKOS podremos etiquetar cada recurso para saber a que categoría de conocimiento pertenece. De esta manera se puede determinar con exactitud en que rama de conocimiento nos hallamos, ya que este se encuentra correctamente organizado y etiquetado, pudiendo también utilizar razonadores para inferir nuevos hechos y clasificaciones de nuestros recursos. Entre los elementos que introduce para ello cabe destacar los siguientes:

Concepto El elemento fundamental del vocabulario SKOS es la **clase** concepto [3]. Un concepto se define como una unidad de pensamiento y es una entidad abstracta independiente del término usado para definirla. Con este elemento se pretende que las ontologías que dan uso a SKOS categoricen los recursos y los agrupen según estos conceptos.

Etiquetas El primer paso para categorizar el conocimiento es aplicar etiquetas a los conceptos. Para poder darles nombre, se les relaciona con un valor en lenguaje natural utilizando alguna de las **propiedades** de etiquetado que SKOS ofrece.

Relaciones Los conceptos comúnmente presentan relaciones semánticas entre sí, estas relaciones pueden ser jerárquicas o no y son, como las etiquetas, **propiedades** RDF. En este proyecto, por su aparición en los datos presentes en la DBPedia, se han utilizado exclusivamente las relaciones jerárquicas; estas relaciones son *broader* y *narrower*, atendiendo a conceptos más generales y más específicos respectivamente. Por ejemplo, el concepto "Mecánica" sería *broader than* "Mecánica de Fluidos", y viceversa, "Mecánica de Fluidos" sería *narrower than* "Mecánica".

Para una descripción mas detallada del vocabulario SKOS léase el apéndice B.1.2.

2.2. Accediendo a los datos

Ahora pasará a describirse brevemente los elementos utilizados para el acceso y recuperación de datos, el lenguaje de consultas sobre datos estructurados SPARQL y el repositorio externo utilizado durante la implementación, la DBPedia.

[3] *Concept*, http://www.w3.org/TR/2009/NOTE-skos-primer-20090818/#secconcept

2.2.1. SPARQL

SPARQL [23] es el acrónimo recursivo de *SPARQL Protocol And RDF Query Language*, y es usado para realizar consultas a uno o varios repositorios RDF. SPARQL es otra recomendación del W3C para la Web Semántica y se le considera el sucesor semántico de SQL.

SPARQL dispone de 4 maneras básicas de realizar peticiones al punto de acceso RDF o *endpoint*: DESCRIBE, ASK, CONSTRUCT, y SELECT. El tipo más utilizado durante este proyecto ha sido SELECT, que sigue un formato de consulta parecido al que utiliza SQL, ya que debe formarse con los elementos *SELECT*, *WHERE*, *FROM* y *ORDER BY*. Al igual que cuando se describe un documento RDF, SPARQL ofrece la posibilidad de definir varios *namespaces* para facilitar la redacción y lectura de las consultas.

Para realizar consultas SELECT, primero se definen los datos por los que se está preguntando, esto se realiza en el *SELECT* escribiendo por orden las variables que queremos mostrar como resultado. Inmediatamente después, en la sección de *WHERE*, se escriben los patrones de tripletas, que no son más que tripletas pero con la posibilidad de reemplazar cualquier parte de estas por variables libres. Todas estas variables van precedidas por el carácter especial "**?**". Para ilustrar cómo se realiza una consulta sencilla se define un ejemplo donde se buscarán los nombres de todas los recursos presentes en el repositorio donde se realice la consulta.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
          ?person foaf:name ?name .
}
```

Cada vez que utilizamos una variable en una tripleta, esta se liga con todas las apariciones del repositorio que concuerden con el resto de la tripleta, en este caso, nuestro patrón nos daría todas las tripletas que tuviesen como predicado a la propiedad identificada por la URI `http://xmlns.com/foaf/0.1/name`.

Se puede encontrar más información sobre cómo se realizan las consultas en SPARQL en el apéndice B.3; siendo la presente sección un extracto de este.

2.2.2. DBPedia

La DBPedia [15], como se ha explicado de manera breve en la introducción, extrae la información presente en la Wikipedia y la etiqueta semánticamente gracias a distintos algoritmos que la recorren y realizan la transformación a datos estructurados.

La DBPedia contribuye a la iniciativa Linked Data [1] con más de 3 millones y medio de recursos, y alrededor de mil millones de tripletas que los describen. Esto sitúa al conjunto de datos aportados por este proyecto en la vanguardia de la

publicación de datos para la Web Semántica. Además de la cantidad de datos suministrados, los recursos de la DBPedia también enlazan con otros muchos conjuntos de datos presentes en la Web.

Como punto de acceso a este repositorio se ha utilizado la URL `http://dbpedia.org/sparql`, donde también pueden ejecutarse consultas vía Web.

En la imagen de la Figura 2.2 puede verse el esquema de enlazado presente entre los distintos conjuntos de datos estructurados presentes en la actualidad en la Web.

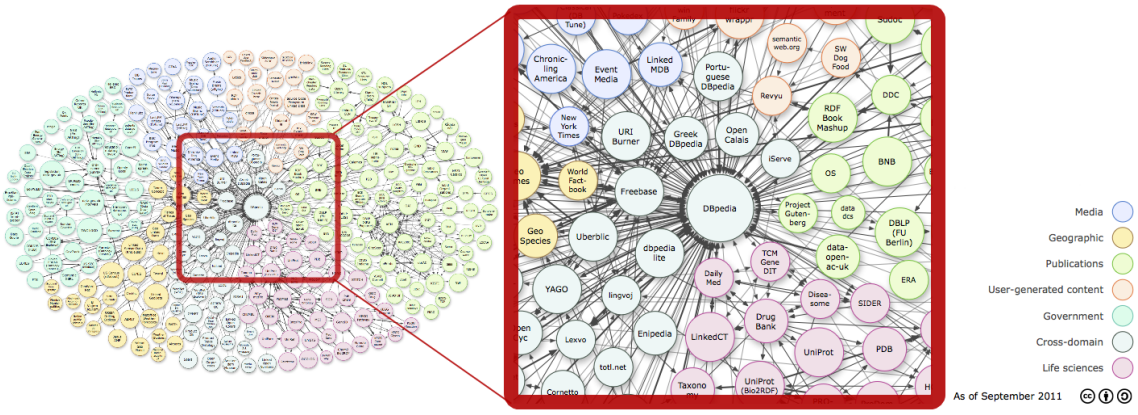


Figura 2.2: Linked Data y DBPedia

Pueden encontrarse los detalles de la estructura que presenta la DBPedia en la sección 4.2.2.

2.3. Implementando la aproximación

Durante la implementación de la aproximación se han utilizado varios lenguajes y herramientas. Como lenguaje de programación ha sido elegido Java, los puntos tenidos en cuenta a la hora de la elección de este lenguaje, así como la lista completa de lenguajes y herramientas utilizados durante la implementación se puede encontrar en el apéndice B. A continuación se expone un subconjunto de este anexo con tres de las herramientas usadas más relevantes para el proyecto.

2.3.1. OWL y SKOS API

Java dispone de numerosas bibliotecas externas de las que se puede hacer uso para añadir funcionalidades. En este sentido tanto OWL como SKOS disponen de API para Java [21, 22], con todos las clases necesarias para utilizar todo el potencial de OWL y SKOS.

Una vez estudiadas las posibilidades de estas herramientas su uso ha sido de gran ayuda a la hora de implementar la aproximación. OWL API ha servido para poder explorar la ontología a bajo nivel, poder diferenciar fácilmente el dominio de búsqueda de la subconjunto extraído de las ontologías del repositorio externo, y realizar inferencias sencillas.

SKOS API hace uso interno de la misma OWL API e incluye el modelo SKOS a sus características, su uso ha servido para crear y almacenar la taxonomía de las categorías de los recursos. Gracias a SKOS API, la recuperación de las categorías de la DBPedia se realiza al inicializar el sistema y queda almacenada en memoria, aliviando al sistema de futuras consultas al repositorio externo para alterar el dominio de búsqueda.

