Indexación mediante Arrays de Sufijos para Recuperación de Información Geográfica*

Nieves R. Brisaboa[1], Miguel R. Luaces[1], Gonzalo Navarro[2], and Diego Seco[1]

[1] Laboratorio de Bases de Datos, Universidade da Coruña
Campus de Elviña, 15071, A Coruña, España
{brisaboa,luaces,dseco}@udc.es
[2] Departamento de Ciencias de la Computación, Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
gnavarro@dcc.uchile.cl

Resumen La recuperación de información geográfica constituye un área de investigación joven, pero que está captando mucha atención debido al interés de los usuarios de repositorios de información digital en obtener información relevante en el lugar geográfico donde se encuentran o que van a visitar. El objetivo principal de dicha área consiste en recuperar información relevante no solo en cuanto a su contenido textual sino también en cuanto a su referente geográfico (es decir, al lugar al que se refiere). Para ello una de las tareas fundamentales es la indexación de la información. La mayoría de las propuestas realizadas hasta la fecha combinan un índice invertido con algún índice espacial.
En este artículo presentamos una estructura de indexación que no emplea un índice invertido para indexar el contenido textual, sino que emplea un array de sufijos. Esto permite dotar a los sistemas de recuperación de información geográfica de nueva funcionalidad. Por ejemplo, no excluye lenguajes humanos que no son fácilmente separables en palabras, como el chino o el coreano. Además, constituye una alternativa a la utilización de un índice invertido cuando la búsqueda de frases es frecuente.

Key words: Recuperación de información geográfica, indexación, array de sufijos, R-tree.

1. Introducción

La recuperación de información geográfica (GIR) constituye un área de investigación incipiente que está captando gran cantidad de atención dentro de las comunidades de sistemas de información geográfica (GIS) y recuperación de información (IR). Los estudios sobre la enorme cantidad de referencias geográficas que se encuentran dentro del texto desestructurado de bibliotecas digitales

* Este trabajo ha sido financiado parcialmente por el Ministerio de Ciencia e Innovación (PGE y Fondos FEDER) [ref. TIN2009-14560-C03-02], y por la Xunta de Galicia (cofinanciado con Fondos FEDER) [refs. 10SIN028E y 2010/17] para el grupo de la Universidade da Coruña; Fondecyt [ref. 1-110066] para el tercer autor; y programa Ángeles Alvariño para el cuarto autor.

y otros repositorios de información han abierto la puerta a la colaboración de investigadores de GIS con la comunidad IR. Además, la era de los dispositivos móviles ha incrementado drásticamente el interés de los usuarios de repositorios de información digital acerca de información relacionada con áreas donde se encuentran o que van a visitar. Esto no se restringe solo al ejemplo clásico de puntos de interés cerca de donde se encuentra el usuario. Algunos otros ejemplos pueden ser el amante del misterio que quiere comprar on-line libros sobre casas encantadas en su zona, o el precavido turista que desea consultar en una biblioteca digital catástrofes naturales cerca de su posible destino vacacional.

Una prueba más del interés suscitado por esta área es el *Workshop* en recuperación de información geográfica [1], realizado por primera vez en 2004 y en el que este año se debatieron cuatro temas fundamentales dentro de la recuperación de información geográfica: definición de huellas geográficas de documentos, indexación, elaboración de rankings, y estrategias de evaluación. En este trabajo nos centramos en la segunda de ellas: la indexación. En la Sección 2 revisamos las principales propuestas realizadas para indexar documentos tanto textual como espacialmente. En general, dichas estructuras combinan un índice invertido con algún índice espacial (un grid, un R-tree, etc.). Aunque la elección del índice invertido está justificada dada su popularidad y demostrada eficiencia, hay ciertos escenarios donde no ofrece toda la funcionalidad necesaria al permitir solo búsqueda de *palabras* y *frases* sobre texto en lenguaje natural. Esto excluye algunos lenguajes humanos, como el chino o el coreano, y también lenguajes donde las palabras se componen como aglutinaciones de las partículas de búsqueda, como el finés o el alemán. Desde la comunidad de *String Processing and Pattern Matching* se han propuesto alternativas eficientes para estos escenarios, siendo muchas de ellas variantes comprimidas del clásico array de sufijos.

