Selección espacial de pivotes dispersos para la búsqueda por similitud en espacios métricos *

Nieves R. Brisaboa[1], **Antonio Fariña**[1], **Óscar Pedreira**[1], **Nora Reyes**[2]

[1]Facultad de Informática, Universidade da Coruña
Campus de Elviña s/n, 15071, A Coruña, España
{brisaboa, fari, opedreira}@udc.es

[2]Departamento de Informática, Universidad Nacional de San Luis
Ejército de los Andes 950, San Luis, Argentina
nreyes@unsl.edu.ar

Abstract

Similarity search is a fundamental operation for applications that deal with unstructured data sources. In this paper we propose a new pivot-based method for similarity search, called *Sparse Spatial Selection* (SSS). The main characteristic of this method is that it guarantees a good pivot selection with a lower computational cost than other methods previously proposed. In addition, SSS adapts itself to the dimensionality of the metric space we are working with, without being necessary to specify in advance the number of pivots to extract. Furthermore, SSS is dynamic, this is, it is capable to support object insertions in the database without getting its efficiency reduced, it can work with both continuous and discrete distance functions, and it is suitable for secondary memory storage. In this work we provide experimental results that confirm the advantages of the method with several vectorial and metric spaces. We also show that the efficiency of our proposal is similar to that of other existing ones over vectorial spaces, although it is better over general metric spaces.

Keywords: Databases, algorithms and data structures

Resumen

La búsqueda por similitud es una operación fundamental en aplicaciones que trabajan con fuentes de datos no estructuradas. En este artículo proponemos un nuevo método de búsqueda por similitud basado en pivotes, que denominamos *Sparse Spatial Selection* (SSS). La principal característica de SSS es que garantiza una buena selección de pivotes con un coste computacional más bajo que otros métodos propuestos anteriormente. Además, SSS se adapta por sí sólo a la dimensionalidad del espacio métrico con el que estamos trabajando, sin que sea necesario especificar de antemano el número de pivotes que se van a extraer. Por otro lado, SSS es dinámico, es decir, es capaz de soportar inserciones de objetos en la base de datos sin que su eficiencia se vea reducida, puede usar tanto distancias continuas como discretas y se adapta bien a memoria secundaria. En este trabajo proporcionamos resultados experimentales que confirman las ventajas del método con distintos espacios vectoriales y métricos. Demostramos también que nuestra propuesta tiene una eficiencia similar a otras ya existentes en espacios vectoriales, aunque es mejor en espacios métricos generales.

Palabras claves: Bases de datos, algoritmos y estructuras de datos

*Este trabajo está parcialmente financiado por CYTED VII.J (RITOS2) y, para los tres primeros autores, por MCYT (PGE y FEDER) refs. TIC2003-06593 y TIN2006-15071-C03-03, y Xunta de Galicia ref. PGIDIT05SIN10502PR.

1. INTRODUCCIÓN

La búsqueda por similitud se ha convertido en una operación cada vez más importante para aplicaciones que trabajan con fuentes de datos no estructuradas. Por ejemplo, las bases de datos multimedia gestionan objetos sin ningún tipo de estructura, como imágenes, huellas dactilares o clips de audio. Recuperar la huella dactilar más semejante a una dada es un ejemplo de búsqueda por similitud. El problema de la recuperación de textos está presente en sistemas que van desde un simple editor hasta grandes buscadores. En este contexto puede ser interesante recuperar documentos similares a una consulta, o palabras muy similares a una dada para corregir errores de edición. Podemos encontrar más ejemplos de aplicación en áreas como la biología computacional (recuperación de secuencias de ADN) o el reconocimiento de patrones (donde un patrón puede clasificarse a partir de otros patrones similares ya clasificados). La búsqueda por similitud es una operación costosa. Esto ha dado lugar a una gran cantidad de trabajos que tratan de hacerla más eficiente y poder así aplicarla en grandes colecciones de datos.