2.3.2. Razonadores DL

Los razonadores son programas informáticos capaces de, partiendo de un conjunto de axiomas o hechos inicialmente presentes, inferir consecuencias lógicas partiendo de un conjunto de reglas. En el caso de este proyecto estas reglas son determinadas por el modelo OWL [13].

Entre las facilidades que ofrece la OWL API se incluye una interfaz para el uso de razonadores DL, lo que ofrece la posibilidad de realizar inferencias en nuestra ontología sin necesidad de implementar un razonador propio dada la presencia de varios de razonadores libres en la Web que implementan esta interfaz. En el caso de este proyecto los razonadores usados han sido HermiT [9] y Pellet [28].

2.3.3. Lucene

Lucene [29] es una librería para la recuperación de información que implementa entre otros un motor de búsqueda de texto. El proyecto Lucene está apoyado por la Apache Software Foundation y dispone de API para varios lenguajes de programación, entre ellos el utilizado durante este proyecto, Java.

Su principal uso durante el proyecto ha sido el de realizar el ordenado por relevancia de los resultados obtenidos de las búsquedas semánticas, aunque su uso también ha sido muy útil como caché de búsquedas.

Capítulo 3

Estado del Arte

Como se ha expuesto anteriormente, el trabajo que se propone en este proyecto incluye realizar búsquedas con contenido semántico sin renegar de las técnicas sintácticas actuales. Hoy en día los usuarios están más acostumbrados a realizar búsquedas basadas en palabras clave, ya que estas son mucho más fáciles de realizar, sin embargo estas no disponen de la expresividad que podrían ofrecernos otros modelos más complejos [3].

Cuando se trata de acceder a datos estructurados utilizando un conjunto de palabras clave, existen diversas aproximaciones, la mayoría de ellas optan por realizar un paso previo, en el cual se determina el significado implícito que el usuario desea de las palabra clave. A partir del cual se construye consulta estructurada [4, 5, 24]. Para realizar este proceso, sin embargo, los métodos actuales requieren de disponer de ontologías muy expresivas [5], o la construcción de grandes grafos a partir de los datos subyacentes [6].

En [24], las palabras clave entrantes son utilizadas en un motor de búsqueda que recoge las entidades (clases, propiedades o individuos) relacionadas con estas palabras, y utiliza una serie de plantillas para construir las consultas partiendo de estas entidades. Sin embargo, para que obtener buenos resultados de este sistema, el usuario debe conocer la estructura de los datos sobre los que se realiza la búsqueda, ya que su motor relaciona las palabras con las entidades de repositorio de datos, haciéndose necesario que al menos uno de los términos de la búsqueda concuerde con una clase de la taxonomía subyacente. Además, las propiedades y clases a usar no pueden ser adaptados para definir nuevos dominios de búsqueda tal y como es posible en nuestro sistema.

Otras aproximaciones, como [6, 25], buscan todos los caminos que se pueden derivar del grafo RDF, hasta una profundidad dada, para generar las consultas. En particular, Q2Semantic [25], en primera instancia enriquece los literales ya presentes en el repositorio RDF para lograr que las palabras clave del usuario encuentren mejor un individuo que les corresponda, y a continuación busca, partiendo de los individuos encontrados, un nexo común a todos ellos, recurso que definirá el centro

de sus resultados. Sin embargo, estas aproximaciones asumen que la fuente de los datos está bajo su control y deben pre-calcular complejos grafos para realizar la traducción de las consultas y, finalmente, acceder a los datos. Estas aproximaciones centran su trabajo al nivel de datos, y no tienen en cuenta la flexibilidad que se puede obtener al adaptar la ontología que los describe tal y como realiza nuestro sistema.

Existen también otros trabajos en el área de las bases de datos que proporcionan una interfaz basada en palabras clave para las bases de datos [30, 31, 32], las cuales traducen el conjunto de palabras clave a consultas SQL. Sin embargo, la mayoría de estos trabajos se basan en el conocimiento obtenido de aplicar técnicas de recuperación de información sintácticas, por lo que no consideran el conocimiento estructural de los datos. Estas soluciones también necesitan tener los datos bajo su control, lo que limita su aplicación para escenarios abiertos como el propuesto por Linked Data.

Capítulo 4

Desarrollo de la solución

4.1. Análisis de requisitos

Una vez estudiada la naturaleza del problema y las aproximaciones existentes se procedió a realizar el análisis de requisitos según el estándar 830-1988 IEEE [7].

A continuación se muestran los requerimientos funcionales y de interfaces externas al sistema, extracto del documento de Especificación de Requisitos Software, presente en el Apéndice A.

Requerimientos funcionales

RF-1 El sistema deberá disponer de, al menos, dos tipos de búsquedas de contenido semántico a la DBPedia, búsquedas por palabras clave y búsquedas de refinado. La entrada y salida para realizar estas búsquedas deberán ser lo más sencillas posibles y abstraer de las tecnologías semántico-sintácticas usadas.

RF-2 Las búsquedas por palabras clave deberán venir ordenadas. El orden establecido será el número de apariciones de la palabra en relación a lo extenso del texto.

RF-3 Las búsquedas deberán adaptarse a la Ontología Dominio, ontología que marca las clases y relaciones o utilizar durante las consultas.

Interfaces externas

IE-1 El sistema dispondrá de una interfaz con la que comunicarse con otros equipos.

IE-2 La interfaz contendrá, al menos, dos tipos de métodos principales que podrán ser invocados, la búsqueda por palabras clave y la búsqueda por refinado, además de otros métodos auxiliares necesarios para la interfaz gráfica.

4.2. Nuestra solución

En esta sección se detallan los aspectos relativos a la solución diseñada, esta comprende: la arquitectura del sistema implementado, la comprensión de los datos estructurados presentes en la DBPedia para la definición de la Ontología Dominio, y la búsqueda final de información en el repositorio externo de datos tanto por palabras clave como de refinado por URI.

4.2.1. Arquitectura del sistema

Para comprender cómo nuestro sistema explota la información semántica presente en el repositorio Linked Data externo, se detallará la arquitectura utilizada, que sigue el esquema de la Figura 4.1.

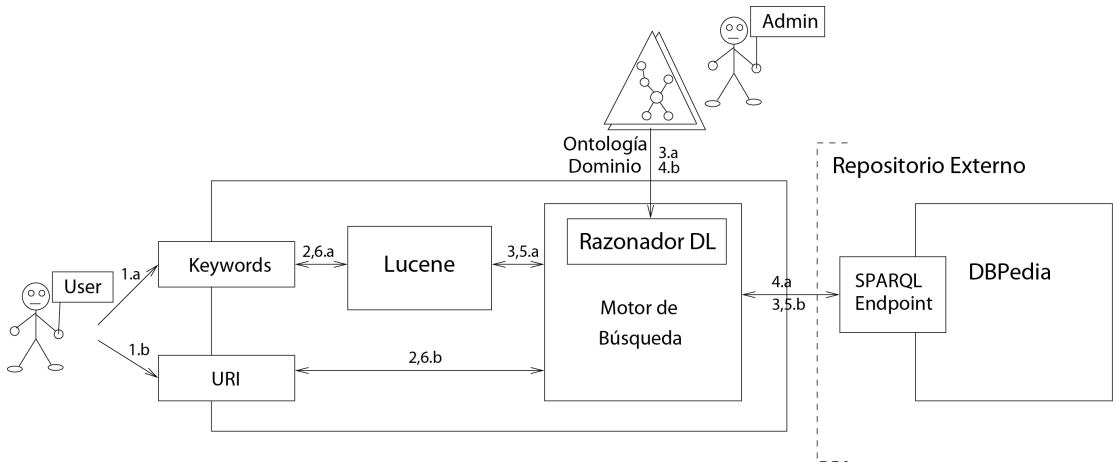


Figura 4.1: Arquitectura del sistema

Inicialmente el sistema requiere una fase de definición del dominio de las búsquedas, el cual se describe en una ontología llamada Ontología Dominio. El dominio de búsqueda definido será usado en la creación de consultas y describirá las clases y propiedades que se obtendrán como resultados. Además, contendrá un subconjunto de la categorización SKOS presente en la DBPedia para guiar el primer tipo de búsqueda. Para tratar y extraer datos de la Ontología Dominio, nuestro sistema hace uso de un razonador de Lógica Descriptiva [8], en nuestro caso HermiT [9] o Pellet [28].

