Compresión de Metadatos Semánticos para la Recuperación Eficiente de Contenidos Audiovisuales

Mario Arias[1], **Oscar Corcho**[2], **Javier D. Fernández**[3,4], **Miguel A. Martínez-Prieto**[3,4], **Mari Carmen Suárez-Figueroa**[2]

[1] Digital Enterprise Research Institute (DERI), National University of Ireland, Galway (Ireland)
mario.arias@deri.org
[2] Ontology Engineering Group (OEG), Universidad Politécnica de Madrid (España)
{ocorcho,mcsuarez}@fi.upm.es
[3] DataWeb Research, Departamento de Informática, Universidad de Valladolid (España)
[4] Departamento de Ciencias de la Computación, Universidad de Chile (Chile)
{jfergar,migumar2}@infor.uva.es

Keywords: Recuperación de información semántica, multimedia, compresión, búsqueda *full-text*, HDT.

Abstract

El crecimiento en la producción de multimedia ha traído consigo la formación de grandes repositorios de contenidos audiovisuales y, a su vez, de grandes colecciones de meta-información sobre los mismos. La representación efectiva de estos metadatos supone un reto por su volumen y por la variedad de información que contienen y cuya extracción da lugar a numerosos problemas de escalabilidad que comprometen el éxito de un buscador multimedia. Este trabajo describe la problemática anterior e introduce un caso de estudio que combina el uso de técnicas avanzadas de compresión y la explotación de tecnologías semánticas como base para el diseño de un motor de búsqueda multimedia. Los resultados obtenidos, en un entorno de experimentación real, muestran cómo nuestra propuesta reduce notablemente el espacio ocupado por las colecciones, soportando la resolución eficiente de un subconjunto de consultas SPARQL y búsqueda *full-text* en espacio comprimido.

1. Motivación

Hoy en día, una cantidad creciente de datos multimedia se produce, procesa y almacena en forma digital. Continuamente consumimos contenidos multimedia en diferentes formatos (texto, audio, vídeo, imagen), en distintos idiomas, y provenientes de diversas fuentes. Los datos de YouTube, recientemente publicados[1], dan una idea de este crecimiento. En este informe se indica que cada minuto se suben a sus repositorios 60 horas de nuevos videos, lo que supone un total mensual que supera la producción conjunta de las tres principales cadenas de televisión (TV) en Estados Unidos durante 60 años. Esta comparación muestra el cambio de escala suscitado por la producción de multimedia en la Web frente a la generación de contenidos asociada con medios de comunicación tradicionales.

[1] www.youtube.com/t/press_statistics

La disponibilidad de grandes cantidades de recursos multimedia implica la necesidad de sistemas de recuperación de información eficientes que faciliten el almacenamiento, la recuperación, y la navegación, no sólo de documentos, sino también de imágenes, audios y vídeos. Las tecnologías comunmente utilizadas en estos sistemas se basan, principalmente, en el análisis y procesamiento del texto que describe los recursos multimedia y que, habitualmente, ha sido generado manualmente por el usuario. En este escenario, el primer aspecto a considerar en el diseño de un buscador multimedia es el *volumen* potencial de los repositorios. Su repercusión se materializa en el tamaño final de la colección de metadatos obtenida al describir los contenidos publicados en el repositorio. Esta (meta) información define las propiedades de los contenidos y, por tanto, establece las "pistas" que seguirá el buscador para decidir si un contenido es relevante ante una determinada consulta.

No obstante, el volumen no es el único aspecto problemático a la hora de representar y consultar estas colecciones. La información utilizada, por ejemplo, para describir un video es diferente a la considerada para un documento de audio, de texto o una imagen. Este problema se acentúa si un único contenido multimedia combina varios de los tipos anteriores, ya que su descripción individual comprende los metadatos de cada uno de sus componentes. Esta *variedad*, en el tipo de los metadatos, implica la necesidad de utilizar modelos de representación lo suficientemente flexibles como para manejar, de forma efectiva, la falta de una estructura estricta en la información.

Este trabajo plantea el desarrollo de la infraestructura subyacente a un motor de búsqueda multimedia construido de acuerdo a (1) el uso de tecnologías semánticas para representar y consultar las colecciones de metadatos y (2) la representación compacta de RDF utilizando el formato HDT (Header-Dictionary-Triples) [1].

