Búsquedas inteligentes de toponimia

Félix José Hernández, Jorge Rosales y José Julio Rodrigo

Depto. de Ingeniería
Cartográfica de Canarias S. A.
C/ Panamá 1, 38009 Santa Cruz de Tenerife
{jrosales, jrodrigo, fhernandeze}@grafcan.com

Resumen

En este documento se explican las estrategias y tecnologías utilizadas en el desarrollo de un sistema, capaz de realizar búsquedas sobre nuestra base de datos de toponimia a partir del lenguaje natural.

Palabras clave: Jornadas, formato de documento, ponencias, toponimia, búsqueda, topónimo, software, libre.

1 Introducción

Muchos de los sistemas de búsqueda de toponimia actuales obligan a facilitar información previa que el usuario no siempre conoce (provincia, municipio, etc.) y en ocasiones poseen distintas interfaces de búsqueda en función de la entidades que se quieran localizar (calles, topónimos, etc.) Esta ponencia describe los resultados obtenidos en el desarrollo de un motor genérico de búsqueda que reciba un parámetro único de entrada y se ajuste a las expectativas de resultado de los usuarios en términos de importancia percibida.

2 Toponimia y búsquedas inteligentes

Una posible definición podría ser la siguiente: La toponimia u onomástica geográfica es una disciplina de la onomástica que consiste en el estudio y origen de los nombres propios de un lugar. Los topónimos, nombre de lugar, en ocasiones tienen su origen en apellidos o nombres propios de personas, pero habitualmente su origen está en algún aspecto físico del lugar que designan [1].

Las búsquedas inteligentes tienen mucho que ver con la inteligencia artificial y los sistemas expertos [2]. Nosotros no seremos tan ambiciosos en esta primera fase. Nos conformaremos con aquellas soluciones que permitan realizar búsquedas sobre campos de texto, que tengan en cuenta la gramática y estén integradas en los sistemas de base de datos o que su instalación sea lo menos costosa posible.

2 Soluciones actuales

Según los requisitos expuestos en el apartado anterior, podemos destacar los siguientes sistemas como posibles soluciones a implantar:

Oracle Text [3], Google [4], PostgreSQL Full Text Search [5], SQL Server Full-Text Search [6]

A continuación describiremos los detalles de la implementación de SQL Server Full-Text Search y el PostgreSQL Full-Text Search para nuestra base de datos de toponimia.

3 SQL Server Full-Text Search

Actualmente el visor geográfico de Cartográfica de Canarias S.L. (GRAFCAN) dispone de dos tipos de búsqueda, uno sobre el callejero y otro sobre la toponimia. Para el callejero tenemos que seleccionar la isla, el municipio y finalmente un literal para la calle, mientras que en toponimia sólo se necesita el literal. Con la finalidad de unificar y simplificar las búsquedas, se decidió crear un único punto de entrada, es decir, una sólo caja de texto y que el sistema fuera lo suficientemente inteligente para devolver los resultados más acordes con la cadena de búsqueda.

La información de topónimos ya la tenemos cargada en base de datos, por lo que nos centramos en buscar alguna característica del propio motor que nos permitiera conseguir nuestro objetivo. Por un lado tenemos el conocido operador LIKE, que consigue devolvernos los registros que contienen un determinado prefijo, sufijo o palabra completa en el texto. Su rendimiento no deja de ser todo lo bueno que esperamos.

Indagando un poco más, encontramos el Full-Text Index. Esta característica ya existía en versiones anteriores al SQL Server 2005, pero en esta, se han realizado importantes mejoras de rendimiento y uso. A diferencia de una consulta LIKE que

realizada sobre millones de filas puede tardar minutos, la consulta de texto puede tardar segundos, en función del número de filas que devuelva.

Otra ventaja consiste en que podemos utilizar este tipo de consultas sobre datos almacenados en columnas de tipo IMAGE o VARBINARY(max), estos tipos permiten el mayor contenido de datos, por lo que podemos almacenar un documento completo en un registro de la base de datos y luego realizar búsquedas inteligentes sobre él.

Los aspectos básicos de las búsquedas de texto son:

- Índices de texto
- Catálogo de texto
- Separador de palabras
- Testigo
- Lematizador
- Filtro
- Llenado o rastreo
- Palabras irrelevantes

Podemos encontrar información detallada sobre estos términos en la siguiente dirección [7].