Una vez esté definida la Ontología Dominio, nuestro sistema dispone de dos métodos de búsqueda diferentes y complementarios, dependientes de la entrada que proporcione el usuario.

a) *Búsquedas basadas en palabras clave.* Este servicio de búsqueda toma como entrada un conjunto de palabras clave (paso 1.a), para primero realizar una primera

búsqueda en el directorio de Lucene (paso 2.a). Este directorio intermedio nos sirve como caché, tanto para mejorar los tiempos de respuesta del sistema, como para aliviar la carga de trabajo que ha de soportar el punto de acceso público del repositorio de datos estructurados (en nuestro caso la DBPedia, que no está bajo nuestro control y puede tener problemas de disponibilidad). Si la búsqueda aún no se ha realizado para las palabras clave entrantes y falla en caché, se envía al Motor de Búsqueda (paso 3.a), el cual, con los datos de la búsqueda y la taxonomía de la Ontología Dominio construye una consulta SPARQL para su lanzamiento sobre el punto de acceso del repositorio. La taxonomía de la que se hace uso para construir la consulta incluye la información sobre los objetos sobre los que queremos realizar la búsqueda, limitando a estos el dominio de la resultados. Cuando los resultados son devueltos (paso 4.a), son almacenados en el directorio Lucene para futuras búsquedas (paso 5.a). Este repositorio Lucene, aparte de servirnos como caché, nos posibilita ordenar los resultados según la relevancia que las palabras clave tengan en cada resultado. Finalmente, los resultados, un conjunto de URIs y los campos de estas elegidos en la Ontología Dominio, son devueltos ordenadas por relevancia (paso 6.a).

b) *Búsquedas de refinado de URI.* Los resultados devueltos por el anterior tipo de búsqueda es un conjunto ordenado de URIs, las cuales son presentadas al usuario para que este pueda explorarlos y analizarlos. En este punto, el usuario puede realizar una búsqueda basada en una de estas URIs. Cuando la envía al servicio (paso 1.b), este pasa la URI recibida directamente al Motor de Búsqueda (paso 2.b). Seguidamente el motor realiza una primera consulta al repositorio de datos estructurados para obtener el tipo de objeto que hay detrás de la URI (paso 3.b). A continuación, busca la definición del tipo en la Ontología Dominio (paso 4.b) para construir una consulta especializada para ese tipo de objeto; esta definición contiene las propiedades[1] necesarias para extraer los datos relevantes y otros objetos relacionados con la URI dada. Este proceso se realiza utilizando la Ontología Dominio junto con un razonador DL. Seguidamente se lanza la consulta creada al punto de acceso (paso 5.b) para terminar devolviendo los resultados obtenidos al usuario (paso 6.b). Los resultados de este tipo de búsquedas no son cacheadas ya que sus consultas son mucho más especializadas y no consumen tanto tiempo como las búsquedas por palabras clave, que son lanzadas sobre todo el dominio.

Téngase en cuenta que el directorio Lucene solo es usado para cachear y rankear los resultados que se han obtenido del repositorio de datos externo. El tipo de ordenación que se realiza es por aparición de las palabras clave en los campos designados (estos campos son definidos al crear la Ontología Dominio, de la que se hablará

[1] Estas propiedades son marcadas como relevantes durante la definición de la Ontología Dominio, por lo que el Motor de Búsqueda las conoce y puede crear una búsqueda especializada para cada tipo de objeto.

en posteriores secciones), priorizando los resultados por mayor número de apariciones en relación a lo largo de los campos en los que se busca; este criterio puede modificarse si para alguna aplicación en concreto de este sistema fuese necesario.

4.2.2. Definiendo el dominio de búsqueda

En la presente sección se explicará, primero, como definimos el dominio de búsqueda para los servicios que ofrece nuestra solución creando la Ontología Dominio, y luego como aplicamos este sistema al caso particular de la DBPedia.

Anotaciones en la Ontología Dominio

El primer paso realizado a la hora de desplegar el sistema propuesto es la definición de la Ontología Dominio, que define el dominio de búsqueda que se quiere presentar a los usuarios. Esta ontología definirá sobre que dominio de conocimiento se lanzarán las consultas al repositorio de datos, lo que influirá determinadamente en los resultados de las búsquedas que el sistema devuelva.

Los repositorios de datos estructurados siguen la estructura de una o varias ontologías, etiquetando los recursos que pueblan sus conjuntos de datos según las clases y propiedades de las que disponen. La Ontología Dominio que se propone no difiere de otras ontologías en más que, mientras las ontologías utilizadas en un repositorio definen efectivamente la estructura de los datos almacenados, nuestra ontología describe la vista que queremos dar del conjunto de datos externos, esto es, la estructura de estos, un subconjunto de la original etiquetada y adaptada al uso en nuestro sistema. Así, la Ontología Dominio proporciona una vista de los datos que se van a recuperar, por lo cual debe estar alineada con la ontología usada por el repositorio de datos externo. Alineada se refiere a que deben existir referencias a la ontología externa dentro de la propia, referencias vinculadas a las entidades que se definen para nuestro sistema. Esto supone el estudio del conjunto de datos disponible para conocer las clases y propiedades más relevantes e interesantes en relación al uso que se le quiera dar al sistema. Este paso debe ser realizado por un experto y proporciona al sistema la visión de un subconjunto de datos de entre todos los presentes, lo que especializa nuestra búsqueda.

Aunque la Ontología Dominio podría ser construida desde cero utilizando las clases y propiedades presentes en los datos a recuperar, se ha optado por utilizar técnicas de extracción de ontologías [14] para obtener un módulo de la ontología a la que los datos se acoplan. Una vez obtenido el módulo/subontología con el conjunto relevante elegido del repositorio, se trabaja sobre ella para definir el dominio de búsqueda, primero creando las clases y propiedades internas que se desean utilizar y luego formando los enlaces necesarios para alinear las entidades externas y las internas. De esta manera se obtiene la alineación mencionada, por un lado se tienen

en la Ontología Dominio las clases y propiedades externas, es decir, las que han sido extraídas de la ontología usada por el repositorio, y por otro las clases y propiedades internas que definen los resultados. Este proceso se realiza para mantener la ontología extraída inalterada, de manera que se puedan alterar las clases y propiedades libremente sin por ello tener que cambiar las originales.

Para este proceso de definición de la Ontología Dominio se tienen en cuenta los siguientes cinco aspectos:

- La **taxonomía** que se define en la Ontología Dominio contiene los objetos que serán considerados por el sistema de búsqueda. Los objetos extraídos por las utilidades de extracción pueden contener entidades que no aporten nada al sistema a implantar, o que se les quiera dar un uso específico, para ello las clases y propiedades análogas a las extraídas pueden ser relacionadas con las originales según convenga. Por ejemplo, si la ontología original poseyera las clases `Coche`, `Motocicleta` y `Camion`, pero nosotros no deseamos mostrar tanto detalle de estos medios de transporte, se podría optar por la creación de una clase `VehiculoMotorizado` que englobase las tres anteriores clases. Para esto se crean clases y propiedades análogas a las extraídas y se enlazan a sus respectivas entidades con OWL *equivalentProperty* [2] y *equivalentClass* [3].