Por un lado, la flexibilidad del modelo de grafo de RDF [2] permite afrontar la **variedad** de colecciones de metadatos de manera efectiva, mientras que SPARQL [3] proporciona los mecanismos de consulta básicos para el buscador. Además, el

modelado semántico de las colecciones de metadatos multimedia habilita la publicación y enlazado de dichos metadatos en el escenario de la Web de datos. Este espacio global de conocimiento, auspiciado por iniciativas como Linked Data, cuenta con un número creciente de fuentes de información cuya escalabilidad comienza a verse comprometida por el *volumen* de sus colecciones y la *velocidad* a la que fluye y se consulta su información. Por otro lado, la compresión de datos que proporciona HDT permite afrontar el **volumen** de las colecciones y ofrece una descomposición en componentes lógicos que facilita su indexación. Esta decisión nos proporciona una alta **velocidad** en la resolución de un subconjunto de consultas SPARQL, además de búsqueda *full-text*, todo ello en espacio comprimido.

Este modelo de las *"tres Vs"* (variedad, volumen, velocidad) caracteriza, genéricamente, la problemática de *Big Data* y la habilidad de nuestra propuesta para afrontarlo posiciona este trabajo en un escenario de gran interés actual.

El resto del artículo se organiza como sigue. La Sección 2 introduce los fundamentos de Linked Data, revisa sus dos estándares principales (RDF y SPARQL) y analiza el estado del multimedia dentro de la Web de datos. La Sección 3 describe el formato binario HDT y la Sección 4 presenta nuestra propuesta para su indexación. La Sección 5 estudia el rendimiento de la solución (en espacio de almacenamiento y tiempo de consulta) sobre un entorno de experimentación real. Finalmente, la Sección 6 analiza el alcance de los resultados actuales y perfila los siguientes hitos de nuestra investigación.

2. Linked Data

La cantidad de datos semánticos publicados en la Web ha experimentado un enorme crecimiento en los últimos años gracias, principalmente, al impulso de la iniciativa conocida como **Linked Data**[2]. Esta iniciativa nació con la idea de "enlazar datos" en la Web, transformándola en una "base de datos global" en la que las cosas y los hechos relativos a las mismas están conectados, favoreciendo el razonamiento sobre esta red de información. Aunque el mayor auge se ha producido en el contexto del sector público (ej., los gobiernos del Reino Unido y de Estados Unidos), este enfoque se está extendiendo a otros sectores, entre los que destacan los medios de comunicación (ej., la BBC o el New York Times), el ámbito universitario y científico, infraestructuras o logística, entre otros.

Este nuevo enfoque pretende explotar la Web como un espacio global de información en el que la navegación se realiza siguiendo los enlaces entre los datos en lugar de realizarse a través de los documentos. De este modo, se pasa de una Web basada en documentos (en la que el usuario humano es el destinatario principal) a una **Web de datos** enlazados en la que herramientas software pueden publicar, navegar, interpretar, visualizar y utilizar estos datos de forma automatizada. La publicación y enlazado de los datos se realiza utilizando RDF, y su valor y utilidad es mayor cuanto mayor sea su interconexión.

La creación de *Linked Data* sigue cuatro principios de diseño: (1) utilizar URIs (*Uniform Resource Identifiers*) para identificar los recursos; (2) usar URIs HTTP para su *de-referenciación* en la Web; (3) ofrecer información sobre los recursos usando RDF; (4) incluir enlaces a otras URIs. Cuando los datos se publican como Linked Data, sus propiedades RDF pueden navegarse de forma comparable a como lo hacemos en un navegador Web a través de los hipervínculos: los clientes pueden navegar de un recurso RDF a otro *de-referenciando* las URIs de los recursos relacionados (enlazados).

2.1. *Descripción y Consulta en Linked Data*

RDF [2] plantea un modelo de datos para representar información acerca de *recursos*. Estos recursos definen cualquier tipo de datos, se describen con expresiones RDF y se referencian con URIs. Las *propiedades* definen relaciones o atributos usados para describir un recurso. Finalmente, las *sentencias* asignan un valor a una propiedad de un determinado recurso mediante **triples** *sujeto, predicado y objeto*. Por lo tanto, una colección RDF está formada por un conjunto de triples que dibuja una estructura de grafo dirigido y etiquetado. La Figura 1 (izquierda[3]) ilustra el uso de RDF para modelar un pequeño fragmento de metadatos sobre multimedia. El nodo `m3:t5-21-10` describe un *video* (nótese el triple representado por la transición de la arista `rdf:type` desde el nodo `m3:t5-21-10` a `m3mm:Video`) en formato *MPEG2* y con unas dimensiones de *720 x 576*. Por su parte, el nodo `m3:individual1310128095945` representa un fragmento del video anterior: `(m3:individual1310128095945, media:isFragmentOf, m3:t5-21-10)`, en el que se muestra contenido sobre el recurso `dbpedia:Andres_Iniesta` que representa a *Andrés Iniesta* en DBpedia.