En este artículo exploramos el uso del array de sufijos en sistemas de recuperación de información geográfica. Esta propuesta hereda las ventajas del uso de arrays de sufijos en recuperación de información. Es decir, es adecuado para lenguajes que no se pueden separar de manera natural en palabras, puede utilizarse en sistemas que requieran la búsqueda de patrones arbitrarios (y no solo palabras o frases) y, en términos de eficiencia, ofrece una interesante alternativa a los índices invertidos para la búsqueda de frases. Además, nuestra propuesta explota ciertas características del array de sufijos que hacen que se pueda combinar de manera natural y elegante con el R-tree (el índice espacial más utilizado en sistemas de información geográfica).

2. Trabajo relacionado

Las estructuras de indexación desarrolladas para sistemas GIR deben permitir la resolución de consultas que tengan tanto una componente textual como una componente espacial. En esta sección revisamos algunas de las propuestas realizadas hasta la fecha.

El primer proyecto de amplio alcance en recuperación de información geográfica en el que se propusieron estructuras de indexación fue el proyecto SPIRIT [2].

Estas estructuras se basan en la combinación de una estructura de *grid* [3] (uno de los índices espaciales más sencillos) con un índice invertido [4,5] (el índice textual clásico). Los autores de este trabajo realizaron pruebas combinando el *grid* y el índice invertido en una única estructura y también manteniéndolos por separado. Su conclusión más importante es que manteniendo ambos índices separados se consigue un menor coste de almacenamiento aunque, por contra, puede implicar mayores tiempos de respuesta. Además, tener separados los índices tiene ventajas en cuanto a la modularidad, y facilidad de implementación y mantenimiento. Sus resultados también muestran que los métodos propuestos son capaces de competir en términos de velocidad y coste de almacenamiento con estructuras de indexación textual clásicas. Aunque la estructura propuesta es muy sencilla, y se han propuesto otras que la superan tanto en velocidad como en coste de almacenamiento, este trabajo es muy relevante ya que ha establecido una de las características distintivas de todas las propuestas posteriores. Dicha característica establece la distinción entre *estructuras híbridas*, que combinan los índices textual y espacial en una única estructura, y *estructuras de doble índice*, que los mantienen por separado.

Trabajos más recientes, como [6,7], describen las dos estrategias base de la indexación en sistemas GIR teniendo en cuenta las propuestas del proyecto SPIRIT. Estas dos propuestas se nombran como *Text-First* y *Geo-First*. A nivel general, ambos algoritmos asumen la existencia de un índice espacial y de un índice textual, y emplean la misma estrategia para resolver las consultas: primero se emplea un índice para filtrar los documentos (el índice textual en el caso de *Text-First* y el índice espacial en el caso de *Geo-First*); el conjunto de documentos resultante se ordena por sus identificadores y posteriormente se filtra usando el otro índice (el índice espacial en el caso de *Text-First* y el índice textual en el caso de *Geo-First*). Estos nombres se pueden emplear también para estructuras híbridas, de tal modo que si la estructura emplea primero el índice textual es de tipo *Text-First*, mientras que si emplea primero el índice espacial es de tipo *Geo-First*.

En [8] los autores proponen emplear un índice invertido y un R-tree [9] (en lugar de la estructura *grid*), y realizan pruebas combinándolos de las tres formas que acabamos de describir. En sus experimentos concluyen que mantener por separado los índices aumenta los tiempos de consulta (esta misma conclusión había sido obtenida en [2]) y que sus estructuras son más eficientes que las que emplean la estructura de *grid*.

En [7], los autores comparan tres estructuras que combinan un índice invertido con la estructura *grid*, con el R-tree, y con curvas de llenado del espacio [10,11]. Las curvas de llenado del espacio se basan en el almacenamiento de los objetos espaciales en un orden determinado por la forma de la curva de llenado. La conclusión de los autores es que la estructura que emplea estas curvas de llenado del espacio mejora el rendimiento de las otras aproximaciones.

Finalmente, en el proyecto STEWARD [12], los autores proponen emplear una estructura que mantenga por separado un índice invertido y un *Quad-tree* [13]. El Quad-tree es una estructura similar al *grid* (ambas son dirigidas por el

espacio y no por los objetos a indexar), que va dividiendo el espacio en cuadrantes hasta que los objetos se pueden almacenar en una página de disco. Además, en este trabajo se propone el empleo de un optimizador de consultas que decida si emplear primero el índice textual o el índice espacial en función de la previsión de cuál va a obtener menos resultados. Para poder emplear este optimizador de consultas el sistema debe almacenar estadísticas que permitan realizar la estimación del número de documentos resultante de una búsqueda de términos clave particular o de una ventana de consulta espacial determinada.