- Las ***ObjectProperty*** (propiedades cuyo rango es una clase) juegan un papel crucial en el sistema, ya que son utilizadas en los pasos de refinado de resultados para proponer resultados relevantes. Aunque las *ObjectProperty* extraídas de la ontología original ya relacionan las clases originales y se podría hacer uso de ellas, el sistema puede adaptarse a las necesidades requeridas por el tipo de búsquedas que se quieran realizar, alterando las propiedades para un uso específico. Por ejemplo, la ontología extraída podría poseer la propiedad `fechaPublicacion` con dominio `Libro`, `Revista` y `CD_Musica`; si para el caso de uso del sistema la propiedad `fechaPublicacion` de las instancias de `CD_Musica` fuese innecesaria, debería optarse por no incluir tal relación en la taxonomía de la Ontología Dominio e incluir solo aquellas que se deseen.

- El **idioma** juega un papel importante cuando se piensa en un sistema que pueda ser fácilmente adaptable a las distintas situaciones. En este sentido la Ontología Dominio también incluye esta información mediante la anotación [4] *@queryLanguage*, necesaria identificar el idioma de las búsquedas. Esta anotación se utiliza tanto en cada una de las *DataProperties* (propiedades cuyo rango es un tipo de dato), para definir en que idioma se buscarán los datos, como anotación general de ontología, para su uso en la búsqueda por palabras clave, donde define el idioma de las palabras clave.

[2] *owl:equivalentProperty*, owl:equivalentProperty
[3] *owl:equivalentClass*, http://www.w3.org/TR/owl-ref/#equivalentClass-def
[4] *Annotations*, http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/#Annotations

- También es necesario describir los **campos a recuperar** del repositorio. Esto significa marcar con anotaciones aquellas propiedades cuyos valores se quieran obtener en los resultados de las búsquedas. La anotación utilizada para este propósito es *@dataRetrievable*, y se puede utilizar tanto en *ObjectProperties* como en *DataProperties*. Su uso dentro de las *DataProperties* es claro ya que estas contienen datos en diferentes formatos con información sobre el recurso. Las *ObjectProperties* por otro lado, aunque su dominio y rango sean instancias de clase, es decir URIs, algunos de los datos que contienen siguen siendo útiles para el usuario o la misma interfaz que de uso al sistema, como por ejemplo las URIs de páginas web externas con más información sobre el recurso.

- Por último, es necesario definir los **campos de búsqueda**, esto es, las propiedades de los objetos que se analizarán cuando se realicen búsquedas por palabras clave. Estas propiedades son *DataProperty* y se realiza su marcado haciendo uso de la anotación *@kwdSearchable*. Las propiedades así etiquetadas son las únicas que se tienen en consideración a la hora de buscar las palabras clave.

El Motor de Búsqueda explota esta información para dirigir las búsquedas sobre el conjunto de datos estructurados. Esto supone que mediante la adaptación de esta Ontología Dominio podamos cambiar el repositorio de datos y seguir ofreciendo la misma vista de los datos que queremos ofrecer, o utilizar una visión alternativa de la información sobre un mismo repositorio; todo depende de como sea construida esta ontología.

Estructura de la DBPedia

En esta subsección se explicará la organización interna de los recursos de la DBPedia y como nuestro sistema hace uso de ella.

Los artículos presentes en la Wikipedia normalmente son de texto libre, sin embargo, también se pueden encontrar en ellos distintos tipos de información estructurada. Entre la información que podemos encontrar destaca las dos siguientes formas:

- Los ***Infoboxes***[5], plantillas para la publicación de información sobre un objeto para dar una visión general de este. Para ello se utilizan pares etiqueta/valor que informan sobre un aspecto concreto del artículo que se está visualizando. Por ejemplo, el *Infobox* presente en el artículo sobre la ciudad de Zaragoza contiene, entre otros, el par `[Elevation, 199m]`, que informa de la altitud de la ciudad.

[5] *Wikipedia Infoboxes*, http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Infoboxes

- Las **categorías**[6] de los artículos. La mayoría de los artículos de la Wikipedia están categorizados según el ámbito de conocimiento al que pertenece.

Sin embargo, la naturaleza de los datos estructurados que ofrece la Wikipedia en relación a los que se esperan de la Web Semántica no solo difieren en estructura y formato. Al extraer los artículos presentes en la Wikipedia y almacenarlos como datos estructurados en la DBPedia existe una divergencia entre la riqueza semántica que uno y otro sistema pueden mostrar, pues el nuevo modelo semántico es capaz de aumentar la descripción presente en los artículos de la web.

Hay que tener en cuenta además que muchos de los editores de artículos de la Wikipedia no siguen estrictamente las recomendaciones de la plataforma, y no todos los artículos hacen uso de las plantillas que se pone a su disposición. Por este motivo las propiedades y valores de estas se pueden encontrar en diferentes formatos y unidades de medida. Esto ha de ser tenido en cuenta al analizar los datos del repositorio para la creación de la Ontología Dominio.

Cuando los artículos son extraídos de la Wikipedia, una vez en la DBPedia, se les conoce como **recursos**. Cada recurso está representado por una URI y está directamente relacionado con el artículo original de la DBPedia (no solamente por haberse extraído la información de este, la DBPedia hace uso de la propiedad *isPrimaryTopicOf* [7] para enlazar ambos objetos). Además, cada recurso hereda la categorización de la Wikipedia. La taxonomía completa de los artículos de la DBPedia está incluida como SKOS en la ontología de la DBPedia; así, la DBPedia nos ofrece una primera visión de los recursos con relación a su categoría.

Dependiendo del contenido de cada artículo, el correspondiente recurso en la DBPedia puede representar también un objeto. Este tipo de clasificación se realiza mediante la clases de OWL, pudiendo ser cada recurso instanciado a una clase perteneciente a alguna de las ontologías usadas por la DBPedia; por lo general las dos ontologías más utilizadas e importantes para esta categorización de los recursos son la DBPedia Ontology [8] y YAGO [9]. De esta manera la DBPedia ofrece, independientemente de la categorización de los artículos antes mencionada, una segunda visión de los recursos disponibles basada en la naturaleza de estos. Desafortunadamente, esta vista de los datos no está presente en todos y cada uno de los recursos disponibles. Existen recursos que, a pesar de disponer una categorización según el contenido de su correspondiente artículo, no se encuentra instanciado a ninguna de las clases usadas, por lo que carecen de la clasificación por objetos que las ontologías nos ofrecen. A estos objectos los identificamos como Artículos dentro de nuestra Ontología Dominio. En la Figura 4.2 puede verse un extracto del conjunto de datos de la DBPedia y se ilustra el caso mencionado.

[6] *Wikipedia Categories*, http://en.wikipedia.org/wiki/Portal:Contents/Categories
[7] *foaf:isPrimaryTopicOf*, `http://xmlns.com/foaf/spec/#term_isPrimaryTopicOf`
[8] *The DBPedia Ontology*, http://wiki.dbpedia.org/Ontology
[9] *YAGO Ontology*, http://www.mpi-inf.mpg.de/yago-naga/yago/

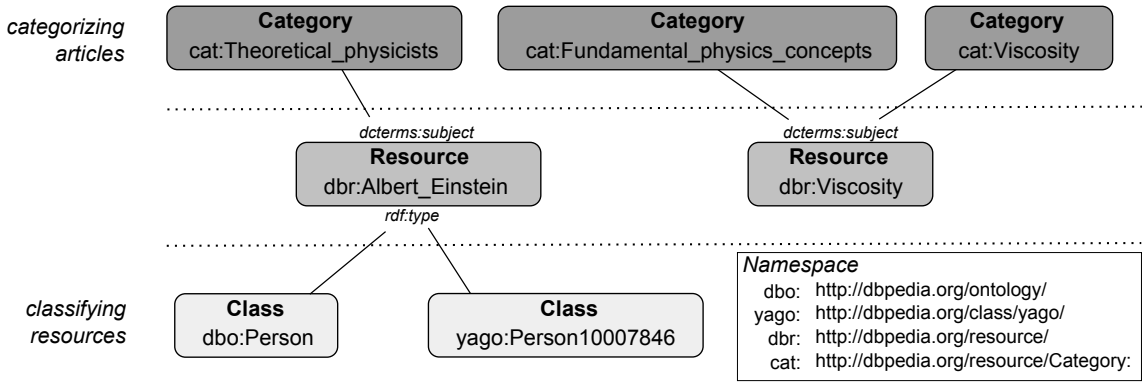


Figura 4.2: Extracto de la DBPedia con las descripciones de los recursos sobre *Albert Einstein* y la *Viscosidad*

Resumiendo, la DBPedia organiza el conocimiento mayoritariamente de dos maneras: la categorización SKOS y la clasificación ontológica. Durante este proyecto, al haberse trabajado con la DBPedia como repositorio de datos, se ha utilizado la categorización SKOS como taxonomía general para las búsquedas por palabras clave; el uso de esta clasificación reduce el conjunto de datos sobre el que se consulta, focalizando las búsquedas de los usuarios y mejorando los tiempos de respuesta cuando se trabaja bajo puntos de acceso público, como ha sido el caso. Por otra parte, el sistema obtiene las definiciones ontológicas de la DBPedia Ontology para su uso en las búsquedas por refinado. Esto se realiza para independizar el proceso de definición del dominio de búsqueda del actual proceso de búsqueda, lo que dota al sistema de mayor flexibilidad. La Ontología Dominio puede ser modificada para mostrar distintas vistas de los mismos datos de forma transparente al usuario; este, en última instancia, solo tendrá que proporcionar la URI del recurso del cual desea más información, siendo el motor de búsqueda junto al uso de razonadores los encargados de construir la consulta, siempre teniendo en cuenta las definiciones presentes en la Ontología Dominio.