El problema de la búsqueda por similitud puede formalizarse mediante el concepto de espacio métrico, que proporciona un marco formal independiente del dominio de aplicación concreto. Un *espacio métrico* $(\mathbb{X}, d)$ está formado por un universo de objetos válidos $\mathbb{X}$ y una *función de distancia* $d : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}^+$ definida entre ellos. La función de distancia determina la semejanza o proximidad entre dos objetos cualesquiera. Esta función es estrictamente positiva ($d(x, y) > 0$ y $d(x, y) = 0 \Leftrightarrow x = y$), simétrica ($d(x, y) = d(y, x)$) y satisface la desigualdad triangular ($d(x, z) \leq d(x, y) + d(y, z)$). El subconjunto finito $\mathbb{U} \subset \mathbb{X}$ de tamaño $n = |\mathbb{U}|$, denominado *diccionario* o base de datos, denota la colección de objetos sobre la que realizamos la búsqueda. Un espacio vectorial es un caso particular de espacio métrico, en el que cada objeto está formado por k coordenadas reales, siendo k la dimensión del espacio vectorial. La definición de la función de distancia depende del tipo de objetos con que estamos trabajando. En un espacio vectorial d podría ser una función de distancia de la familia L_s, definida como $L_s(x, y) = (\sum_{1 \leq i \leq k} |x_i - y_i|)^{\frac{1}{s}}$. Por ejemplo, L_1 es la *distancia Manhattan*, L_2 la *distancia euclídea* y $L_\infty = \max_{1 \leq i \leq k} |x_i - y_i|$ es la *distancia máxima*. En una colección de palabras podemos utilizar como función de distancia la distancia de edición $d(x, y)$, calculada como el número de inserciones, borrados o sustituciones necesarios para transformar una cadena x en una cadena y.

Hay tres tipos de consulta típicos que se pueden realizar sobre una colección de objetos en un espacio métrico. En la *búsqueda por rango* se recuperan los objetos $u \in \mathbb{U}$ que están a menos de una distancia r de la consulta q, esto es $\{u \in \mathbb{U} \ / \ d(q, u) \leq r\}$. En la búsqueda del *vecino más cercano* se recupera el objeto más similar a q, es decir, $\{u \in \mathbb{U} \ / \ \forall \ v \in \mathbb{U}, \ d(q, u) \leq d(q, v)\}$. La búsqueda de los k *vecinos más cercanos* es una generalización de la operación anterior. En este caso se recupera el conjunto $A \subseteq \mathbb{U}$ tal que $|A| = k$ y $\forall \ u \in A, \ v \in \mathbb{U} - A, \ d(q, u) \leq d(q, v)$. El tipo más básico de consulta es la búsqueda por rango, y las demás pueden implementarse en función de ésta [8]. En cualquier caso, la única información que podemos utilizar para realizar la búsqueda es la que proporciona la función de distancia. La forma básica de implementar estas operaciones consiste en comparar cada objeto de la colección con la consulta.

El problema es que, en general, la evaluación de la función de distancia tiene un coste computacional elevado, por lo que la operación de búsqueda no es eficiente cuando la colección tiene un gran número de elementos. El principal objetivo de los métodos de búsqueda en espacios métricos es precisamente reducir el número de evaluaciones de la función de distancia. Para lograrlo se construyen a

priori estructuras de índices sobre la colección que, junto con la desigualdad triangular de la función de distancia, permiten descartar elementos sin compararlos directamente con la consulta. Los diferentes métodos creados para realizar búsqueda eficiente en espacios métricos se diferencian entre sí en diversos factores. Algunos métodos sólo permiten que la función de distancia tome valores discretos (como, por ejemplo, la distancia de edición). En cambio, otros fueron diseñados para trabajar con funciones de distancia continuas. Ésta es una cuestión importante, ya que restringe la aplicación del método a determinados dominios. Algunos métodos son estáticos, en el sentido de que la colección no puede crecer una vez creado el índice. En cambio, los métodos dinámicos soportan inserciones en la base de datos, o incluso crean el índice según se añaden objetos a la colección inicialmente vacía. Otro factor importante es la posibilidad de almacenar estas estructuras en memoria secundaria de forma eficiente, y el número de operaciones de E/S necesarias para acceder a ellas. Estas cuestiones no deben descuidarse debido a su importancia en la aplicabilidad y eficiencia del método.