En resumen, la mayoría de las estructuras propuestas hasta la fecha combinan un índice invertido con algún índice espacial. La elección del índice invertido es sensata y está justificada teniendo en cuenta que su eficiencia ha sido ampliamente demostrada y que es una de las estructuras más utilizadas en IR. Sin embargo, como ya mencionamos en la introducción de este artículo, existen ciertos escenarios donde no es posible su uso. Las principales alternativas al índice invertido para estos escenarios, en donde la separación en palabras no es posible, se engloban en lo que se denomina índices *full-text*. Dichos índices permiten la búsqueda eficiente de cualquier patrón arbitrario (y no solo de palabras o frases) en grandes repositorios de información. Además, se ha dedicado mucho esfuerzo a la compresión de los mismos y actualmente ocupan tamaño proporcional al texto comprimido (es decir, menos que el texto original) permitiendo búsquedas de cualquier patrón arbitrario más rápido que si se realizasen sobre el texto descomprimido. Estos índices que permiten reemplazar el texto (ya que es recuperable desde el propio índice) se han denominado *auto-índices* [14].

En este trabajo exploramos el uso de arrays de sufijos en sistemas GIR. En concreto, combinamos un array de sufijos con un R-tree. Dado que estas dos estructuras constituyen la base de nuestro índice, las explicamos con más detalle en la siguiente sección.

3. Componentes de nuestra propuesta

Como acabamos de mencionar, los bloques constituyentes de nuestra estructura son el array de sufijos y el R-tree. Para hacer el artículo auto-contenido, en esta sección explicamos ambas estructuras en más detalle. El lector familiarizado con dichas estructuras puede saltarse esta sección ya que no contiene ningún detalle particular de nuestra aportación. De igual modo, remitimos al lector que quiera ampliar información sobre los arrays de sufijos a [14] y sobre el R-tree a [15].

3.1. Arrays de sufijos

Los índices *full-text* surgen en dominios donde la separación en palabras no es posible. Esto incluye, además de los lenguajes humanos ya mencionados como chino o coreano, otras aplicaciones como ADN, genes, proteínas, audio, etc. En este contexto, los arrays de sufijos [16] aparecieron en los noventa como una mejora importante sobre los árboles de sufijos [17] ya que requieren mucho

menos espacio y mantienen prácticamente la misma funcionalidad y eficiencia. Dado un texto T, su array de sufijos SA es una permutación de todos los sufijos en orden lexicográfico. Dicha permutación requiere $n \log n$ bits para un texto de n caracteres y es muy eficiente para buscar patrones arbitrarios o, dicho de otro modo, cualquier subcadena del texto. Dado que el SA almacena todos los sufijos en orden lexicográfico, es fácil ver que todos los sufijos prefijados por un patrón P se encuentran contiguos en el SA. Por tanto, si se quieren localizar todas las ocurrencias de P en T es suficiente con realizar dos búsquedas binarias para encontrar el inicio y el final del intervalo donde se encuentran dichas ocurrencias. Si el largo del patrón es m, en cada paso de la búsqueda binaria se pueden realizar hasta m comparaciones y esto resulta en una complejidad de $O(m \log n)$ para el algoritmo de búsqueda.

Los SA no son *auto-índices*, es decir, no reemplazan el texto sino que son complementarios al texto. Además, en la práctica requieren hasta cuatro veces el tamaño del texto. Esto ha motivado una gran cantidad de trabajo en lo que se denominan *auto-índices comprimidos*, es decir, índices que reemplazan el texto (ya que puede ser recuperado a partir de ellos), ocupan tamaño proporcional al texto comprimido, y resuelven las operaciones propias de un índice *full-text* de manera eficiente. El *array de sufijos comprimido* (CSA) [18,19] es uno de los auto-índices comprimidos más populares. Además, en dominios donde la búsqueda de patrones arbitrarios no es interesante, existen variantes del CSA orientadas a palabra [20] que ofrecen una alternativa interesante a los índices invertidos especialmente para búsqueda de frases y cuando hay poco espacio disponible para el índice. En la Figura 1 mostramos un ejemplo de un array de sufijos orientado a palabra. Como paso previo a la construcción del array de sufijos se construye un *vocabulario* con todas las palabras distintas del texto y se crea un array de enteros T_{id} donde cada palabra se reemplaza por su correspondiente posición en el vocabulario. El SA se construye sobre este array T_{id}.

En la figura mostramos también un ejemplo de búsqueda del patrón "*el CERI*". El resultado de la misma son las ocurrencias del patrón en el texto, es decir, las posiciones 1 y 7. Como se puede observar, dichos valores se encuentran consecutivos en el SA. Insistimos en esta característica de que el resultado de la búsqueda es un rango, ya que es clave para la combinación con un R-tree.