En el marco particular de este PFC y por las especificaciones del proyecto Alfa III GAVIOTA, se han recuperado de la categorización SKOS de la DBPedia las categorías inferiores a **Mecánica** [10], ya que el mencionado proyecto solamente requería de conocimiento de tal ámbito. Se ha utilizado la propiedad *abstract* para realizar las búsquedas de palabras clave, ya que nos ofrece un extracto del artículo de texto libre original de la DBPedia. Además, por lo estudiado en los datos del repositorio usado, para las búsquedas de refinado, se han incluido en la Ontología Dominio las clases de las personas, instituciones y artículos (estos últimos son especiales, pues la DBPedia Ontology no los reconoce como clases, siendo los únicos que no tienen clasificación en ese sentido, este particular caso se explicará con detalle en el segundo apartado de la siguiente subsección).

[10] Categoría **Mecánica**, http://dbpedia.org/page/Category:Mechanics

4.2.3. Búsqueda de datos estructurados

Una vez se ha creado la Ontología Dominio, el sistema desarrollado es capaz de realizar dos tipos de búsquedas al repositorio Linked Data externo elegido: búsquedas por palabras clave y búsquedas de refinado de URI. Ambos tipos hacen uso de la ontología previamente creada de diversas formas para enfocar las búsquedas.

Búsqueda con palabras clave

Cuando el usuario desea realizar una consulta con palabras clave, el motor de búsqueda debe construir una consulta SPARQL con los datos proporcionados por el usuario y el dominio de búsqueda definido. Este tipo de búsquedas proporcionará al usuario información sobre Artículos de la DBpedia. Cuando nombramos a Artículos nos referimos a la clase interna definida en la Ontología Dominio para tipar los recursos de la DBpedia que no son instancia de clase, es decir, aquellos recursos que no representan un objeto en sí mismos, sino que representan conocimiento sobre algún tema en particular.

Para ello, haciendo uso de los razonadores DL, el motor deberá preguntar primero a la Ontología Dominio los siguientes puntos:

- Las **propiedades** en las cuales se ha de realizar la búsqueda de palabras clave. Dada la estructura de Linked Data (la cual sigue el modelo RDF basado en tripletas) el sistema debería buscar estas palabras clave en cada uno de los elementos de las tripletas (sujeto, predicado y objeto) si no se especifica las propiedades en las que debemos buscar. Esto conllevaría tiempos de procesamiento para las consultas inasumibles ya que el sistema implementado trabaja sobre un punto de acceso público que no está bajo nuestro control.

- La **taxonomía** de los objetos que se están buscando. Para explotar de manera óptima los datos estructurados que ofrece Linked Data el sistema enfoca la búsqueda a aquellas clases y propiedades que se han marcado anteriormente en la Ontología Dominio. El motor de búsqueda hace uso de estos datos y crea consultas adaptadas al dominio de búsqueda definido.

Con esta información, el motor de búsqueda es capaz de construir consultas SPARQL para ser lanzadas contra el punto de acceso público del repositorio de datos elegido. Para realizar el debido filtrado en las consultas SPARQL, el motor de búsqueda hace uso de la función *regex* [11], que busca las cadenas dadas en los valores de las propiedades elegidas. En el caso concreto de este proyecto, la estructura de las consultas SPARQL para la búsqueda por palabras clave sigue el esquema de la Figura 4.1.

[11] Función *regex*, http://www.w3.org/TR/rdf-sparql-query/#funcex-regex

SELECT	?uri ?abstract ?web
WHERE	?uri dbo:abstract ?abstract
	?uri foaf:page ?web
FILTER	regex(?abstract, [**keyword**])

Tabla 4.1: Estructura básica de consulta SPARQL para palabras clave

Esta consulta, en primera instancia, recoge todos los recursos que tienen las propiedades *abstract* y *page* enlazadas a ellos. A continuación filtra los resultados haciendo uso de la palabra clave proporcionada, buscándola en la propiedad elegida, en nuestro caso *abstract*.

Una consulta como esta, lanzada contra todo el conjunto de datos que la DBPedia nos ofrece, tardaría demasiado tiempo, para mitigar este problema, el sistema hace uso de la taxonomía para limitar los recursos candidatos sobre los que se lanzará la consulta, forzando a que solo se realice con las instancias de los objetos que se desee. Consultar sobre un conjunto de datos menor, además de mejorar los tiempos de respuesta, al estar limitados por ámbito de conocimiento, enfoca la búsqueda y ofrece mejores resultados, acordes al dominio de búsqueda definido.

Las propiedades que definen una taxonomía dependen del lenguaje de modelado que se ha utilizado para expresar la ontología. En OWL, la principal propiedad para definir la taxonomía es *subclassOf* [12], mientras que sobre SKOS esta se modela con las propiedades *broader* y *narrower*. Entonces, dependiendo del lenguaje de modelado el sistema se adapta para definir los límites en el ámbito de las consultas. Siguiendo al anterior esquema de consulta SPARQL que el sistema utiliza, para enfocar la consulta a partir de las categorías SKOS, se deben limitar las tripletas añadiendo las líneas SPARQL de la Figura 4.2.

WHERE	?subCategories	skos:broader	[**chosenCategory**]
	?uri	dcterms:subject	[**chosenCategory**]
	?uri	dcterms:subject	?subCategories

Tabla 4.2: Filtrado por categorías en SPARQL

Estas lineas reducen el conjunto de recursos que se han de comprobar con la función regex a únicamente aquellas que están en el dominio de búsqueda deseado, el cual es estrictamente un subconjunto de todo el dominio de la DBPedia.

En la implementación realizada, la categoría que reduce el número de tripletas sobre las que se consultan las palabras clave está en un extracto guardado localmente de la categorización SKOS que posee la DBPedia. Esta sección de las consultas que limita el número de recursos a partir de las categorías de los artículos, dentro de la propia implementación, se encuentra separada de la Ontología Dominio; se ha implementado de esta manera para facilitar la extracción categorías en tiempo de

[12] *owl:subclassOf*, http://www.w3.org/TR/owl-ref/#subClassOf-def

ejecución. La definición de la Ontología Dominio es un proceso complejo, donde se extraen las entidades elegidas de la ontologías usadas por la DBPedia, esto requiere, como se ha expuesto anteriormente, el conocimiento de los datos estructurados presentes y las propiedades de las que se hace uso, por lo que se crea *offline* antes de iniciar el sistema. Sin embargo, la elección de la categoría raíz para las búsquedas con palabras clave es un proceso mucho más sencillo, basta con que el usuario obtenga una lista de las etiquetas [13] de las categorías y cambie la categorías raíz utilizando un método de la interfaz que implemente el sistema, lo que puede hacerse durante la ejecución.