En general, según [8], hay dos clases de métodos de búsqueda, los basados en *clustering* y los basados en *pivotes*. Los *métodos de búsqueda basados en pivotes* seleccionan un conjunto de objetos de la colección como *pivotes*, y el índice se construye calculando la distancia entre éstos y cada elemento de la base de datos. Durante la búsqueda, se calcula la distancia de la consulta q a cada pivote y de este modo algunos elementos de la colección pueden descartarse utilizando la desigualdad triangular y las distancias precalculadas a los pivotes. Por ejemplo, si (q, r) es una consulta, y $x \in \mathbb{U}$ un objeto de la colección, podemos descartar directamente x si $|d(p_i, x) - d(p_i, q)| > r$ para algún pivote p_i. Los métodos basados en pivotes más conocidos son *Burkhard-Keller-Tree* (BKT) [5], *Fixed-Queries Tree* (FQT) [2], *Fixed-Height FQT* (FHQT) [1], *Fixed-Queries Array* (FQA) [7], *Vantage Point Tree* (VPT) [14] y sus variantes [3] [15], *Approximating and Eliminating Search Algorithm* (AESA) [13] y LAESA (*Linear AESA*) [10].

Los *métodos basados en clustering* particionan el espacio métrico en un conjunto de regiones de equivalencia, cada una de ellas representada por un centro de *cluster*. En la búsqueda, se descartan regiones completas dependiendo de la distancia de su centro de *cluster* a la consulta. Los métodos basados en *clustering* más importantes son *Bisector Tree* (BST) [9], *Generalized-Hyperplane Tree* (GHT) [12], *Geometric Near-neighbor Access Tree* (GNAT) [4] o *Spatial Approximation Tree* (SAT) [11]. En [8] y [16] puede encontrar dos excelentes revisiones que tratan en profundidad el problema de la búsqueda en espacios métricos y presentan con detalle estos algoritmos.

En este artículo presentamos *Sparse Spatial Selection* (SSS), un nuevo método de selección de pivotes que permite trabajar con funciones de distancia continuas y colecciones dinámicas, además de adaptarse bien a memoria secundaria. Es un método dinámico, en el sentido de que la colección puede estar vacía inicialmente y/o crecer o decrecer con el tiempo. La aportación más importante del método es la estrategia de selección de pivotes que genera un número de pivotes que es función de la dimensión intrínseca del espacio, algo interesante tanto desde el punto de vista teórico como desde el punto de vista práctico. Además, la técnica de selección de pivotes es dinámica y capaz de adaptarse a los nuevos elementos insertados en la colección conservando la eficiencia del índice en la búsqueda.

El resto del artículo está estructurado de la siguiente forma: la sección 2 describe el problema de la selección de pivotes y su importancia en la eficiencia de los métodos basados en pivotes. La sección 3 presenta el método propuesto en este trabajo, y en la sección 4 se muestran y discuten los resultados experimentales obtenidos en distintas pruebas realizadas para estudiar el comportamiento del método. Por último, la sección 5 presenta las conclusiones obtenidas y futuras líneas de trabajo.

2. TRABAJOS PREVIOS EN SELECCIÓN DE PIVOTES

Es evidente que el conjunto concreto de objetos seleccionados como pivotes influye en la eficiencia del método en la búsqueda por similitud. El número de pivotes, su "ubicación" en el espacio métrico y la "ubicación" de cada uno de ellos con respecto a los demás pivotes determinan la capacidad del índice para descartar elementos sin compararlos directamente con la consulta. La mayoría de los métodos de búsqueda basados en pivotes seleccionan los pivotes de forma completamente aleatoria. Además, no se conoce ninguna indicación para determinar el número óptimo de pivotes, un parámetro que depende del espacio métrico concreto con que estamos trabajando. Estas cuestiones, a las que quizá no se ha dado la importancia que merecen, hacen que la calidad de los resultados obtenidos dependan del espacio métrico concreto y del azar. En trabajos anteriores se han propuesto varias heurísticas para la selección de pivotes. Por ejemplo, en [10] se propone seleccionar como pivotes los objetos que maximicen la suma de las distancias a los pivotes ya seleccionados. En [14] y [4] se siguen heurísticas que tratan de obtener pivotes lejanos entre sí. En [6] se trata en profundidad este tema y se demuestra empíricamente que este factor afecta al rendimiento del método.