3.2. R-tree

El R-tree [9] es uno de los métodos de acceso espacial más populares y se puede considerar un ejemplo paradigmático. Esta estructura se basa en un árbol balanceado derivado del B-tree [21] que divide el espacio en rectángulos de cobertura mínima (MBRs) agrupados jerárquicamente. El número de nodos hijo de cada nodo interno varía entre un mínimo y un máximo. El árbol se mantiene balanceado dividiendo aquellos nodos que tienen un número de descendientes por encima del umbral de carga máxima y combinando aquellos otros que tienen un número de descendientes por debajo del umbral de carga mínima. Cada nodo hoja tiene asociado un MBR que delimita el área del espacio que cubre ese nodo. Además, los nodos internos también almacenan un MBR que delimita el área que

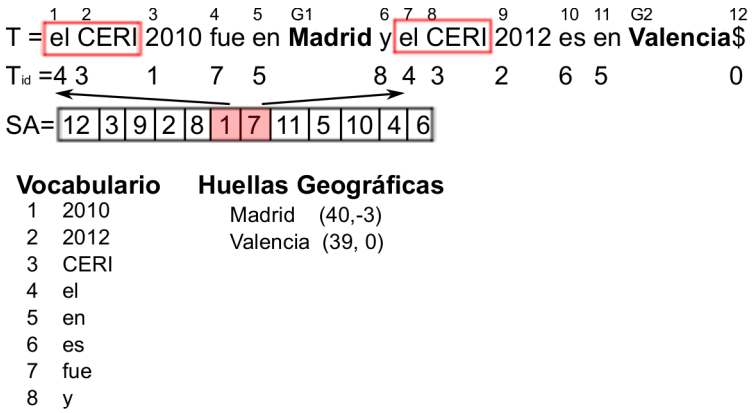


Figura 1. Ejemplo de array de sufijos orientado a palabra (WCSA). En esta figura adelantamos que las referencias o huellas geográficas reciben un tratamiento especial, por lo que no se indexan en el array de sufijos SA.

cubren todos sus descendientes. La descomposición del espacio que proporciona el R-tree es adaptativa (es decir, dependiente de la distribución de los objetos geográficos indexados) y puede presentar solapes (es decir, los nodos del árbol pueden representar regiones no disjuntas). Aunque no ofrece garantías teóricas (en el peor caso puede visitar todos los nodos aun cuando el resultado de la consulta sea vacío) ha demostrado ser muy eficiente en la práctica y se encuentra disponible en la mayoría de las extensiones espaciales para bases de datos. Sobre la propuesta original de Guttman se han ido proponiendo muchas variantes para mejorar su eficiencia.

Aunque el R-tree soporta varias operaciones, por ejemplo distintas variantes de los vecinos más cercanos, en este trabajo solo explotamos su eficiencia para resolver consultas de rango. En dos dimensiones, estas consultas se corresponden con rectángulos y el resultado son todos los objetos geográficos que tienen algún punto en común con el rectángulo de consulta. Aunque su uso más habitual es con datos geográficos, no debe olvidarse que es un índice multidimensional y presenta una buena escalabilidad ante el aumento del número de dimensiones.

4. Nuestra propuesta

En esta sección describimos cómo combinar el SA con un R-tree. Como se ha explicado en la sección anterior, existen muchas variantes del SA que ofrecen diferente funcionalidad y son adecuadas para diferentes escenarios. Todas ellas comparten la propiedad de que la búsqueda de un patrón resulta en un rango de posiciones consecutivas en el array de sufijos. Ésta es la única propiedad que asume nuestra estructura y, por tanto, puede trabajar con cualquier SA. La elección del SA depende del dominio donde se vaya a utilizar. Por ejemplo,

se puede emplear con un WCSA resultando en una alternativa a la estructura presentada en [8]. En lo sucesivo denotamos como SA cualquier array de sufijos.