El resultado de la consulta construida es un conjunto de tripletas <**URI**, **propiedad_i, valorDePropiedad_i**>. Estas tuplas son en primer lugar almacenadas en el repositorio local de Lucene junto al par < **palabraClave**, **categoría** > que han conducido a tal resultado. Este par extra de datos que son almacenados localmente con Lucene son necesarios para a) recuperar posteriormente la información asociada a cada búsqueda sin mezclar resultados que puedan incluir las mismas palabras clave viniendo de otras categorías y b) mantener el índice clasificado, de manera que al liberar al repositorio de resultados de la búsqueda de una palabra clave no eliminemos entre ellos datos pertenecientes a otras búsquedas cacheadas.

Nuestro sistema se beneficia del repositorio local Lucene de dos maneras:

- En primer lugar, y ya que el conjunto de recursos obtenidos de las consultas SPARQL no viene ordenado, con Lucene podemos obtener este mismo conjunto de manera ordenada respecto a la frecuencia relativa de las palabras clave.
- Y en segunda instancia, el disponer de un repositorio local nos sirve de cache para futuras búsquedas, aliviando la dependencia que se tiene de los recursos del punto de acceso público (cuyos tiempos de respuesta pueden variar significativamente entre consultas).

Así pues, el sistema implementado proporciona al usuario un conjunto ordenado de recursos semánticamente relacionados a partir de unas palabras clave, para ello se hace uso tanto de la información semántico de estos como de técnicas sintácticas de recuperación de información. Por ejemplo, al realizar una búsqueda por palabras clave con la entrada "fish movement" en nuestro sistema y en la misma Wikipedia (Tabla 4.3), los resultados obtenidos difieren significativamente. Nuestro sistema se centra en el dominio de "Mecánica", tal y como se ha definido en la Ontología Dominio, y solo devuelve resultados pertenecientes a este ámbito del conocimiento. Mientras que la misma búsqueda en la Wikipedia realiza la búsqueda bajo cualquier dominio.

[13] Las etiquetas aquí mencionadas hacen referencia a la propiedad *label* de RDFS (http://www.w3.org/TR/rdf-schema/#ch_label). Esta propiedad contiene una versión de la URI para ser leída por humanos.

Wikipedia	Nuestro sistema
Fish migration	Tripeladism
Lateral line	Fish locomotion
Murray cod	Role of skin in locomotion
Fishing lure	Aquatic locomotion
Bear Island	Dynamical system

Tabla 4.3: Resultados de la búsqueda de "fish movement" en nuestro sistema y la Wikipedia

Se puede observar que los resultados ofrecidos por el sistema implementado son mucho más específicos del dominio definido, siendo más útiles para búsquedas más focalizadas en un dominio concreto. Tanto en este caso concreto, donde se consultaron los datos de la DBPedia 3.6, como en las pruebas del sistema realizadas, se ha optado por utilizar el inglés como idioma para consultar, ya que la cantidad de recursos disponibles cuyos valores de propiedades se encuentran en inglés es mucho mayor que el conjunto de recursos disponibles en otros idiomas. En concreto, de las 1.890 millones de tripletas RDF que componen el conjunto de datos de la DBPedia, 400 millones están extraídas de los artículos ingleses de la Wikipedia, mientras que el resto se reparten entre los otros 111 idiomas utilizados para la recuperación.

Búsqueda de refinado de URI

Los resultados del anterior tipo de búsqueda por palabras clave son un conjunto de recursos semánticos con sus respectivas propiedades, identificados cada uno de ellos por una URI. En las búsquedas de refinado, el usuario selecciona uno de estos recursos para que el sistema le sugiera resultados semánticamente relacionados. Este refinado se realiza a partir de las propiedades que han sido incluidas en el dominio de búsqueda como relevantes, o lo que es lo mismo, las *ObjectProperties* de la Ontología Dominio cuyo dominio o rango incluyen la clase del recurso a refinar.

Inicialmente, el Motor de Búsqueda desconoce el tipo de instancia que encierra la URI, por lo cual consulta al repositorio de datos externo las clases a las que pertenece el recurso de la URI. Este aspecto marca la independencia entre los dos tipos de búsqueda, puesto que aunque entre los resultados de la búsqueda por palabras clave podamos obtener la clase del recurso, el sistema deja abierta la posibilidad de que no se disponga de tal información sobre la URI entrante, como podría ser en caso de haber sido obtenida por otros medios.

Aunque la arquitectura propone el paso 3.b (donde en la búsqueda de refinado se extrae el tipo de la URI del repositorio externo), este puede ser obviado según se realice la implementación de la interfaz. En la descripción de esta arquitectura se ha propuesto el caso más general, que es partir de una URI, dejando el trabajo de averiguar el tipo de objeto detrás de esta al sistema. Sin embargo, si esta URI

proviene de resultados anteriores, en estos mismos podríamos saber ya que tipo de objeto estamos tratando, por lo que enviando al servicio de búsqueda por refinado el par **[URI,tipo]** ahorraría al sistema de una consulta extra. Si se dispone del tipo a priori, es recomendable establecer una interfaz con ese par de entrada ya que, tanto mejora el rendimiento de nuestro sistema, como alivia la carga del punto de acceso al repositorio externo. Este punto se ha implementó finalmente en la interfaz realizada en el sistema.

Una vez obtenidas las clases de la URI, se consulta la ontología para obtener una clase interna (definida por nosotros en la Ontología Dominio) compatible con su definición externa. Cuando el sistema ha recuperado la clase interna de la URI proporcionada puede empezar a buscar los posibles resultados relacionados, esta búsqueda puede tener en cuenta dos aspectos sobre los datos a recuperar:

- La primera es qué buscar. Esto se refiere al conjunto de objetos que son relevantes. La Ontología Dominio ya posee la definición del recurso que queremos refinar, con todas las relaciones que nos llevarán a más recursos relacionados con la URI de entrada. Cuando se realiza una búsqueda de este tipo, al Motor de Búsqueda se le proporciona una clase como entrada. El motor busca en la ontología todas las relaciones que partan de esta y construye la consulta SPARQL para obtener todos los datos relevantes marcados en la Ontología Dominio. Sin embargo, la aproximación diseñada también puede ignorar parcialmente esta definición y buscar solo los recursos de un tipo específico, este tipo es proporcionado por el usuario. En este tipo de búsqueda, en vez de buscar todas las relaciones de la clase a refinar, la consulta se construye partiendo solo de las relaciones que nos llevan de una a otra clase específica, estrechando más el dominio de los resultados.

- El otro aspecto es qué profundidad de búsqueda utilizar. Por los principios de la iniciativa Linked Data, los enlaces presentes en la DBPedia "no terminan", se puede partir de un recurso cualquiera y haciendo uso de sus *ObjectProperties* llegar a otro recurso, que a su vez conducirá a otros relacionados y así sucesivamente. El sistema implementa la posibilidad de realizar búsquedas teniendo en cuenta este factor y añadir una profundidad a la hora de encontrar resultados relacionados. Por ejemplo, si el usuario dispusiese de una URI de tipo Persona y buscase por Artículo(s), ambas clases internas de la Ontología Dominio, debería proporcionar las URIs del recurso y la clase Artículo, además de la profundidad p, para que el sistema buscase los posibles caminos que conducen de una a otra clase con un uso máximo de p propiedades.

Toda las comprobaciones semánticas necesarias para extraer los dominios y rangos de las propiedades hace uso de razonadores DL, de esta manera se pueden detectar posibles búsquedas incoherentes y omitir las consultas SPARQL que no vayan a proporcionar resultados de acuerdo al dominio de búsqueda.

Durante este tipo de búsqueda de refinado de URI no se hace uso del repositorio local Lucene. Se omite este paso ya que 1) la ordenación de los datos según su relevancia no se puede obtener a partir del dominio de búsqueda, y no se puede encontrar una relación de palabras clave que indique que aspectos del recurso a refinar son más relevantes al usuario, y 2) el conjunto de datos sobre los que la consulta opera es más especializado y menos numeroso, por lo que las consultas de este tipo suelen ser más ligeras por lo que son procesadas de manera más rápida.