La aportación más importante de [6] es un criterio para comparar la eficiencia de dos conjuntos de pivotes del mismo tamaño. Sea $\{p_1, p_2, \ldots, p_k\}$, con $p_i \in \mathbb{U}$, un conjunto de pivotes. Dado un objeto $u \in \mathbb{U}$, denotamos por $[u] = (d(u, p_1), d(u, p_2), \ldots, d(u, p_k))$ a la tupla formada por las distancias de u a cada pivote. Tenemos así un espacio $\mathbb{P} = \{[u] \ / \ u \in \mathbb{X}\}$, que resulta ser además un espacio vectorial $\mathbb{R}^k$. Sobre los elementos de este espacio podemos definir la función de distancia $D_{\{p_1,p_2,\ldots,p_k\}}([x],[y]) = \max_{1\leq i\leq k} |d(x,p_i) - d(y,p_i)|$. Así, tenemos un espacio métrico $(\mathbb{P}, L_\infty)$. En estas condiciones, dada una consulta q y un radio r, la condición para descartar $u \in \mathbb{U}$ puede traducirse de $|d(p_i,u) - d(p_i,q)| > r$ para algún pivote p_i, en $D_{\{p_1,p_2,\ldots,p_k\}}([q],[u]) > r$. Un conjunto de pivotes es mejor cuantos más objetos pueda descartar. Entonces, un conjunto de pivotes es mejor que otro si hace mayor la probabilidad de $D_{\{p_1,p_2,\ldots,p_k\}}([q],[u]) > r$. Denotando por μ_D a la media de la distribución de D, dicha probabilidad aumenta al maximizar μ_D. Así, el criterio propuesto por los autores establece que el conjunto de pivotes $\{p_1, p_2, \ldots, p_k\}$ es mejor que el conjunto $\{p'_1, p'_2, \ldots, p'_k\}$ si $\mu_{\{p_1,p_2,\ldots,p_k\}} > \mu_{\{p'_1,p'_2,\ldots,p'_k\}}$.

En [6] se presentan varias estrategias de selección basadas en el criterio de eficiencia anterior. La primera, conocida como *Random*, selecciona un conjunto aleatorio de pivotes. La siguiente, denominada *Selection* consiste en seleccionar N grupos aleatorios de pivotes y quedarse con el que presente un mayor valor para μ_D. También proponen *Incremental*, una selección incremental en la que el siguiente pivote seleccionado es el objeto que maximice el valor de μ_D al unirlo a los pivotes ya seleccionados. Por último, proponen una estrategia iterativa denominada *Local Optimum*, en la que se parte de un conjunto aleatorio de pivotes y en cada paso se reemplaza por otro el pivote que menos aporte a μ_D. El rendimiento obtenido con muchas de estas técnicas depende en gran medida del espacio métrico concreto con el que estamos trabajando. La conclusión de estos trabajos es que los buenos pivotes deben estar distantes entre sí y también del resto de objetos de la base de datos, aunque esta condición no asegura siempre que sean buenos pivotes.

Una cuestión que no se ha tratado hasta ahora es la forma de determinar el número óptimo de pivotes, el valor de k. La eficiencia de la búsqueda presenta una fuerte dependencia de este parámetro. Además, su valor óptimo puede diferir mucho en distintos espacios métricos. En todas las técnicas de selección de pivotes que hemos visto, este valor debe ser fijado de antemano. En [6] se realizan los cálculos para diferentes números de puntos y de ese modo se obtiene el número óptimo de pivotes.

Los datos muestran que el número óptimo de puntos varía muchísimo de un espacio métrico a otro, y tiene una importancia enorme en la eficiencia del método, de ahí el interés de ajustarlo lo mejor posible.

3. NUESTRA PROPUESTA

En esta sección presentamos un nuevo método de búsqueda basado en pivotes. La principal aportación de nuestro método es la estrategia de selección de pivotes. A diferencia de otros, en los que los pivotes de referencia se eligen de forma aleatoria, nuestro método selecciona un conjunto dinámico de pivotes bien distribuidos en el espacio, lo que permite descartar más objetos en la búsqueda. Además el conjunto de pivotes es dinámico, en el sentido de que es capaz de adaptarse al crecimiento de la base de datos sin que su eficacia quede reducida. Veremos en primer lugar la estrategia de generación de pivotes, sus características y su relación con la dimensión intrínseca del espacio métrico. Después exponemos el proceso de construcción del índice y el comportamiento de éste ante el crecimiento de la base de datos, para finalizar explicando la operación de búsqueda.