La estructura está diseñada para trabajar en sistemas GIR. Por tanto, asumimos la existencia de una etapa anterior a la indexación en la que se anota cada sufijo con los referentes geográficos a los que se refiere (ver [22] para más información sobre la tarea de geo-referenciación de topónimos). El caso más habitual en sistemas GIR consiste en anotar cada documento con un conjunto de objetos geográficos que representan los lugares mencionados en el texto de dicho documento. Nuestra estructura funciona con dicho esquema (todos los sufijos que forman parte de un documento heredan su huella geográfica), pero no se restringe a él. En nuestro sistema se define el concepto de *unidad geo-referenciada* consistente en un conjunto de sufijos consecutivos que comparten huella geográfica. Dichas unidades pueden representar documentos, pero también párrafos, sentencias, o incluso sufijos que se encuentran a menos de una cierta distancia de la posición en el texto donde se menciona una referencia geográfica. Volviendo sobre el ejemplo ilustrado en la Figura 1, se podría asumir que el texto está compuesto de dos unidades geo-referenciadas: el rango $[1-5]$ se corresponde con la huella geográfica *Madrid* y el rango $[6-11]$ se corresponde con *Valencia*. Si refinamos la búsqueda del ejemplo como *el CERI* en {(37,0),(41,2)} (coordenadas de un rectángulo de consulta que contiene la Comunidad Valenciana), el resultado ya no serán las posiciones 1 y 7, sino solo la 7. Este tipo de consultas constituyen el objetivo de nuestra estructura.

Siguiendo la tendencia marcada en la literatura de índices para GIR, proponemos dos formas de combinar las componentes de nuestra estructura. La más ingenua consiste en una estructura de doble índice que mantiene el SA y el R-tree por separado. El SA se construye con todos los sufijos del texto y el R-tree se construye con todos los referentes geográficos (cada referente geográfico almacena como identificador la posición en el SA del sufijo al que se refiere). En esta estructura, el algoritmo de consulta realiza un primer paso en el que se resuelven de manera independiente las componentes textual y espacial de la consulta. El resultado final se obtiene en un segundo paso realizando la intersección de los resultados parciales obtenidos en el paso anterior.

El problema de esta variante es que la intersección que se realiza en el segundo paso del algoritmo puede ser muy costosa e ineficiente. Es decir, puede haber muchos candidatos resultantes de la búsqueda textual y muchos candidatos resultantes de la búsqueda espacial, pero la intersección de las dos listas de candidatos ser vacía. La idea es almacenar cierta información extra que permita acelerar dicha intersección. En nuestra propuesta aprovechamos que el R-tree es muy eficiente para resolver búsqueda de rangos y que la búsqueda en un SA resulta en un rango. Esto nos permite combinar ambas estructuras de manera natural y elegante. El SA se mantiene igual que en la variante anterior. Por su parte, el R-tree pasa de ser bidimensional a ser tridimensional. Como tercera dimensión se emplea la posición en el SA. Es decir, por cada objeto geográfico que contenía el R-tree en la variante anterior, se construye un objeto tridimensional con las coordenadas originales en las dos primeras dimensiones más una

tercera dimensión que es la posición en el SA que ocupa el sufijo que tiene como huella geográfica dicho objeto. La Figura 2 muestra un ejemplo de este proceso. En ella, los ejes X e Y representan el espacio geográfico original (es decir, el espacio al que son traducidas las huellas geográficas de los sufijos) y el eje Z representa la posición del sufijo en el SA. Debe aclararse que el R-tree es un índice multidimensional, por lo que no es necesario ningún cambio sobre el mismo. Por el mismo motivo, si se desea extender esta estructura para soportar búsquedas espacio-temporales en GIR, la extensión es igual de natural. Además, el R-tree soporta la inserción tanto de puntos, que dan lugar a puntos tridimensionales (d, e y f en la figura), como de rectángulos, que dan lugar a cajas planas o rectángulos que varían su posición en la tercera dimensión (a, b, y c).

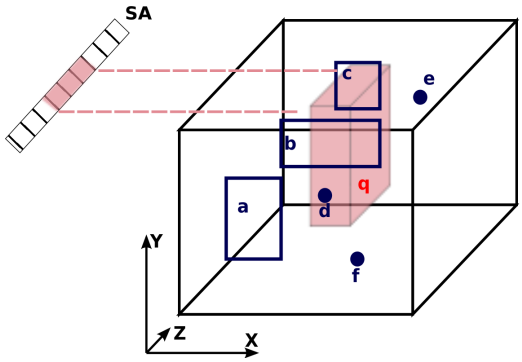


Figura 2. Ejemplo de proyección del rango resultante de una búsqueda textual en el SA a una consulta tridimensional q en el espacio indexado por el R-tree.

El algoritmo de búsqueda realiza en primer lugar una búsqueda textual en el SA, obteniendo como resultado un rango. En segundo lugar, se construye una consulta tridimensional con la parte espacial de la consulta original más el rango obtenido en el paso anterior (ver Figura 2). Finalmente, se resuelve dicha consulta tridimensional en el R-tree obteniendo el resultado de la consulta espacio-textual.

