Descubrimiento y geoprocesado de Información espacial utilizando un buscador semántico

Javier Márquez[1], David Cifuentes[1], J.E. Córcoles[1], Antonio Quintanilla[1]

[1] Universidad de
Castilla-La Mancha

javier.marquez@uclm.es
david.cifuentes@uclm.es
corcoles@dsi.uclm.es
antonio.quintanilla@uclm.es

Resumen

El número de servicios e información espacial ofrecida por las Infraestructuras de Datos Espaciales (IDEs) ha aumentado considerablemente. A pesar de ello, no se ha apreciado un aumento considerable del número de usuarios que demandan estos servicios. El motivo es la dificultad con que se encuentran estos para la recuperación de información de las IDEs. Para contribuir a un cambio de tendencia, es necesario crear sistemas más fáciles de descubrimiento de información. Para ello se ha desarrollado un motor de búsqueda de ámbito mundial para buscar capas (WMS) y features (WFS) de servicios OGC. Además, el motor de búsqueda tiene en cuenta las relaciones semánticas entre los conceptos, los operadores topológicos y los contextos geográficos específicos incluidos en las consultas. Los tiempos de respuesta son similares a los motores de búsqueda Web clásica, gracias a la estructura de indexación de contenidos, que incluye un grid y ficheros invertidos asociados a cada celda del grid. Además, los resultados de consulta de usuario son ordenados por criterios conceptuales y geográficos.

Palabras clave: Web Semántica, motor de búsqueda, servicios OGC, recuperación de información.

1. Introducción

Muchos de los esfuerzos en el desarrollo de la Web se basan en la llamada Web Semántica. Dentro de la Web Semántica, un dominio que requiere especial atención es la Web Semántica Geoespacial. La enorme variedad de codificación semántica hace que sea especialmente difícil de tramitar las solicitudes de información geoespacial. En un futuro próximo, la Web Semántica Geoespacial permitirá recuperar servicios y/o recursos espaciales y no espaciales mediante consultas sencillas. Para ello es necesario incorporar semántica a la información espacial y la definición de estructuras que puedan llevar a cabo consultas eficientes. Con el fin de hacer esto posible, la solución se enmarca en dos áreas: Recuperación de Información –Information Retrieval - (IR) y el descubrimiento de servicios geoespaciales (DSG) [1] de forma automática en Internet.

El sistema que planteamos gestiona consultas con el patrón <*qué, operador, donde*>, tales como *"Ríos que cruzan Londres"*. La mejora de rendimiento del sistema se ha reforzado con la agregación de relaciones semánticas entre conceptos (sinónimos, hiperónimos, hipónimos, etc.) utilizando ontologías, la indexación de los contenidos para acelerar los tiempos de respuesta, el soporte de operadores topológicos, y un geoparser para conocer el ámbito geográfico.

2. Descripción del sistema

El sistema propuesto tiene una estructura típica de motor de búsqueda Web, es decir, el cliente envía peticiones a un servidor, que generará respuestas adecuadas. Las peticiones serán del tipo *<capas y/o features, qué, operador, donde>*. Ejemplo, *"Capas con ríos y carreteras en Madrid"*, mientras que las respuestas tienen la forma *<capa, servidor>* para consultas de capas y *<nombre, geometría, URL>* para features. La URL puede estar relacionada con una ontología RDF o una respuesta *DescribeFeature* de un servicio WFS. La arquitectura del sistema se basa en el diseño conceptual [5] que se ha refinado después de explorar varias alternativas de implementación. La arquitectura del servidor es la parte más compleja de nuestra arquitectura. Se han construido los siguientes módulos:

Crawler: Busca enlaces correspondientes a servicios geoespaciales estandarizados por el OGC a través de Internet de una manera similar a [1]. Cuando se encuentra un servicio válido, se obtienen los metadatos mediante solicitudes *GetCapabilities*. Por último, el crawler (rastreador) registra los metadatos del servicio disponible (servidores y capas de información) en un sistema de información (base de datos).

Inference engine (IR): Analiza los metadatos del servicio para calcular el área geográfica. Un algoritmo se ejecuta teniendo en cuenta los topónimos y las coordenadas encontradas en los metadatos del servicio. El motor de inferencia también analiza los títulos de las capas con el fin de descubrir la verdadera semántica de la información. Esto se realiza haciendo peticiones a ontologías disponibles en Internet mediante SPARQL.