SPARQL Protocol and RDF Query Language [3] es un lenguaje para la consulta de colecciones RDF en base a conjunciones y disyunciones de *patrones de triples* (TP). Estos TPs siguen la estructura de un triple RDF pero permiten que sujeto, predicado u objeto sean variables. Por tanto, un TP coincide con un subgrafo RDF cuando los términos de ese subgrafo pueden sustituir a las variables. En el ejemplo previo, el TP `(m3:individual1310128095945, m3:shows, ?O)` obtiene `dbpedia:Andres_Iniesta` en la variable **?O**.

2.2. *Multimedia en Linked Data*

La última estimación[4] cifra el tamaño de la nube de *Linked Data* en 31 billones de triples procedentes de diferentes dominios. Dichas estadísticas reportan la existencia de 29 colecciones que exponen unos 2 billones de triples sobre multimedia. *BBC Music* que expone triples sobre el contenido musical de la BBC; *Event Media* que incluye descripciones multimedia relacionadas con eventos; o *LinkedMDB* que describe películas y actores, son ejemplos de estas colecciones.

El valor añadido, dentro de la nube de *Linked Data*, tiene que ver con la interconexión existente entre estas coleccio-

[2] `http://linkeddata.org/`

[3] La zona inferior declara los prefijos usados en el ejemplo: `m3` representa la URI `http://www.buscamedia.es/ontologies/M3/#` y los recursos/propiedades que prefija se identifican en su ámbito.

[4] `www4.wiwiss.fu-berlin.de/lodcloud/state/` (Septiembre, 2011)

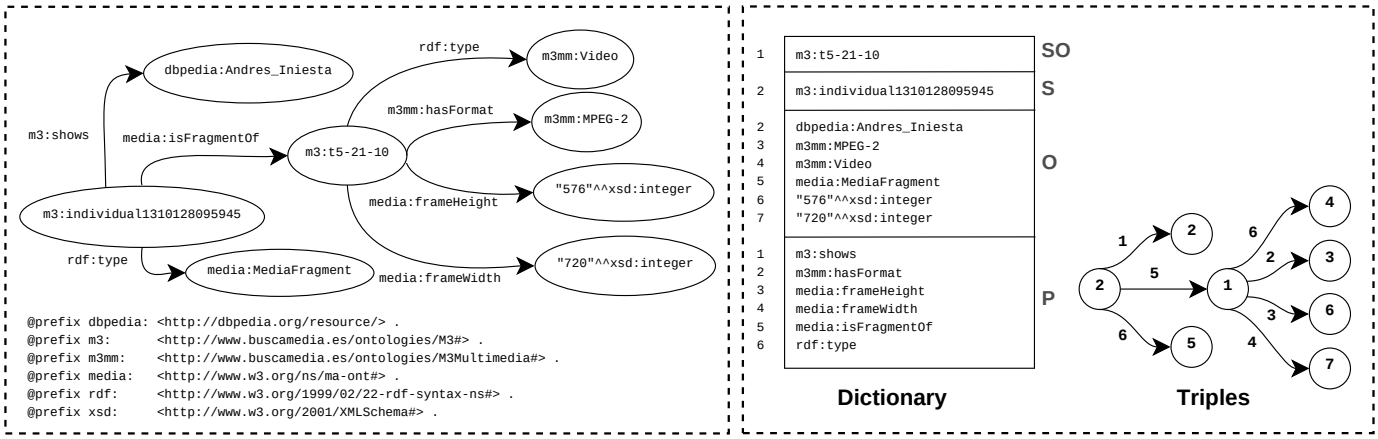

Figura 1. Grafo RDF describiendo metadatos multimedia y su representación en HDT (componentes *Dictionary* y *Triples*).

nes; por ejemplo, las películas y los actores incluidos en LinkedMDB están enlazados con sus descripciones en DBpedia, permitiendo acceder a una información más completa de cada recurso. Sin embargo, un aspecto importante que no está cubierto actualmente es la existencia de descripciones semánticas de grado más fino; por ejemplo, el enlazado entre un actor y sus apariciones en diferentes *frames* de un video [4]. A medida que los usuarios y los proveedores de contenido continúen publicando sus datos multimedia (con distintos grados de granularidad), el número de triples disponibles en la Web de datos se incrementará sustancialmente y con ello los problemas de escalabilidad asociados a su recuperación.