Dado que nuestra estructura es una generalización del array de sufijos, obtiene como resultado una *lista de ocurrencias*. Es decir, dada una consulta, la estructura reporta las posiciones del texto donde aparece el patrón buscado y que tienen una huella geográfica que tiene una intersección no vacía con la consulta espacial. Dichas posiciones serán de caracter o de palabra según lo sea el array de sufijos empleado. Este estándar de facto en la comunidad de *pattern matching* puede resultar chocante para el lector familiarizado con la recuperación de información más tradicional donde el resultado más común es una *lista de documentos*. Nos gustaría remarcar que en los últimos años se ha dedicado mucho esfuerzo a adaptar los arrays de sufijos (y otros índices *full-text*) a este estándar [23,24]. Básicamente se guarda cierta información adicional en paralelo

con el índice textual que indica para cada posición a qué documento pertenece. Una vez solucionada la búsqueda textual el problema se reduce a reportar los identificadores diferentes que caen dentro del rango. Adaptar dichas estructuras a nuestro índice no es inmediato ya que el resultado final de la estructura ya no es un rango. Dejamos como línea abierta el explorar el potencial de nuestra propuesta para realizar *listado de documentos* en lugar de *listado de ocurrencias*.

5. Experimentos

En esta sección presentamos algunos experimentos realizados para comprobar el rendimiento de nuestra propuesta. Estos resultados preliminares no tratan de ser exhaustivos, sino una prueba de concepto que muestre la aplicabilidad de nuestra solución. Todos los experimentos presentados aquí se realizaron en una máquina Intel Core i3 M330@2.13GHz, con 4GB de memoria RAM y sistema operativo Ubuntu (kernel 2.6.32-39). El código en C se compiló con `gcc` versión 4.4.3 empleando las directivas de compilación `-m32 -O3`.

El conjunto de datos utilizado para las pruebas corresponde a los experimentos para *reconocimiento de nombres de lugar* en CoNLL y pertenece al *Reuters Corpora* [25]. En concreto se emplearon los ficheros en inglés etiquetados y revisados a mano. En total la colección se compone de 1.393 documentos. El número de sufijos es 1.089.032 y el número de referencias geográficas localizadas es 10.599 (una media de 7 por documento).

Para la implementación de nuestra estructura empleamos como array de sufijos un WCSA. En concreto, empleamos la implementación que se presenta en [20]. En cuanto al R-tree, empleamos la implementación de Marios Hadjieleftheriou que está disponible en [26].

En nuestros experimentos comparamos la variante de doble índice de nuestra propuesta (es decir, la que mantiene el array de sufijos y el R-tree por separado) con la variante híbrida (es decir, la que emplea un R-tree tridimensional utilizando como tercera dimensión la posición en el array de sufijos). Las consultas realizadas se componen de una parte textual y una parte espacial. La parte textual está formada por patrones compuestos por una palabra (dado que utilizamos un array de sufijos orientado a palabra). En cuanto a la parte espacial, se crearon ventanas de consulta rectangulares cuyo tamaño representa el 1 % del área cubierta por las huellas geográficas de los documentos. La tabla 1 muestra los resultados de este experimento. En la primera columna se muestra el espacio que necesita cada variante y las dos columnas restantes se corresponden con distintos tipos de consulta: contar el número de ocurrencias y mostrar un fragmento de texto en torno a dichas ocurrencias, respectivamente. Las variaciones en esta última columna son mínimas ya que solo considera el tiempo para mostrar las ocurrencias una vez localizadas por lo que, en teoría, el tiempo esperado es el mismo y las variaciones en la práctica se deben únicamente al proceso de medición de tiempos.

Estos resultados concuerdan con las conclusiones previas obtenidas por Vaid et al. [2] y Zhou et al. [8]: la estructura de doble índice requiere menos espacio

	Espacio (MB)	Cont. (ms/occ.)	Most. (ms/carácter)
Doble índice	158	5,03	0,00101
Híbrida	181	0,005	0,00099