Módulo de indexación: Este subsistema maneja la estructura de indexación de contenidos. Esta estructura se compone de una cuadricula (grid) que cubre todo el área geográfica del buscador. Además, existen dos ficheros invertidos asociados a cada celda de la cuadrícula: uno de ellos se utiliza para la indexación de capas obtenidas de WMS (ráster) y otro para servicios WFS (vector), que se utiliza para la indexación de las features individuales obtenidas a partir de los WFS. Un fichero invertido es una estructura de datos indexados que guarda una correspondencia entre el contenido, tales como palabras o números, y sus ubicaciones en un sistema de información.

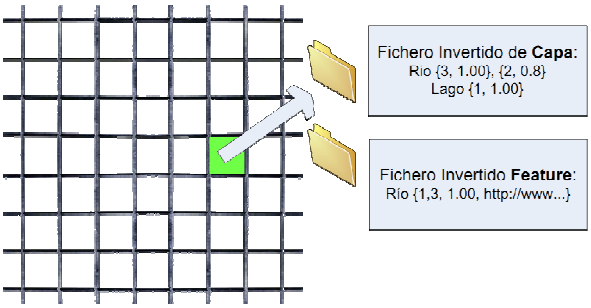

Figura 2. Estructura de Indexación. El grid abarca el área geográfica. Cada celda contiene dos ficheros

El módulo de indexación actualiza los ficheros invertidos asociados a las celdas de la cuadricula cuyo bounding box de capa intersecte o contenga el bounding box de la capa.. Los nuevos registros indexados en el fichero invertido *f* tienen la forma *{c: l_i, s_i}*, donde *c* es un concepto representado por una o más palabras; *li* es el identificador de una capa cuyo título contiene el concepto *c*, también puede que *c* no tenga contenido el título de la capa l_i, pero sí una relación semántica con otro concepto en el título; y finalmente, s_i es una puntuación, cuyo rango es [0, 1], que mide el nivel de fiabilidad por la que la capa l_i contiene información sobre el concepto *c*.

En cuanto a los ficheros invertidos para la indexación de features por cada capa WFS encontrada por el crawler, se envía una petición *GetFeature* para la recuperación de sus características, indexando sólo la información que interesa en este caso; topónimo y geometría de cada feature.

La actualización se produce en los ficheros invertidos que afecta a las celdas obtenidas del cruce geométrico. Los nuevos registros indexados *f* tiene la forma *{c: f_i, l_i, s_i, r_i}*, donde *c* es un concepto representado por una o más palabras; f_i es un identificador interno de feature, l_i es el identificador de capa que ofrece la feature f_i, y cuyo título contiene el concepto *c,* o por el contrario, una relación semántica con otro concepto del título de la capa. s_i es la puntuación de las capas indexadas. Y por último, r_i es una URL que puede obtener más información de la feature f_i. La URL puede ser una ontología en Internet construida sobre RDF o una solicitud *DescribeFeature* de WFS. El topónimo y la geometría de cada feature se almacenan en la base de datos.

Query parser (analizador de consultas): Es capaz de navegar por la estructura de índice con el fin de responder apropiadamente a las consultas de usuario. Las respuestas tienen un conjunto de resultados que se ordenan utilizando las puntuaciones s_i previamente calculadas. El analizador de consultas accede a los datos del sistema de información por cada capa *li,* obteniendo datos sobre (título, escala máxima y mínima, bounding box, etc.) y/o servicios (propietario, URL, forma de respuesta, etc.). El modulo gestiona los operadores topológicos contenidos en las consultas, lo que permite obtener características geográficas, lista de topónimos de features, geometrías y URLs.