3. Header-Dictionary-Triples (HDT)

El formato binario HDT [1] surge como respuesta a las necesidades de publicación e intercambio de grandes colecciones RDF en la Web de datos[5]. HDT describe un formato de serialización que facilita el procesamiento automatizado de RDF en espacio comprimido. Es importante señalar que los formatos utilizados, tradicionalmente, para este propósito (N3[6], RDF/XML[7] o Turtle[8]) poseen una sintaxis "orientada al humano", dando lugar a representaciones verbosas difícilmente procesables por herramientas software.

HDT modela una colección RDF mediante tres componentes que cumplen un rol bien definido: (i) **Header (H)**: expone los metadatos que describen la colección (procedencia, estadísticas, etc.) y caracteriza la organización física de los otros componentes. Se utiliza como punto de entrada a la colección; (ii) **Dictionary (D)**: cataloga los términos (URIs, *blank nodes* y literales) que etiquetan los nodos y aristas del grafo RDF. Su organización se implementa mediante una función biyectiva que relaciona cada término con un valor entero (ID) que lo identifica en la colección; y (iii) **Triples (T)**: proporciona una representación efectiva del grafo RDF resultante de sustituir cada término por el ID que lo identifica en el diccionario.

[5] www.w3.org/Submission/2011/03/
[6] www.w3.org/DesignIssues/Notation3
[7] www.w3.org/TR/REC-rdf-syntax/
[8] www.w3.org/TeamSubmission/turtle/

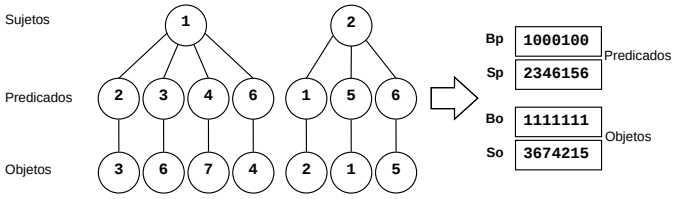

Figura 2. Implementación del componente *Triples*.

La capacidad de HDT para lograr compresión reside en *Dictionary* y *Triples*. La parte derecha de la Figura 1 ilustra su configuración al representar el grafo RDF anterior.

Dictionary. Este componente organiza los términos en cuatro particiones obtenidas de acuerdo al rol de cada término en la colección. La partición **SO** representa todos los términos que actúan como sujeto y objeto: |SO|, y los identifica en el rango [1, |SO|]. En el caso actual, sólo m3:t5-21-10 cumple esta condición y se identifica mediante el valor 1. Por su parte, las particiones **S** y **O** representan, respectivamente, el resto de términos que juegan roles (exclusivos) de sujeto y objeto. Sus identificadores se asignan a partir de |SO|+1. Aunque esta decisión supone el solapamiento de IDs entre sujetos y objetos, su interpretación es trivial considerando que los nodos con un ID común sólo juegan un rol en la colección y este determina la elección de una u otra partición. El ejemplo actual tiene un sólo sujeto: m3:t5-21-10 (identificado como 2) y seis objetos (rango 2-7). Finalmente, la partición **P** representa todos los términos utilizados como predicado: |P|, y los identifica, independientemente (ya que etiquetan aristas), en el rango [1, |P|]. El ejemplo identifica seis predicados en el rango 1-6.

Cada una de estas particiones se representa de forma independiente, de tal forma que puede ser considerada un subdiccionario dentro del componente global. Nótese que las particiones presentan, internamente, un orden lexicográfico que facilita que todos los términos con un prefijo común aparezcan contiguos. Esta propiedad nos permite alcanzar altos niveles de compresión utilizando codificación diferencial para representar cada término respecto a su predecesor inmediato. La implementación propuesta utiliza codificación *Front-Coding* [5] en cada subdiccionario y sigue los fundamentos descritos en [6].

Triples. El componente *Triples* representa la estructura de IDs que modela el grafo RDF. Como puede observarse en la Figura 1, esta estructura es similar al grafo original pero reemplaza los términos por sus IDs correspondientes.

La implementación de este componente requiere un análisis más detallado (ver Figura 2). En la parte izquierda se plantea una representación alternativa del grafo de IDs en la que cada uno de los nodos sujeto se sitúa como raíz de un árbol. Cada uno de estos árboles lista ordenadamente, en el nivel intermedio, los predicados con los que se relaciona el sujeto y contiene, en las hojas, los nodos objeto alcanzables desde el sujeto a través de cada uno de los predicados. Por lo tanto, esta nueva representación modela el grafo como un bosque con tantos árboles como sujetos diferentes se utilicen en la colección.