Los resultados de estas búsquedas proporcionan nuevos recursos a partir de los cuales se puede repetir la búsqueda sucesivamente, teniendo así una navegación por recursos relacionados entre ellos sin abandonar el dominio de búsqueda definido.

Siguiendo el ejemplo de "fish movement", el usuario podría elegir el Artículo "dynamical system" (siendo Artículo clase interna) de entre los resultados dados por el sistema para realizar una búsqueda de refinado por URI. Los resultados que obtendrá del refinado siempre dependerán de como haya sido definida la Ontología Dominio. En este caso, se definen dos propiedades cuyo dominio o rango incluyen el tipo Artículo: **conocidoPor** y **sujetoDe**, que representan respectivamente las personas que son conocidas por el Artículo y las categorías a las que pertenece este. Los resultados que se obtienen de esta búsqueda de refinado se encuentran en la Tabla 4.4. La búsqueda se ha realizado en la versión 3.6 de la DBPedia.

Tipo	**Nombre**
Persona	Krystyna Kuperberg
Persona	Jean-Christophe Yoccoz
Persona	Yakov G. Sinai
Persona	Denis Blackmore
Persona	Bill Parry (mathematician)
Persona	John Guckenheimer
Categoría	Systems
Categoría	Dynamical Systems
Categoría	Systems Theory

Tabla 4.4: Resultados de refinado de búsqueda del Artículo "dynamical system"

4.3. La interfaz

Esta sección describe brevemente las interfaces que utiliza el sistema, el servicio web y la interfaz gráfica.

Aunque este proyecto se enmarque dentro del proyecto Alfa III GAVIOTA, el sistema ofrece casos de uso de los que no se hace uso en tal proyecto. Por ello se ha realizado, aparte del servicio web instalado para la comunicación con la interfaz del

proyecto realizado por Daniel Martínez, una interfaz gráfica para dar uso a todos los servicios implementados.

4.3.1. Servicio web

El sistema expuesto posee la capacidad, como se ha explicado, de realizar dos tipos de búsquedas: por palabras clave y de refinado de URI. Para poder dar uso a este sistema se decidió montarlo sobre un servicio web con los métodos necesarios para poder utilizar adecuadamente estos dos tipos de búsqueda.

El servicio web dispone de varios métodos que se pueden invocar para realizar los dos tipos de búsquedas, así como otros para consultar la categorización SKOS. Los detalles del servicio web implementado se pueden encontrar en la Sección B.4.6 y la descripción de sus método en la Sección D.1.

4.3.2. Interfaz gráfica

Para poder dar uso al sistema y servicio web implementados se ha creado un Applet. Este Applet, escrito en Java y embebido en una página web, utiliza los métodos del servicio web mencionados para realizar las búsquedas que ofrece al usuario. Esta web con el Applet puede encontrarse actualmente en `http://sid.cps.unizar.es/HybridKeywordSearch/`.

El uso de este tipo de interfaz nos proporciona la capacidad de realizar las búsquedas que se ofrecen en el servicio web, pudiendo mostrar simultáneamente documentos HTML relativos a los recursos recuperados. De este modo la web muestra por un lado, la interfaz creada como Applet, donde se insertarán los datos y se realizarán las búsquedas. Por otro lado el usuario podrá escoger por visualizar, o bien la página web del artículo de la Wikipedia del recurso que se desee, o bien la web de su misma URI, web que pone a disposición la DBPedia. Para la comunicación entre el Applet y la página web se ha hecho uso de Javascript.

En la Figura 4.3 se muestra la web diseñada. El manual de usuario pueden encontrarse en el Apéndice D.

Ya que esta utilidad sigue estando en desarrollo para añadir algunos de los puntos mencionados en la Sección 5.2 sobre trabajo futuro, la web está sujeta a cambios, y podrían encontrarse algunas diferencias con respecto a lo expuesto en este documento.

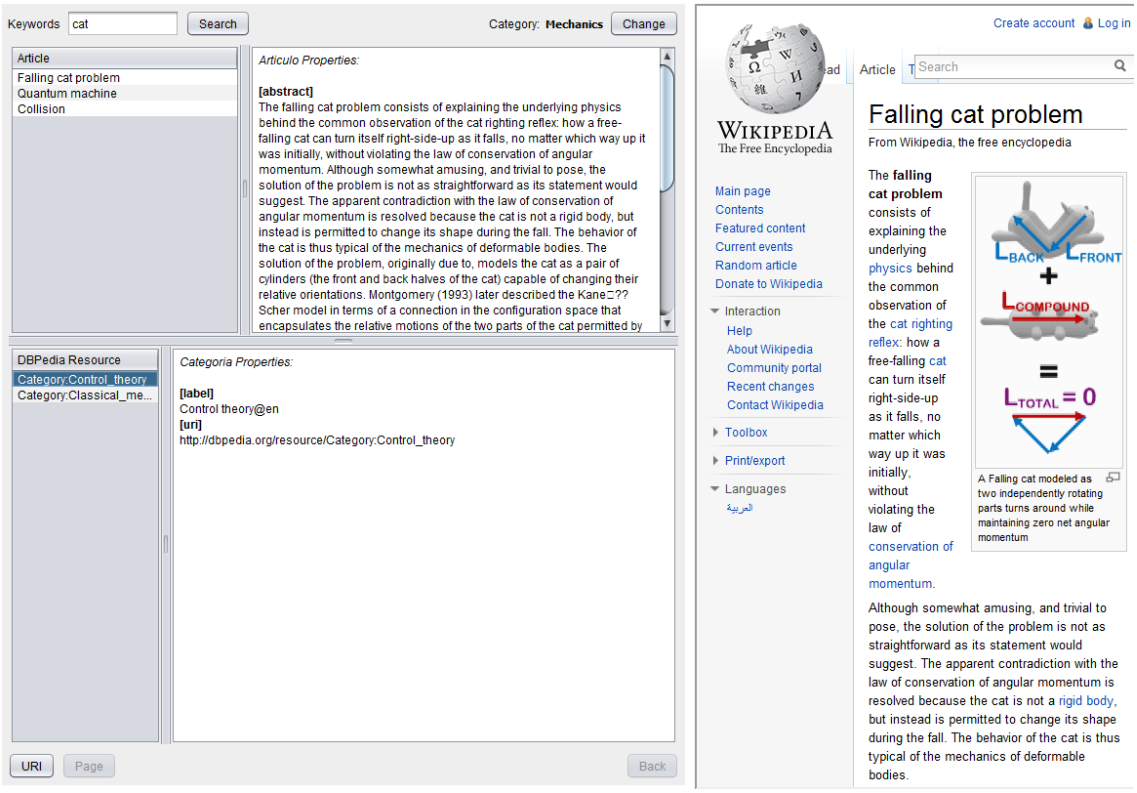


Figura 4.3: Interfaz gráfica

Capítulo 5

Conclusiones

5.1. Conclusiones generales

La conclusión más importante que se obtiene del trabajo en este proyecto es que se ha conseguido realizar la implementación de un sistema híbrido de búsquedas basadas en palabras clave y guiado por ontologías. Este sistema combina la facilidad de uso de las búsquedas por palabras clave con los beneficios de la recuperación de datos estructurados de manera eficiente. En particular, el sistema diseñado e implementado posee las siguientes características:

- Proporciona una búsqueda por palabras clave guiada por la Ontología Dominio, evitando que las consultas realizadas consuman demasiado tiempo. Para este propósito utiliza la información de la taxonomía del dominio de búsqueda. Este proceso es libremente configurable, ya que la búsqueda puede ser restringida según se construya la Ontología Dominio y estén definidas sus clases, propiedades y anotaciones.

- Explota la definición de los objetos del dominio para sugerir resultados relacionados en la búsqueda. Esto se realiza con la ayuda de razonadores DL, que realizan las operaciones necesarias para evitar consultas inconsistentes.