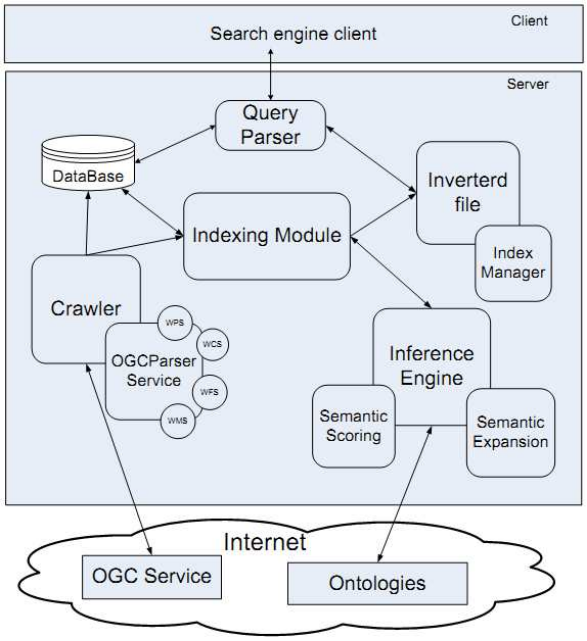

Figura 1. Arquitectura del sistema. Comunicación entre los diferentes módulos.

3. Proceso de actualización de la estructura de indexación

En el apartado anterior se ha visto la estructura de indexación, que consiste en una cuadrícula que cubre el ámbito geográfico del motor de búsqueda. Este ámbito puede ser la Tierra, un continente, país o cualquier zona, en función de las necesidades de la aplicación. Asociados a cada celda de la cuadrícula, hay dos ficheros invertidos: uno de ellas para indexar las capas (WMS) que ofrece los servicios OGC y otro para la indexación de features obtenidas a partir de servicios WFS (Figura 2). El número total de ficheros invertidos es dos por celda. Esta arquitectura reduce el cálculo de operaciones topológicas en tiempo de consulta, ya que el análisis se centra en las celdas afectadas del ámbito. Además, la razón por la cual se necesita dos ficheros invertidos por celda, es debido a que la información indexada es diferente para capas (WMS) y features (WFS).

El proceso de actualización de la estructura de índice se lleva a cabo en tiempo de indexación, por lo que los usuarios finales no son conscientes de ello. Este proceso se inicia después de encontrar un nuevo servicio WMS o WFS. El proceso se divide en las siguientes partes: indexación de capas (WMS) y/o de features si se trata de un servicio WFS, expansión semántica de los ficheros invertidos utilizando ontologías externas y asignación de puntuación para el ranking de resultados.

3.1 Indexación de Capas

La primera parte del algoritmo es la actualización de la estructura de índices. En él se actualizan los ficheros invertidos una vez ha sido analizada la respuesta "*GetCapabilities*" del servicio (WMS o WFS indistintamente). Los siguientes pasos se realizan para cada capa l_i del servicio.

1 Obtener las celdas del grid que intersectan entre el bounding box de la.

2 Obtener conceptos bien conocidos de ontologías externas a partir de metadatos del título de capa.
3 Cada palabra del título de la capa que no haya sido reconocida por las ontologías, se indexa en todos los ficheros invertidos asociados a esas celdas *{palabra$_i$: l_i; 1:00}* y además tendrá la puntuación máxima 1.00. Por ejemplo, una capa llamada "*Estados de la UE*", donde "*UE*" no es reconocido por la ontología como "*Unión Europea*".
4 Las palabras del título reconocidas como conceptos bien conocidos por la ontologías, serán indexadas en todos los ficheros invertidos asociados a sus celdas: *{palabra$_i$: l_i; 1:00}*. Por ejemplo, "*Estado*" en una capa llamada "*Estados de la UE*".
5 Si un conjunto de palabras consecutivas del título de capa (*palabra$_i$, palabra$_{i+1}$…,palabra$_{i+n}$*) son reconocidas por la ontología como un sólo concepto, se registran como palabras conjuntas *{palabra$_i$palabra$_{i+1}$…palabra$_{i+n}$: l_i; 1:00}*. Por ejemplo, *"Carretera secundaria"* en una capa llamada *"Carreteras secundarias del Reino Unido"*.