En la parte derecha de la Figura 2 se muestra la implementación final del componente *Triples*. Como puede observarse, cada uno de los dos niveles almacena una *secuencia de bits* y una *secuencia de enteros*. La secuencia de bits $\mathcal{B}_p$, en el primer nivel, da una representación binaria del número de predicados asociados con cada sujeto. Cada uno de los bits `1` simboliza el inicio de un nuevo sujeto y su cardinalidad se representa por el número de bits `0` que lo suceden antes del siguiente `1`. Por ejemplo, la representación del primer sujeto comienza en la primera posición de $\mathcal{B}_p$ y hay tres bits `0` antes del siguiente bit `1`. Por lo tanto, el primer sujeto se relaciona con cuatro predicados cuyos identificadores se listan, en la secuencia de enteros $\mathcal{S}_p$, a partir de la posición en la que se inicia la representación del sujeto (*predicados 2,3,4,6*). El razonamiento de las secuencias usadas para modelar el nivel de objetos es similar. En este caso, $\mathcal{B}_o$ modela el número de objetos relacionados con un determinado par sujeto-predicado y sus identificadores se localizan en $\mathcal{S}_o$. Por ejemplo, el par formado por el *sujeto 1* y el *predicado 3* (representado en la segunda posición del nivel de predicados) se relaciona con el *objeto 6*, ya que este es el valor alcanzado en $\mathcal{S}_o$ a través del segundo bit `1` en $\mathcal{B}_o$.

4. HDT como Motor de Búsqueda Multimedia

La estructuración interna de HDT permite afrontar el problema de la búsqueda y recuperación de RDF en espacio comprimido como ya se indicaba en su propuesta original [1], dado que permite resolver todos los TPs que fijan el sujeto en una consulta SPARQL. Esta idea ha sido recientemente extendida [7] para resolver eficientemente todos los TPs. Los tiempos de consulta reportados son competitivos en el estado del arte mientras que el espacio utilizado por la nueva representación (denominada **HDT-FoQ**: *HDT Focused on Querying*) contribuye al objetivo de trabajar en espacio (altamente) comprimido.

Esta sección detalla el uso de HDT-FoQ como base de un motor de búsqueda multimedia. Para ello planteamos los mecanismos de indexación usados en los componentes *Dictionary* y *Triples* con el objetivo de proveer una propuesta que integre la resolución eficiente de consultas SPARQL y búsqueda *full-text*.

4.1. Indexación del Componente *Dictionary*

La indexación, propuesta en HDT, sugiere la codificación diferencial de los términos del diccionario. A pesar de conseguir representaciones compactas del componente, esta solución muestra una efectividad desigual a la hora de modelar los diferentes tipos de términos utilizados en una colección RDF.

La codificación diferencial de URIs y *blank nodes* permite obtener representaciones indexadas altamente comprimidas debido a que los términos comparten largos prefijos [8]. Además, este tipo de índices garantizan la resolución eficiente de operaciones básicas de traducción entre términos e IDs. Sin embargo, la diversidad de contenidos que puede darse en un literal complica su representación. En este caso, la codificación diferencial no obtiene buenas tasas de compresión [8] y, adicionalmente, limita la resolución de búsquedas tipo *full-text* que requieren acceso a cualquier fragmento del literal.

Nuestra propuesta revisa la implementación del componente modificando, exclusivamente, la indexación de los literales en la partición de objetos. Para ello construimos, sobre el conjunto de literales, una estructura de auto-indexación (*self-indexing*) conocida como *FM-Index* [9]. El estudio realizado en [8], muestra como el *FM-Index* obtiene representaciones más compactas de los literales a costa de empeorar, ligeramente, la eficiencia de las operaciones entre términos e IDs. No obstante, el uso de esta técnica nos permite resolver búsquedas *full-text* sobre los literales, dada su capacidad para localizar las ocurrencias de una subcadena cualquiera. Esta búsqueda generaliza el algoritmo que obtiene el ID asociado a un término [6]: busca el patrón solicitado y, a continuación, obtiene el ID asociado a cada uno de los literales en los que aparece.

4.2. Indexación del Componente *Triples*

Como se indicaba anteriormente, la propuesta original de HDT permite resolver todos los TPs que suministran el sujeto. La eficiencia de este logro reside en la utilización de unas estructuras de pequeño tamaño que permiten acceder a las secuencias de bits en tiempo $O(1)$ [10] y, con ello, navegar eficientemente los árboles que representan la estructura del grafo.

Por lo tanto, HDT indexa por sujeto el componente *Triples*, pero "abandona" las dimensiones predicado y objeto. Este es el punto de partida de la propuesta HDT-FoQ [7]: añadir a la infraestructura de HDT sendos mecanismos que indexen por predicado y por objeto, dentro de un espacio comprimido.