- El sistema puede ser montado sobre repositorios externos Linked Data sin que suponga una sobrecarga para estos, pudiendo proporcionar distintas vistas del conjunto de datos estructurados. El sistema utiliza la Ontología Dominio para proporcionar distintas vistas que también dependerán del repositorio externo elegido.

5.2. Trabajo futuro

A continuación se exponen algunas de las posibles mejores que podría incluir el sistema diseñado:

- Añadir traducción de expresiones DL (definiciones más expresivas en la ontología) para enriquecer el tipo de consultas SPARQL que se pueden realizar. Esto nos permitiría controlar más detalladamente los tipos de objetos/artículos que queremos desde la Ontología Dominio ya que no nos tendríamos que restringir a la relación taxonómica.

- La búsqueda de refinado de URI que el sistema implementa no realiza ninguna ordenación de los datos recuperados. Sin embargo, con el estudio adecuado tanto de los datos como del uso que los usuarios puedan hacer de ellos, podrían atribuirse a las relaciones entre objetos un peso según su relevancia. Estos pesos podrían definir la ordenación de los resultados en las búsquedas de este tipo.

- El uso de Lucene como caché, tal y como se explica durante este documento, esta pensado exclusivamente para las búsquedas por palabras clave. La implementación realizada limita el ámbito sobre el que opera la Ontología Dominio al motor de búsqueda, quedando el uso de Lucene limitado al caso de uso de los artículos sin tipo; solo tiene en cuenta consultas de palabras clave para Artículos; de no ser así y marcando las pautas para futuros trabajos, Lucene podría hacer uso de la Ontología Dominio para realizar el proceso de almacenado e indexado de manera similar a como actúa el motor de búsqueda.

5.3. Conclusiones personales

Durante el tiempo que me ha llevado realizar este proyecto he podido aprender cómo diseñar un sistema que trabaje con datos estructurados y sobre buenas prácticas a la hora de implementarlo y en la redacción de documentos. Sobre todo, he aprendido sobre tecnologías semánticas, he podido apreciar el potencial que encierra la Web Semántica y ha diferenciar los distintos conceptos de este paradigma cada vez más utilizado. Con esto he podido aprender lenguajes que hasta ahora desconocía, como OWL, RDF, SPARQL y afianzar mis conocimientos sobre Java y los servicios web.

Mientras realizaba el proyecto también pude participar en la publicación de un artículo sobre el mismo tema con mi director Carlos Bobed y el profesor Eduardo Mena, esto me ha servido para apreciar la importancia de la labor de investigación y el trabajo en equipo.

En resumen, el proyecto ha hecho que consiguiera adquirir conocimientos interesantes de cara a mi futuro profesional y además desarrollar ciertas habilidades, como dar soluciones a problemas que aparecen en el transcurso de un proyecto real, que a priori no se dan en trabajos desarrollados en la formación de la carrera.

Bibliografía

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009. 1.1, 2.2.2

[2] N. Shadbolt, W. Hall, and T. Berners-Lee. The semantic web revisited. *Intelligent Systems, IEEE*, 21(3):96 –101, jan.-feb. 2006. 1.1

[3] E. Kaufmann and A. Bernstein. Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):377 – 393, 2010. 3

[4] H. Fu and K. Anyanwu. Effectively interpreting keyword queries on rdf databases with a rear view. In *Proc. of 10th Intl. Semantic Web Conference (ISWC'11), Germany*. Springer, 2011. 3

[5] C. Bobed, R. Trillo, E. Mena, and S. Ilarri. From keywords to queries: Discovering the user's intended meaning. In *Proc. of 11th Intl. Conf. on Web Information System Engineering (WISE'10), China*. Springer, 2010. 3

[6] T. Tran, D. M. Herzig, and G. Ladwig. Semsearchpro: Using semantics throughout the search process. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):349 – 364, 2011. 3

[7] IEEE Recommended Practice for Software Requirements Specifications, 4.1. *ANSI/IEEE* Std. 830-1998. 4.1

[8] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Pastel-Scheneider. The Description Logic Handbook. Theory, Implementation and Applications. *Cambridge University Press*, 2003. 2.1.2, 4.2.1

[9] Shearer R, Motik B, Horrocks I. HermiT: a highly-efficient OWL reasoner. *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008): 26-27 October 2008; Karlsruhe, Germany.* 2.3.2, 4.2.1, B.4.3

[10] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. *Technical Report KSL 92-71. Knowledge Acquisition*, 1993. 2.1.1

[11] MJL Lapuente. Tesis Doctoral. Hipertexto: El nuevo concepto de documento en la cultura de la imagen. Dpto. de Biblioteconomía y Documentación. Universidad Complutense de Madrid. 2.1.1

[12] Resource Description Framework (RDF). http://www.w3.org/RDF/. 2.1.3, 2.1.3

[13] OWL 2 Web Ontology Language Document Overview. http://www.w3.org/TR/owl2-overview/. 2.1.2, 2.3.2, B.4.3

[14] E. Jimenez, B. Cuenca, U. Sattler, T. Schneider, and R. Berlanga. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *Proc. of 5th European Semantic Web Conference (ESWC'08), Spain*. Springer, 2008. 4.2.2, C.1

[15] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165, 2009. 2.2.2

[16] C. Bobed, G. Esteban, and E. Mena, Ontology-driven Keyword-based Search on Linked Data, *Proc. of the 16th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES 2012), San Sebastián (Spain)*, IOS Press, volume To appear, September 2012. 1, E

[17] R. Studer, V. R. Benjamins, D. Fensel. Knowledge Engineering: Principles and Methods. *Data & Knowledge Engineering 25 (1998)*. Pages: 161-197. 2.1.1

[18] RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/. 2.1.2

[19] XML Schema 1.1. http://www.w3.org/TR/xmlschema11-1/. http://www.w3.org/TR/xmlschema11-2/. 2.1.2

[20] SKOS, Simple Knowledge Organization System. http://www.w3.org/2004/02/skos/. 2.1.4, B.1.2

[21] Matthew Horridge, Sean Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web Journal 2(1), Special Issue on Semantic Web Tools and Systems, pp. 11-21*, 2011. 2.3.1, B.4.2

[22] SKOS API, http://skosapi.sourceforge.net. 2.3.1, B.4.2

[23] SPARQL, http://www.w3.org/TR/rdf-sparql-query/. 2.2.1, B.3

[24] Y. Lei, V. S. Uren, and E. Motta. SemSearch: A search engine for the semantic web. In *Proc. of 15th Intl. Conf. on Knowledge Engineering and Knowledge Management (EKAW'06), Czech Republic*. Springer, 2006. 3

[25] Q. L. Haofen Wang, Kang Zhang, D. T. Tran, and Y. Yu. Q2semantic: A lightweight keyword interface to semantic search. In *Proc. of 5th European Semantic Web Conference (ESWC'08), Spain*. Springer, 2008. 3

[26] *Knowledge-based and Intelligent Engineering Systems*, http://kes2012.kesinternational.org/. E

[27] Proyecto Alpha III GAVIOTA, http://alfagaviota.info/. 1

[28] E.Sirin, B. Parsia, B. C. Grau , A. Kalyanpur , Y. Katz . Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web 5(2): 51–53*, Junio 2007. 2.3.2, 4.2.1

[29] Lucene search library, http://lucene.apache.org/core/. 2.3.3, B.4.5

[30] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, Parag, and S. Sudarshan. BANKS: Browsing and keyword searching in relational databases. In *Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB'02), China.* Morgan Kauffman, 2002. 3

[31] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. In *Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB'02), China.* Morgan Kauffman, 2002. 3

[32] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *Proc. of 18th Intl. Conf. on Data Engineering (ICDE'02), USA*. IEEE, 2002. 3

[33] ProSÉ, http://krono.act.uji.es/people/Ernesto/safety-ontology-reuse/.

B.4.1, C.1