3.2 Indexación de Features

Esta parte del algoritmo sólo se realiza si se ha descubierto un nuevo servicio WFS, actualizando los ficheros invertidos para la indexación de features. Los pasos se ejecutan por cada capa l_i de WFS:

1. Se realizan peticiones *"GetFeature"* al servicio WFS para recuperar todas las features f_i de las capas l_i. Si la respuesta es demasiado compleja, es posible realizar varias peticiones a áreas más pequeñas – divisiones de igual tamaño- (tiles).
2. Se recuperan dos de los campos del servicio: topónimo (nombre descriptivo) y geometría. Para conocer los campos del conjunto se realiza peticiones *"DescribeFeatureType"* al servicio WFS para obtener los tipos de la entidad analizada. Los tipos de campo dan una idea implícita de la información semántica.

Los topónimos se incluyen normalmente con etiquetas: *gml:name*, propiedades *gml: AbstractFeatureType* o *xsd: string*. La geometría por lo general utilizan la etiqueta *gml: GeometryPropertyType*. Sin embargo, este enfoque puede no ser suficiente ya que puede que haya más de un etiqueta con este tipo, por lo que debe realizarse una selección manual. Se podría dar una solución más real al problema semántico de servicios WFS, pero no hay una solución para dicho problema en este artículo. Otras soluciones hacen uso de perfiles que asocian campos con su significado semántico de forma manual [4, 6].

3. Obtener las celdas para cada capa a través de la intersección entre el bounding box de la capa l_i y la cuadrícula. Este paso fue realizado en la primera parte del algoritmo.
4. Peticiones a las ontologías externas con el fin de obtener conceptos bien conocidos de los metadatos del título de capa (previamente se aplica un proceso de reducción de la palabra a su raíz – stemming-).
5. Cada palabra del título de la capa que no ha sido reconocida como un concepto bien conocido por las ontologías, es indexada en todos los ficheros invertidos asociados a sus celdas $\{palabra_i: f_i; l_i; 1:00; r_j\}$. El proceso se repite para cada f_i.
6. Las palabras del título que hayan sido reconocidos como conceptos bien conocidos por las ontologías, serán indexadas en todos los ficheros invertidos asociados a sus celdas: $\{palabra_i: f_i, l_i; 1:00; r_j\}$. Este proceso se repite para cada f_i.
7. Si un conjunto de palabras consecutivas ($palabra_i$,$palabra_{i+1}$.. $palabra_{i+n}$) del título de capa, ha sido reconocido como un sólo concepto, se registra como un único concepto en cada una de sus celdas $\{palabra_i palabra_{i+1}..palabra_{i+n}: f_i; l_i; 1:00; r_j\}$ Este proceso se repite para cada f_i.

3.3 Expansión semántica

La tercera parte del algoritmo añade nuevos conceptos a los ficheros invertidos de capas y features. Estos conceptos no están explícitamente en el título de la capa pero, en cambio, son conceptos relacionados semánticamente. Otros trabajos relacionados con IR y los servicios OGC hacen uso de la expansión semántica, ampliando el título original mediante sinónimos, hiperónimos,.., obtenidos de Servicios de Ontología Web (SOW)[3]. Por ejemplopara una capa titulada *"Autopistas en Madrid"*, sus términos ampliados serán algo así como *"autopistas, carreteras, red de carreteras"*. Estos términos estarán indexados de tal manera que para consultas *"carreteras en Madrid"*, la capa será recuperada. Nuestra propuesta es la separación de la indexación y puntuación de acuerdo a la similitud semántica.

El sistema recupera los resultados con expansión semántica como un motor de búsqueda tradicional, además de ordenarlos de acuerdo a criterios semánticos. Otros autores hacen uso de la expansión en tiempo de consulta de usuario y no en tiempo de indexación. La decisión de emplear la expansión en la indexación, es debido a que los requisitos de almacenamiento como título de capa no son exigentes y el rendimiento del motor de búsqueda es mejorado, ya que es transparente al usuario. Sostenemos que si una capa proporciona información sobre *"ríos"*, también tiene que ser recuperada para la consulta *"Hidrografía"* y viceversa. El nuevo enfoque añade información semántica basada en la recuperación de información clasificada [2]. La explicación es la siguiente:

Si *{r1,r2,..,rn}* son relaciones semánticas (ej: hipónimos, hiperónimos, sinónimos, merónimos,...); w_i es el peso de un relación r_i; t_i es un árbol construido con conceptos relacionados con r_i (ej: *"rio"* sería hijo en un árbol de hiperónimos de *"corrientes de agua"*). $K(i)_{a,b}$ es el número de niveles del árbol entre el concepto *a* y *b*, por tanto, un nuevo registro $\{c: l_i; s_i\}$ donde *c* es un concepto que no está explícito en el título de capa, semánticamente está relacionado por r_i con el concepto del título de capa (C_{title}), tendrá la siguiente puntuación:

$$si(c) = w_i^{k(i)c,ctitle}$$

Es posible manipular el orden de resultados de la consulta de usuario sólo ajustando w_i. Por ejemplo, la capa *li "Parques de Madrid"* debe ser recuperada como *"Zonas verdes de Madrid"* de los primeros resultados ya que *"Zonas verdes"* es considerado un sinónimo de *"Parque"* (semánticamente son iguales). Por otra parte, las r_i como hiperónimo, debe tener un valor w_i por debajo de su máxima.

Por último, es necesario poner un punto final en el proceso de cálculo de los nuevos conceptos que se relacionan semánticamente con otros contenidos, mediante peticiones recursivas a ontologías en tiempo de indexación. Estas ontologías suelen tener recursos mediante propiedades RDFS como *"subClassOf"* o *"isSubClassOf"*, que son útiles para inferir las relaciones de hiponimia e hiperonimia. Otras propiedades RDFS como *"label"* se pueden utilizar para obtener sinónimos. Todas estas solicitudes pueden hacerse a través de lenguajes de consulta como SPARQL.

4. Procesado de consultas

En esta sección, se describe el proceso interno de búsqueda de la consulta, desde el momento en que un usuario introduce un texto hasta la respuesta obtenida. Para ello se mostrarán un par de ejemplos:

Ejemplo 1 - *"Autopistas que cruzan Francia"*: La primera tarea es distinguir en la consulta que corresponde al *"qué"*, y al *"dónde"*. Para ello, necesitaremos un geoparser. El geoparser podría ser externo (ej: GoogleMaps o GeoNames), el problema es que sólo proporcionan puntos y es necesario la geometría completa. Una solución puede ser el uso de topónimos y de geometrías a partir de servicios WFS. Además, cada registro del geoparser debe tener un campo con las celdas del grid -ver estructura de indexación- que contienen la geometría. Permitiendo centrarse en el área geográfica de interés y descartar los ficheros invertidos a celdas lejanas.El geoparser es una lista de registros *{topónimo: geometría, celdas}*. Obviamente, *"Francia"* será encontrado por el geoparser; como el *"dónde"*, y el resto de la consulta el *"qué" ("Autopistas"*). En caso de que el geoparser encuentra múltiples

resultados, el usuario tendrá que desambiguar. En ocasiones, la vista actual del mapa se utiliza para resolverlo de forma automática.

La segunda tarea es acceder a los ficheros invertidos para recuperar *"autopistas"*en los ficheros invertidos. El operador topológico "cruzan" obtendrá las celdas a consultar. La tercera tarea es filtrar los elementos preseleccionados, en el ejemplo *"crucen"* con Francia. La estructura de grid permite descartar los elementos que no comparten celdas con Francia. Tanto la geometría de capas (bounding box) y de features pueden ser leídas desde la base de datos. La última tarea es ordenar todos los elementos válidos por su puntuación s_i, aquellos resultados que son semánticamente más cercanos *"autopista"* deben colocarse en primer lugar. Lo aconsejable definir un umbral inferior de puntuación para aquellos elementos que no vayan a ser recuperados. Esto dependerá del tipo aplicación, metadatos u otros datos de capas y/o features disponibles en la base de datos.

Ejemplo 2 - *"Ríos al norte de Londres"*: Este ejemplo presenta la principal diferencia en el operador topológico con respecto al Ejemplo 1. En este caso, las celdas correctas son aquellas que cortan o contienen la geometría de Londres, y también las celdas adyacentes al norte, porque son los candidatos para contener los ríos del norte de Londres. Por lo tanto, la única diferencia cuando se cambia el operador topológico, es el conjunto de celdas y de ficheros invertidos a tener en cuenta. El resto de tareas siguen siendo las mismas.