Indexación por Predicado. El componente *Triples* puede ser indexado por predicado a través de la secuencia $\mathcal{S}_p$. El único requisito es encontrar una representación que permita acceder a las ocurrencias de cada predicado en dicha secuencia.

HDT-FoQ modela $\mathcal{S}_p$ utilizando una estructura sucinta denominada *wavelet tree* [11]. La elección de esta estructura reside en que sus costes están directamente relacionados con el número de valores diferentes que representa y, en el caso actual, la cantidad de predicados utilizados en una colección RDF ($|P|$) tiende a ser un valor muy pequeño. La utilización del *wavelet tree* nos permite localizar cada ocurrencia de un predicado en tiempo $O(log|P|)$, pagando un incremento de $o(n)log|P|$ bits en el espacio ocupado por la nueva representación de $\mathcal{S}_p$.

Indexación por Objeto. A diferencia del caso anterior, la indexación por objeto requiere añadir un tercer nivel a la estruc-

	Original	HDT-FoQ	Virtuoso
event media	187,75	72,59	328,02
yovisto	708,35	156,95	778,02
linkedmdb	850,31	89,30	570,02
freebase	3354,09	302,80	3516,02
dbtune	9733,06	862,23	4220,02
all	14833,56	1555,35	8054,03

Cuadro 1. Espacio de almacenamiento *(en MB)*.

	Q_1	Q_2	Q_3	Q_4
HDT-FoQ	0.18 ms	42.51 s	5.63 s	5.21 s
Virtuoso	2.60 ms	86.54 s	5.90 s	11.95 s

Cuadro 2. Tiempos de consulta.

tura del componente *Triples*. El propósito del nuevo nivel es facilitar la navegación de los árboles desde las hojas (objetos) hacia la raíz (sujetos) pasando por el nivel de predicados.

Este nuevo nivel requiere sendas secuencia de bits ($\mathcal{B}_{oP}$) y enteros ($\mathcal{S}_{oP}$), con configuraciones similares a las explicadas para los niveles superiores. Esto es, $\mathcal{B}_{oP}$ representa el número de pares predicado-sujeto relacionados con cada objeto y $\mathcal{S}_{oP}$ los referencia, ordenadamente, de acuerdo al ID del predicado.

5. Resultados Experimentales

En esta sección estudiamos el rendimiento de nuestra propuesta como base para la construcción de un motor de búsqueda multimedia. Para ello hemos descrito una entorno de experimentación con *dumps* de cinco colecciones publicadas en el área de multimedia de la Web de datos: `event media`[9] (descripciones multimedia sobre eventos), `yovisto`[10] (videos sobre charlas en conferencias y clases magistrales), `linkedmdb`[11] (películas y actores), `freebase`[12] (la partición relativa a TV), y `dbtune`[13] (colecciones musicales).

Adicionalmente, generamos un *mashup* (referido como `all`) con todas estas colecciones. Esta decisión busca estudiar el rendimiento obtenido al modelar una colección que combine diferentes tipos de metadatos. Todas las colecciones se representan utilizando nuestra propuesta basada en HDT-FoQ y sus resultados se comparan respecto a los obtenidos por `Virtuoso`[14]. La elección de este sistema de almacenamiento y consulta de RDF radica en la generalización de su uso y, a su vez, en su capacidad para resolver búsqueda *full-text*.

El Cuadro 1 muestra, en la primera columna, el tamaño original de cada colección (en formato NTriples) y, en las dos siguientes, el espacio que ocupa su representación en cada uno de los sistemas (nótese que para Virtuoso se incluyen los índices necesarios para resolver *full-text*). El espacio ocupado por las representaciones basadas en HDT-FoQ es $\approx 9-11\,\%$ del original para las colecciones más grandes, mientras que en las pequeñas los números aumentan: $22{,}16\,\%$ (`yovisto`) y $38{,}66\,\%$ (`event media`). Este rendimiento hace que la representación del *mashup* `all` use un $10{,}49\,\%$ de su tamaño original, avalando la capacidad de nuestra propuesta para representar de forma efectiva cualquier tipo de metadato audiovisual. Por su parte, los números de Virtuoso muestran como sólo `linkedmdb` y `dbtune` se representan en un espacio menor al original. Indicar que los índices para *full-text* representan, en los casos más costosos un $25\,\%$ del tamaño total de la representación, pero supone sólo un $3\,\%$ para `all`. En estas condiciones, Virtuoso representa el *mashup* en ≈ 5 veces el espacio de HDT-FoQ, dando una idea del ahorro obtenido por un buscador que elija nuestra propuesta frente a un *RDF store* general.