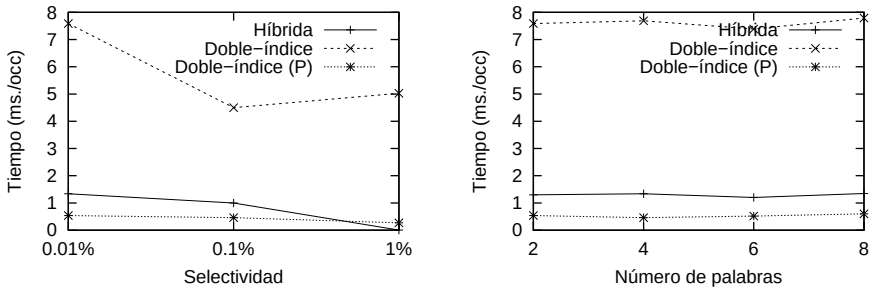
Tabla 1. Comparación de las variantes de doble índice e híbrida.

que la estructura híbrida, pero la eficiencia de la segunda es mucho mayor. Este resultado es fácilmente explicable ya que se puede interpretar que la estructura híbrida está almacenando información adicional para evitar el tener que realizar la intersección de resultados parciales. Sin embargo, nuestra estructura tiene una peculiaridad con respecto a las anteriores que hace que se pueda optimizar drásticamente el comportamiento de la estructura de doble índice. La clave es observar que en la estructura de doble índice se están insertando muchos objetos geográficos repetidos que tan solo se diferencian en el identificador (es decir, en el sufijo que representan). Por tanto, es posible realizar un preprocesado de los datos e insertar cada objeto geográfico una única vez, almacenando externamente para cada objeto geográfico una lista de los sufijos con los que se asocia. Con esta optimización se consigue reducir el tamaño del índice a poco más de 6MB con un tiempo de consulta de 0,27ms/occ. Observamos también que para consultas espaciales más selectivas (es decir, rectángulos de consulta más pequeños) la estructura de doble índice optimizada llega a ser más eficiente que la híbrida. Por ejemplo, repitiendo el mismo experimento con consultas que representan el 0,01 % del espacio la estructura de doble índice tarda 0,54ms/occ frente a los 1,34ms/occ de la estructura híbrida.

Es fácil de ver que la misma optimización no se puede realizar de manera trivial en la estructura híbrida ya que cada objeto geográfico no tiene por qué corresponderse con un rango contiguo de sufijos en el array de sufijos (recordemos que el array de sufijos es una permutación del texto original). Sin embargo, es posible que existan rangos en el array de sufijos que compartan la misma huella geográfica. Por ejemplo, en un array de sufijos orientado a palabra y donde el texto proviene de diferentes documentos, es probable que existan ciertas construcciones sintácticas que se repitan dentro del mismo documento pero que no aparezcan en los otros. Dichas construcciones formarán rangos consecutivos en el array de sufijos. Dejamos como línea de trabajo futuro el explorar la reducción del tamaño de la estructura híbrida teniendo en cuenta estas regularidades.

A continuación estudiamos en más detalle los tiempos de consulta cuando variamos la selectividad de la parte espacial de las consultas (Figura 3(a)) y cuando variamos el número de palabras que componen el patrón de la parte textual (Figura 3(b)). La estructura *Doble-índice (P)*, se corresponde con la optimización de la estructura de doble índice que acabamos de explicar.

A la vista de estos resultados la influencia de ambos parámetros en el tiempo de consulta no parece muy importante. Sin embargo, debe tenerse en cuenta que los tiempos mostrados son en milisegundos por ocurrencia y, por tanto, consultas que devuelven más resultados compensan en cierto modo el tardar más tiempo en completarse.



(a) Componente espacial. (b) Componente textual.

Figura 3. Influencia de las componentes espacial y textual en el tiempo de consulta.

6. Conclusiones y trabajo futuro

En este artículo proponemos el uso de arrays de sufijos para indexar la componente textual en estructuras de indexación para recuperación de información geográfica (a diferencia de trabajos previos que emplean índices invertidos). Las ventajas de nuestra propuesta son similares a las del uso de arrays de sufijos en recuperación de información. Es decir, es adecuado para lenguajes que no se pueden separar de manera natural en palabras, puede utilizarse en sistemas que requieran la búsqueda de patrones arbitrarios (y no solo palabras o frases), y ofrece una interesante alternativa a los índices invertidos sobre todo para la búsqueda de frases.