En cuanto a las consultas, existen dos formas de hacerlas, una mediante la cláusula CONTAINS y otra mediante FREETEXT. La principal diferencia entre ellas es que para utilizar FREETEXT simplemente basta con una palabra o frase de entrada, en este caso es el motor de búsqueda quien se encarga de 'parsear' esta frase en sus posibles combinaciones verbales, de género o si llevan o no tildes. Este tipo de búsquedas se denominan también 'difusas'. Por otro lado, tenemos la opción CONTAINS que nos permite utilizar operadores en la frase de entrada. Tenemos un NEAR o separar las palabras con comillas dobles, un ejemplo sería "reflector" NEAR "trasero".

Para nuestra primera versión del buscador de toponimia decidimos utilizar el tipo FREETEXT, de esta forma liberamos al usuario del conocimiento de operadores que exige el otro método CONTAINS, eso sí, en la siguiente versión crearemos un apartado de búsqueda avanzada donde lo utilizaremos, seguro que los resultados serán mucho más ajustados y rápidos.

Con este tipo de búsquedas obtenemos bastantes resultados, y el orden en que nos lo presenta corresponde con un criterio interno del motor de base de datos. Este criterio o clasificación 'rank', cuya fórmula aparece en la figura 1 y su detalle en [8], se basa principalmente en el número de veces que se repite un término en el texto.

```
Rank = Σ[Terms in Query] w * ( ( ( k1 + 1 ) tf ) / ( K + tf ) ) * ( ( k3 + 1 ) qtf / ( k3 + qtf ) ) )
Where:
w is the Robertson-Sparck Jones weight.
In simplified form, w is defined as:
w = log10 ( ( ( r + 0.5 ) * ( N - R + r + 0.5 ) ) / ( ( R - r + 0.5 ) * ( n - r + 0.5 ) )
N is the number of indexed rows for the property being queried.
n is the number of rows containing the word.
K is ( k1 * ( ( 1 - b ) + ( b * dl / avdl ) ) ).
dl is the property length, in word occurrences.
avdl is the average length of the property being queried, in word occurrences.
k1, b, and k3 are the constants 1.2, 0.75, and 8.0, respectively.
tf is the frequency of the word in the queried property in a specific row.
qtf is the frequency of the term in the query.
```

Figura 1. Cálculo del 'rank' para consultas FREETEXT

En nuestro caso, la búsqueda de topónimos, ¿qué tenemos que hacer si queremos que al buscar por 'santa cruz de tenerife' aparezca el término municipal como primer resultado? Pues bien, como vimos antes, el algoritmo de cálculo se basa en el número de veces que se repite el término en el registro, por lo que probamos a localizar el registro correspondiente al municipio de 'santa cruz de tenerife' y modificamos su campo 'toponimo' (el que hemos configurado previamente para que utilice la funcionalidad de Full-Text) repitiendo varias veces su contenido, es decir, si inicialmente su valor era 'santa cruz de tenerife', ahora sería 'santa cruz de Tenerife santa cruz de Tenerife santa cruz de tenerife'.

Siguiendo con el razonamiento anterior, optamos por realizar lo mismo con los términos de Calles, Avenidas, Carreteras, Farmacias, Colegios, etc., obteniendo en muchos casos los resultados esperados.

Cabe destacar que de vez en cuando se nos cuela algún resultado 'inesperado' que nos obligar a analizar su comportamiento y adaptar el campo 'toponimo' a nuestras necesidades. No hay que olvidar que estamos utilizando la búsqueda 'difusa', que se presta a este tipo de comportamientos.

Podemos concluir que el rendimiento de esta solución es aceptable, quizás nos quede un poco grande para la toponimia, ya que la idea sería buscar en registros con grandes cantidades de texto. En cualquier caso, es una opción a tener en cuenta

que facilita la interacción con el usuario, tanto a la hora de la entrada de datos como en la rapidez y calidad de sus resultados.

El siguiente paso sería utilizar la información geográfica, es decir, cuando buscamos un topónimo, lo que realmente queremos es saber dónde está situado exactamente. De momento, tenemos resuelto las búsquedas de texto, pero sería muy útil agregar información espacial, así por ejemplo el usuario podría dibujar una ventana sobre el visor geográfico y luego lanzar la búsqueda de texto.