A continuación evaluamos el rendimiento obtenido ante diferentes operaciones de recuperación de contenido. Esta experimentación se ha llevado a cabo en una máquina de procesador AMD Opteron 270 2Ghz, Dual Core y con 4GB de memoria principal DDR2 800Mhz. Nuestra experimentación fija un conjunto de cuatro consultas genéricas para la recuperación de contenidos audiovisuales: a) Q_1 y Q_2 estudian el rendimiento en la resolución de consultas SPARQL, mientras que b) Q_3 y Q_4 añaden el coste de la búsqueda *full-text* empleada para satisfacer el filtrado `regex`:

Q_1:
```
SELECT ?p ?o WHERE
{
 <uri_contenido>?p ?o .
}
```

Q_2:
```
SELECT ?s ?p ?o WHERE
{
 ?s rdf:type <uri_tipo>.
 ?s ?p ?o .
}
```

Q_3:
```
SELECT ?s ?o WHERE
{
 ?s <uri_predicado>?o .
 FILTER regex(?o, "text") .
}
```

Q_4:
```
SELECT ?s ?p ?o WHERE
{
 ?s <uri_predicado>?oo .
 ?s ?p ?o .
 FILTER regex(?oo, "text")
}
```

- **Q_1**: recupera toda la información asociada al *contenido audiovisual* referido como sujeto
- **Q_2**: recupera toda la información asociada a los contenidos de un determinado *tipo*.
- **Q_3**: utiliza un *patrón* (de texto) para recuperar todos los recursos que lo refieren y las referencias completas.
- **Q_4**: utiliza un *patrón* (de texto) para recuperar toda la información asociada con aquellos recursos que lo refieren.

La experimentación[15] se lleva a cabo sobre un conjunto de 50 instancias de cada consulta (de Q_1 utilizamos 125000 para obtener una medición más precisa) y las ejecutamos sobre la colección `all` en cada uno de los sistemas. Con el objetivo de obtener unos resultados más representativos de un escenario de uso real, para cada experimento hemos diseñado una etapa de *warm-up* (con el mismo número de instancias de cada una de las consultas) que le permita a Virtuoso sacar provecho de sus mecanismos de caché. Los tiempos de resolución por instancia (promediados sobre 5 réplicas) se muestran en el Cuadro 2.

HDT-FoQ presenta un amplio margen de ganancia respecto a Virtuoso en las consultas estructurales: Q_1 y Q_2 (nótese que el tiempo de resolución de Q_1 está más de un orden de magnitud por debajo del resto de consultas). La necesidad de búsqueda *full-text* en Q_3 aproxima los tiempos de ambos sistemas, pero en Q_4 la diferencia vuelve a crecer hasta los márgenes de las primeras consultas. La resolución de búsquedas de palabras

[9] www.eurecom.fr/∼troncy/ldtc2010/2010-06-15-N3_Events.zip
[10] www.yovisto.com/labs/dumps/latest.ttl.tar.gz
[11] http://queens.db.toronto.edu/∼oktie/linkedmdb
[12] http://download.freebase.com/datadumps/2012-07-26/
[13] http://km.aifb.kit.edu/projects/btc-2011/
[14] http://www.openlinksw.com/
[15] http://dataweb.infor.uva.es/consultas-lacnem2012.tgz

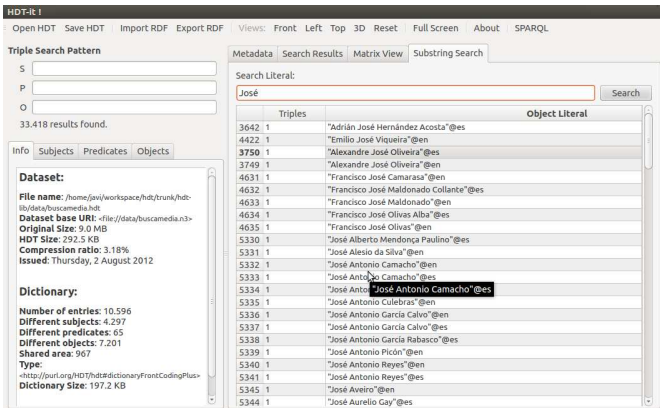

Figura 3. Vista del prototipo de buscador multimedia.

completas (en Q_3 y Q_4) se ha realizado, en Virtuoso, mediante `bif:contains`, cuya eficiencia supera a la de `regex`. Aún así, HDT-FoQ obtiene unos tiempos de consulta más competitivos, complementando nuestros logros en espacio y avalando su eficiencia para su uso en un motor de búsqueda multimedia.