La estructura que presentamos obtiene como resultado las posiciones del patrón buscado en el texto (esto atiende a un modelo habitual para la comunidad de *String processing and pattern matching*). Una primera línea de trabajo futuro consiste en la adaptación de la misma al modelo habitual en recuperación de información (es decir, obtener como resultado los identificadores de los documentos relevantes para la consulta). Aunque existen algunos trabajos [23,24] que permiten hacer esto con arrays de sufijos, su adaptación a nuestra estructura no es inmediata ya que el resultado final de la misma deja de ser un rango contiguo del array de sufijos. La adaptación al modelo de IR constituye un paso previo a realizar una comparación justa con la estructura de Zhou et al. [8] que emplea un índice invertido y un R-tree. Finalmente, otra línea de trabajo futuro consiste en reducir el tamaño de la estructura híbrida. La idea consiste en emplear ciertas regularidades existentes en el array de sufijos para reducir el número de objetos tridimensionales a insertar en el R-tree (ya que es probable que posiciones consecutivas del array de sufijos compartan la misma huella geográfica).

Referencias

1. Purves, R., Gaio, M., Bucher, B.: Geographic information retrieval tutorial at agile (2012) Fecha de consulta: Marzo de 2012. Disponible en:

http://www.geo.uzh.ch/~rsp/girt.
2. Vaid, S., Jones, C.B., Joho, H., Sanderson, M.: Spatio-Textual Indexing for Geographical Search on the Web. In: Proc. of SSTD. (2005) 218 – 235
3. Nievergelt, J., Hinterberger, H., Sevcik, K.C.: The grid file: An adaptable, symmetric multi-key file structure. In: Proc. of the ECI Conference. (1981) 236–251
4. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley (1999)
5. Witten, I.H., Moffat, A., Bell, T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images. Academic Press (1999)
6. Martins, B., Silva, M.J., Andrade, L.: Indexing and ranking in Geo-IR systems. In: Proc. of GIR, ACM Press (2005) 31–34
7. Chen, Y.Y., Suel, T., Markowetz, A.: Efficient query processing in geographic web search engines. In: Proc. of SIGMOD. (2006) 277–288
8. Zhou, Y., Xie, X., Wang, C., Gong, Y., Ma, W.Y.: Hybrid index structures for location-based web search. In: Proc. of CIKM, ACM (2005) 155–162
9. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In Yormark, B., ed.: SIGMOD'84, Boston, Massachusetts, ACM Press (1984) 47–57
10. Morton, G.M.: A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd. (1966)
11. Böhm, C., Klump, G., Kriegel, H.P.: Xz-ordering: A space-filling curve for objects with spatial extension. In: Proc. of the SSD Conference. (1999) 75–90
12. Lieberman, M.D., Samet, H., Sankaranarayanan, J., Sperling, J.: STEWARD: Architecture of a Spatio-Textual Search Engine. In: ACMGIS. (2007) 186 – 193
13. Nelson, R.C., Samet, H.: A consistent hierarchical representation for vector data. In: Proc. of the SIGGRAPH Conference. (1986) 197–206
14. Navarro, G., Mäkinen, V.: Compressed full-text indexes. ACM Comput. Surv. **39**(1) (2007)
15. Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A.N., Theodoridis, Y.: R-Trees: Theory and Applications. Springer-Verlag New York, Inc. (2005)
16. Manber, U., Myers, G.: Suffix arrays: a new method for on-line string searches. In: SODA'90. (1990) 319–327
17. Apostolico, A.: The myriad virtues of subword trees. In: Combinatorial Algorithms on Words, NATO ISI Series. (1985) 85–96
18. Sadakane, K.: Compressed text databases with efficient query algorithms based on the compressed suffix array. In: ISAAC. (2000) 410–421
19. Grossi, R., Vitter, J.S.: Compressed suffix arrays and suffix trees with applications to text indexing and string matching. In: STOC. (2000) 397–406
20. Fariña, A., Brisaboa, N.R., Navarro, G., Claude, F., Places, A.S., Rodríguez, E.: Word-based self-indexes for natural language text. TOIS **30**(1) (2012) 1–34
21. Bayer, R., McCreight, E.M.: Organization and maintenance of large ordered indices. Acta Inf. **1** (1972) 173–189
22. Amitay, E., HarÉl, N., Sivan, R., Soffer, A.: Web-a-where: geotagging web content. In: SIGIR'04, New York, USA, ACM (2004) 273–280
23. Muthukrishnan, S.: Efficient algorithms for document retrieval problems. In: SODA. (2002) 657–666
24. Navarro, G., Nekrich, Y.: Top-k document retrieval in optimal time and linear space. In: SODA. (2012) 1066–1077
25. Lewis, D.D., Yang, Y., Rose, T., , Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research **5** (2004) 361–397
26. Hadjieleftheriou, M.: Spatial index library. (2009) Fecha de consulta: Marzo de 2012. Disponible en: http://libspatialindex.github.com/.