5. Pruebas

Los experimentos tienen por objeto medir el rendimiento del motor de búsqueda a través de ratios de recuperación de información y tiempos de respuesta. Se han utilizado dos conjuntos de consultas:

1 – Pruebas para estudiar los ratios de recuperación de información y clasificación de resultados (sub-secciones 5.1 y 5.2) basadas en consultas a capas WMS con el operador – contiene –

2- El otro estudio está basado en consultas más complejas para analizar el comportamiento del motor (sub-sección 5.3) en el descubrimiento de features.

5.1 Análisis de los ratios de recuperación de información

El rendimiento del sistema se ha comparado con un motores de búsqueda tradicionales basado en palabras clave. Se han indexado casi un centenar de capas, a las que un subconjunto de 19 preguntas de la forma <*lo que, en, dónde*> comparan el rendimiento del motor (Figura 4).

1	*Lagos en Castilla-La Mancha*
2	*Rios en Toledo*
3	*PNOA en España*
4	*Ortofoto en Albacete*
5	*PNOA en Castilla-La Mancha*
6	*Estanques de agua en Guadalajara*
7	*Mapa de relieve en España*
8	*Divisiones administrativas en Castilla-La Mancha*
9	*Vias pecuarias en Guadalajara*
10	*Curvas de nivel en España*
11	*Aguas estancas en Cuenca*
12	*Imagen Landsat en Castilla-La Mancha*
13	*Escuelas en Albacete*
14	*Escuelas secundarias en Toledo*
15	*Construcciones en España*
16	*Embalses en Castilla-La Mancha*
17	*Hidrografía en España*
18	*Red de transporte en Madrid*
19	*Estaciones de tren en Albacete*

Figura 3. Evaluación de consultas al sistema. Testeo utilizado entre un buscador clásico y el sistema definido.

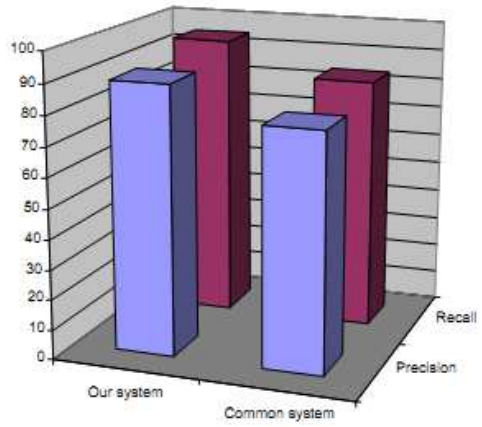

Figura 4. Comparación de precisión y recuperación. El sistema obtiene mejores resultados en ambos indicadores.

Se ha evaluado la *precisión* - fracción de documentos relevantes recuperados en la búsqueda- y la *recuperación* – fracción de consultas relevantes que son recuperadas con éxito- .

El sistema realiza una mejora del 10% gracias a los conceptos relacionados, la recuperación es muy alta y la precisión es suficientemente alta. Debido a las limitaciones de espacio del artículo no se ha podido mostrar un análisis más profundo de las pruebas.

5.**2 Clasificación de resultados**

Se han comprobado los resultados que aparecen en las primeras posiciones, son igual o mejor a los motores de búsqueda tradicional. Aquellas consultas cuyos conceptos tienen una relación semántica (ej: consulta 6 y 11) obtienen mejores resultados. Los motores tradicionales no son eficientes para clasificar capas. Muchos de los algoritmos IR se basan en factores que no son significativos para este tipo de consultas, demostrando un mejor enfoque a la importancia semántica de la información y las coberturas geográficas.

5.3 Búsqueda de Features basado en Grid

Este estudio se centra en la medición de tiempos de respuesta para consultas de la forma <*qué, operador, dónde*> para búsqueda de features. El banco de pruebas consiste en 20 preguntas complejas con diferentes operadores (Figura 5).