6. Conclusiones y Trabajo Futuro

Este trabajo muestra un caso práctico de gestión y consulta de medatatos multimedia representados con tecnologías semánticas. Hemos introducido el uso de compresión como mecanismo para abordar los problemas de escalabilidad subyacentes al manejo de estas colecciones y hemos demostrado su viabilidad en un entorno de experimentación real en la Web de datos. Este logro demuestra la capacidad de nuestra solución para afrontar, de forma general, el desarrollo de un buscador multimedia, lo que está estrechamente relacionado con el proyecto Buscamedia.

Buscamedia[16] tiene como objetivo principal la creación de un buscador semántico para contenidos multimedia (texto, imagen, audio y vídeo). Este buscador se basa en (a) la utilización de la red de ontologías M3[17], que representa conocimiento multimedia de diferentes dominios y en diferentes idiomas, y (b) técnicas de lenguaje natural que permitan interactuar con el buscador sin la necesidad de aprender una sintaxis de consulta. Adicionalmente, como parte del proyecto, se están generando colecciones de datos conformes a M3. Estas colecciones se refieren a videos de fútbol, baloncesto y Fórmula 1 que han sido emitidos, en castellano o catalán, por distintas cadenas de TV.

La Figura 3 muestra un primer prototipo de buscador semántico construido sobre la propuesta desarrollada en este artículo y aplicada a un subconjunto de los datos generados en Buscamedia. La representación que obtiene de estos metadatos ocupa menos del 5 % de su tamaño original y su funcionalidad permite consulta SPARQL, búsqueda *full-text*, además de ofrecer un mecanismo que navega las relaciones entre los metadatos de forma comparable a cómo se hace en la WWW.

[16] `http://www.cenitbuscamedia.es/`

[17] `http://mayor2.dia.fi.upm.es/oeg-upm/index.php/es/ontologies/224-buscamedia-ontologies-m3`

Nuestro trabajo futuro enfoca la evolución de HDT-FoQ hacia una cobertura total del lenguaje SPARQL que nos permita asentar la infraestructura requerida para la culminación de diferentes objetivos relacionados con el proyecto Buscamedia.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la Science Foundation Ireland: Grant No.∼SFI/08/CE/I1380, Lion-II (*primer autor*), el proyecto Buscamedia: CENIT 2009-1026 (*grupo UPM*), el Ministerio de Ciencia e Innovación de España: TIN2009-14009-C02-02 y Fondecyt: 1-110066 (*grupo UVa/UChile*). Todos los autores pertenecen a la Red Temática Española de Linked Data: TIN2010-10811-E.

Referencias

[1] J. Fernández, M. Martínez-Prieto, and C. Gutierrez, "Compact Representation of Large RDF Datasets for Publishing and Exchange," in *Proc. of ISWC*, 2010, pp. 193–208.

[2] *RDF Primer*. W3C Recommendation, 2004, `www.w3.org/TR/rdf-primer/`.

[3] *SPARQL Query Language for RDF*. W3C Recommendation, 2008, `www.w3.org/TR/rdf-sparql-query/`.

[4] B. Schandl, B. Haslhofer, T. Bürger, A. Langegger, and W. Halb, "Linked data and multimedia: the state of affairs." *Multimedia Tools Appl.*, vol. 59, no. 2, pp. 523–556, 2012.

[5] I. H. Witten, A. Moffat, and T. C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 1999.

[6] N. Brisaboa, R. Cánovas, F. Claude, M. Martínez-Prieto, and G. Navarro, "Compressed String Dictionaries," in *Proc. of SEA*, 2011, pp. 136–147.

[7] M. Martínez-Prieto, M. Arias, and J. Fernández, "Exchange and Consumption of Huge RDF Data," in *Proc. of ESWC*, 2012, pp. 437–452.

[8] M. Martínez-Prieto, J. Fernández, and R. Cánovas, "Querying RDF Dictionaries in Compressed Space," *ACM SIGAPP Applied Computing Reviews*, vol. 12, no. 2, pp. 64–77, 2012.

[9] P. Ferragina and G. Manzini, "Opportunistic Data Structures with Applications," in *Proc. of FOCS*, 2000, pp. 390–398.

[10] R. González, S. Grabowski, V. Mäkinen, and G. Navarro, "Practical Implementation of Rank and Select Queries," in *Proc. of WEA*, 2005, pp. 27–38.

[11] R. Grossi, A. Gupta, and J. Vitter, "High-order entropy-compressed text indexes," in *Proc. of SODA*, 2003, pp. 841–850.