Si además lo tenemos todo integrado en el mismo motor, mejor. Lamentablemente, la versión de SQL Server 2005 no dispone de tipo de datos para almacenar geometrías, probaremos con la versión 2008 o daremos el salto al un entorno PostgreSQL+PostGis+Full-Text Search, que veremos en el siguiente apartado.

4 PostgreSQL Server Full-Text Search + PostGis

Con el fin de tener una 'segunda opinión' en el tema de las búsquedas inteligentes, intentamos la configuración de un sistema con PostgreSQL. Este motor de base de datos, bastante extendido en la actualidad, en su versión 8.3 posee también la característica 'Full Text Search' [9].

Aunque la idea es la misma que con SQL Server, a la hora de su implementación existen algunas diferencias, donde destacaría la creación de un campo de tipo 'tsvector' en la misma tabla donde tenemos los datos (topónimos).

Desde el punto de vista de las consultas, el motor nos permite utilizar dos opciones , por un lado, la utilización de operadores como &, !, etc., o como vimos en SQL Server que sea el propio motor quien 'analice' la cadena de entrada. Utilizaremos está última opción.

Otro dato importante es la clasificación de los resultados, lo que conocemos con el 'rank'. Con este motor tenemos algo que no vimos en SQL Server y es que podemos asignar pesos a cada campo de la tabla, de tal forma que los resultado estén ordenador de mayor a menor importancia. Este es el ejemplo de asignación de pesos para nuestra tabla de toponimia:

Campo	N	Ejemplo
Localización	A	Tenerife SANTA CRUZ DE TENERIFE 38038
Clasificación	B	Términos Municipales

Nombre	C	Santa Cruz de Tenerife

Tabla 1. Registro de ejemplo de la tabla topónimos.

Campo	Ejemplo
Tsvector	'cruz':3A,10C 'sant':2A,9C '38038':6A 'termin':7B 'tenerif':1A,5A,12C 'municipal':8B

Tabla 2. Ejemplo de representación interna 'ts_vector'.

Destacamos que PostgreSQL nos ofrece hasta un nivel D (el de menor importancia).

Otra funcionalidad del Full-Text Search, es la capacidad de resaltar lo términos de búsqueda en el texto de cada registro, por ejemplo, si buscamos por 'santa cruz de tenerife, el primer resultado es: Santa Cruz de Tenerife. Utiliza la negrita en HTML para destacar cada término.

Existe una serie de ficheros o diccionarios en los que se apoya esta característica de PostgreSQL: diccionario simple, sinónimos, palabras de parada, etc. Uno que tuvimos que modificar o mejor dicho adaptar fue el de sinónimos. El problema estaba en que cuando buscábamos por topónimos de Las Palmas, los primeros registros en aparecer eran de La Palma. El funcionamiento de sistema se quedaba en este caso con el lexema 'Palm' y con lo que, efectivamente, aparecían primero los registros palmeros. Agregando a la tabla de sinónimos 'Palmas' conseguimos que el sistema obtuviera como lexema de búsqueda 'Palmas'.

Otro caso parecido ocurre con el término 'guagua' o autobús. Procedemos de igual forma y ahora cuando alguien consulta por autobús le aparecen también los registros de las 'guaguas'.

Afortunadamente el motor de base de datos PostgreSQL es capaz de almacenar información geográfica, mediante el complemento PostGIS. Si unimos las búsquedas inteligentes con Full-Text Search y su capacidad de devolver la información espacial, tenemos el servicio o los servicios que estábamos buscando.

5 Servicios de toponimia

Tipo	Dirección

XML	http://195.57.95.94:8000/toponimiaxml/1/10/santa%20cruz%20tenerife/
KML	http://195.57.95.94:8000/toponimiakml/1/10/Santa%20Cruz%20de%20Tenerife/1/9102/
XML BBOX	http://195.57.95.94:8000/toponimiaxmlbbox/1/10/plaza/302497.39986611576,3129752.492385236,334841.12575733615,3149379.535074254/0/0/

Tabla 3. Servicios de toponimia (direcciones temporales).

Si nos fijamos en el tipo XML, vemos que los parámetros los pasamos en la propia dirección, es decir, 'toponimiaxml' es la función, el '1' es la página, el '10' corresponde con el número de registros por página y finalmente 'santa cruz de tenerife' es el nombre del topónimo a buscar.

6 Clientes

Actualmente, sólo el visor de IDE Canarias consume los servicios del apartado anterior.