Resultados tras analizar los tiempos de respuesta:

Celdas afectadas: celdas del grid tenidas en cuenta en una consulta. El número es directamente proporcional a la superficie de las features que forman parte del *"dónde"*. Ejemplo, *"Provincias contenidas en España"*, las celdas afectadas son mayor que

1	Municipios contenidos en Baleares
2	Regiones que tocan Baleares
3	Regiones contenidas en Barcelona
4	Regiones contenidas en Madrid
5	Municipios contenidos en Madrid
6	Regiones cerca de Valencia
7	Regiones que tocan Valencia
8	Provincias cerca de Badajoz
9	Provincias que tocan Badajoz
10	Regiones que tocan Galicia
11	Municipios contenidos por Galicia
12	Provincias contenidos por Cataluña
13	Regiones cerca de Cataluña
14	Regiones que tocan Cataluña
15	Provincias cerca de Extremadura
16	Municipios contenidos Aragón
17	Provincias contenidas por Andalucía
18	Regiones cerca de Andalucía
19	Provincias cerca de Andalucía
20	Regiones contenidas en España

Figura 5. Banco de pruebas utilizadas para analizar el rendimiento.

"Provincias contenidas en Madrid" (Madrid es una región de España), las celdas afectadas serían menor. A mayor número de celdas afectadas, mayor número de a ficheros invertidos.

Features afectadas: Son las features indexadas en las celdas afectadas y son utilizadas en las operaciones topológicas.

Operador: consultas con el mismo*"qué"* y *"dónde"* (celdas afectadas) obtienen tiempos diferentes de respuesta para diferentes operadores. Los costes computacionales de operadores son diferentes.

6. Conclusión

La nueva arquitectura presentada para el descubrimiento de capas y features que ofrecen los servicios OGC en Internet, tiene un alcance diferente a los catálogos de servicios (CSW) disponibles en las IDE. Como se ha demostrado, por lo que se refiere a los servicios geoespaciales, los resultados pre-calculados asociados a capas, teniendo en cuenta aspectos semánticos y geográficos, mejoran los resultados de consultas por ranking basadas en palabras claves. Además, se ha demostrado que el uso de ontologías de dominio junto con los metadatos del servicio es una buena manera de mejorar la precisión y recuperación de información. Además, el el rendimiento en la búsqueda de features con una estructura de indexación basada en grid con ficheros invertidos para la gestión geográfica / textual es prometedor para ciertas condiciones, aunque existen otras estructuras como los R-tree o los grid multinivel que son más eficaces. En futuros trabajos se probarán éstas otras estructuras.

Agradecimientos. Este trabajo es fruto de una de las líneas de I+D prioritarias del IDR y de SIGTEL Geomática S.L., la cual cuenta con financiación del Ministerio de Ciencia e Innovación, Subprograma IMPACTO (proyecto “RED DE INTELIGENCIA E INNOVACIÓN DE TURISMO”) y Subprograma Torres Quevedo.

Referencias

[1] N. Chen, J. Gong, and Z. Chen. A high precision ogc web map service retrieval based on capability aware spatial search engine. In Proceedings of the 2nd international conference on Advances in computation and intelligence, ISICA'07, pages 558-567, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] K. Janowicz, M. Schwarz, and M. Wilkes. Implementation and evaluation of a semantics-based user interface for web gazetteers. In Visual Interfaces to the Social and the Semantic Web (VISSW 2009) Workshop in conjunction with the International Conference on Intelligent User Interfaces (IUI 2009), Sanibel Island, Florida, 2009.

[3] J. Lacasta, J. Nogueras-Iso, R. Béjar, P. R. Muro-Medrano, and F. J. Zarazaga-Soria. A web ontology service to facilitate interoperability within a spatial data infrastructure: Applicability to discovery. Data Knowl. Eng., 63:947{971, December 2007.

[4] M. Lutz, U. Einspanier, E. Klien, and S. Huebner. An Architecture for Ontology-based GI Discovery and Retrieval of Geographic Information. In Ontology-based Discovery and Composition of Geographic Information Services (SWSDN2004), pages 179{188, 2004.

[5] J. Márquez, J. E. Corcoles, and A. Quintanilla. A semantic index structure for integrating ogc services in a spatial search engine. In IEEE International Conference On Open Systems, Kuala Lumpur, Malaysia, 2010, 2010.

[6] M. Paul and S. K. Ghosh. An approach for service oriented discovery and retrieval of spatial data. In Proceedings of the 2006 international workshop on Service-oriented software engineering, SOSE '06, pages 88-94, New York, NY, USA, 2006. ACM.