Visor IDE Canarias: http://www.idecan.grafcan.es/idecan/

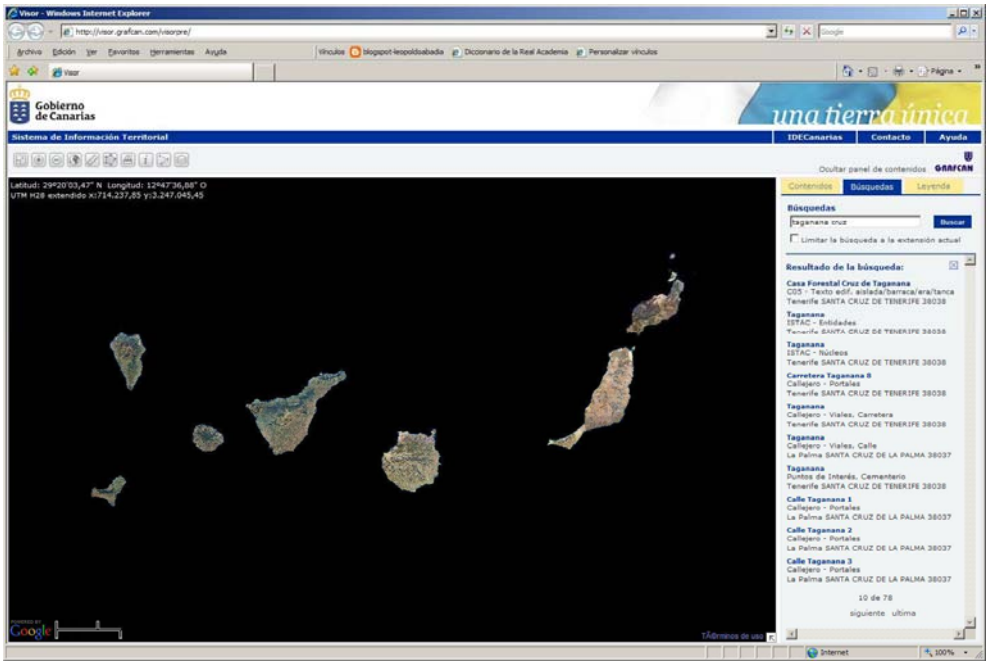

Figura 2. Cliente IDE Canarias

7 Conclusiones

El tema de las búsquedas inteligentes sobre texto es una tarea que ya implementan la mayoría de los motores de búsqueda, nosotros nos hemos aprovechado de ella para resolver nuestra necesidad de un sistema de consulta rápido, simple y que permita una interacción con lenguaje natural con el usuario, sobre nuestra base de datos de toponimia.

La siguiente tabla muestra una serie de ejemplos de búsqueda:

Texto a buscar
aeropuerto el hierro
casas rurales en la laguna
embalse santa cruz

calle del castillo
calle del castillo, 12
cines en la laguna

Tabla 3.Ejemplo de búsquedas.

Dentro de las líneas futuras a llevar a cabo podemos destacar la optimización, mejora de rendimiento, seguridad y su posible integración con los servicios WFS.

8 Anexo estadísticas

Resumen de topónimos de Canarias por isla.

Isla	Cantidad
Tenerife	242.257
Gran Canaria	168.905
Lanzarote	43.094
La Palma	31.571
Fuerteventura	25.348
La Gomera	18.560
El Hierro	5.942
Total	**535.677**

Tabla 4.Resumen por isla.

Referencias

[1] Wikipedia, http://es.wikipedia.org/wiki/Toponimia

[2] Wikipedia, http://es.wikipedia.org/wiki/Sistema_experto

[3] Oracle Text, http://www.oracle.com/technology/products/text/index.html

[4] Google, http://infolab.stanford.edu/~backrub/google.html

[5] PostgreSQL, http://www.postgresql.org/docs/8.3/interactive/textsearch.html

[6] SQL Server, http://technet.microsoft.com/en-us/library/ms142547.aspx

[7] Microsoft SQL Server 2005, Full-Text Search Concepts, http://technet.microsoft.com/en-us/library/ms142547(SQL.90).aspx

[8] SQL Server 2005, Cálculo de la clasificación, http://technet.microsoft.com/en-us/library/ms142524.aspx

[9] PostgreSQL Full Text Search, http://www.postgresql.org/docs/8.3/static/textsearch.html