3.1. Estrategia de selección de pivotes

Sea $(\mathbb{X}, d)$ un espacio métrico, $\mathbb{U} \subset \mathbb{X}$ una colección de objetos y M la distancia máxima entre dos objetos cualesquiera, $M = \max\{\, d(x,y) \,/\, x, y \in \mathbb{U} \,\}$. Inicialmente el conjunto de pivotes está formado por el primer elemento de la colección. Después, para cada elemento $x_i \in \mathbb{U}$, x_i se selecciona como pivote si y sólo si está a una distancia mayor o igual que $M \times \alpha$ de todos los pivotes ya seleccionados, siendo α una constante cuyos valores óptimos están en torno a $0{,}4$ como veremos más adelante. Es decir, un objeto de la colección se toma como pivote si está situado a más de una fracción de la distancia máxima de todos los pivotes. Por ejemplo, si $\alpha = 0{,}5$, un objeto se selecciona como pivote si está a más de la mitad de la distancia máxima de todos los pivotes. El siguiente pseudocódigo resume este proceso de selección de pivotes:

$PIVOTES \leftarrow \{x_1\}$
for all $x_i \in \mathbb{U}$ **do**
  if $\forall\, p \in PIVOTES,\ d(x_i, p) \geq M \times \alpha$ **then**
    $PIVOTES \leftarrow PIVOTES \cup \{x_i\}$
  end if
end for

Parece evidente que los pivotes seleccionados de esta forma estarán algo distantes entre sí (a más de $M \times \alpha$), algo que muchos autores señalan como una característica deseable del conjunto de pivotes [6]. Pero nuestra estrategia de selección va más allá. Al exigir que la distancia entre dos pivotes cualesquiera sea mayor o igual que $M \times \alpha$, nos aseguramos de que estén bien distribuidos en el espacio. Es importante tener en cuenta que nuestros pivotes no están muy distantes entre sí, ni tampoco distantes de los demás objetos de la colección, dos condiciones que según trabajos anteriores deben cumplir los buenos pivotes. Nuestra hipótesis es que, estando bien distribuidos en el espacio, el conjunto de pivotes será capaz de descartar más objetos en la búsqueda.

Algo muy importante de esta estrategia de selección de pivotes es su carácter dinámico y adaptativo. El conjunto de pivotes se adapta al crecimiento de la base de datos. Cuando se inserta un nuevo elemento x_i, se compara con los pivotes ya seleccionados y se añade a este conjunto si es necesario.

Así, el conjunto de pivotes sigue estando bien distribuido por todo el espacio métrico. De hecho, la colección podría estar inicialmente vacía, algo muy interesante de cara a la aplicación práctica de este método. De esta forma, el hecho de que la colección crezca no supone una reducción en el rendimiento del algoritmo.

3.2. Dimensión intrínseca del espacio

En un espacio vectorial, la dimensión del espacio es el número de componentes de cada vector. Aunque los espacios métricos generales no tienen una dimensionalidad explícita, se habla de su *dimensionalidad intrínseca* siguiendo la misma idea que en espacios vectoriales. Este interesante concepto permite distinguir espacios métricos de baja y alta dimensionalidad. La eficiencia de los algoritmos de búsqueda por similitud es peor en los espacios métricos con una dimensionalidad elevada. Por ejemplo, cualquier método de búsqueda se comportará peor en un espacio vectorial de dimensión 15 que en un espacio vectorial de dimensión 10. Esto se ha comprobado empíricamente en muchos trabajos realizados en espacios métricos. En [8] se analiza con detalle este concepto y su influencia en la eficiencia de los métodos de búsqueda. En dicho artículo se propone una forma de aproximar la dimensionalidad intrínseca como $\rho = \frac{\mu}{\sigma^2}$, siendo μ y σ^2 la media y varianza del histograma de distancias entre puntos del espacio métrico $(\mathbb{X}, d)$ respectivamente.

Otro aspecto importante de nuestro método es que el número de pivotes seleccionados no depende del tamaño de la colección, sino de la dimensión intrínseca del espacio métrico. Por ejemplo, con $\alpha = 0{,}5$, en un espacio de una dimensión podríamos tener sólo dos pivotes, en dos dimensiones podríamos tener como mucho cuatro pivotes, etc. Esta cuestión no ha sido considerada por ningún método de búsqueda anterior. Así, una ventaja de nuestro método de selección de pivotes es que el número de pivotes extraídos se adapta a la dimensionalidad intrínseca del espacio aunque ésta sea desconocida. Esto es especialmente relevante cuando se trabaja con espacios en los que no es posible ni tan siquiera aproximar su dimensionalidad para determinar el número adecuado de pivotes. Si el número de pivotes es demasiado bajo, puede que no sea capaz de cubrir todas las dimensiones del espacio, haciendo la búsqueda menos eficiente. Si nuestra propuesta genera un número de pivotes que depende de la dimensionalidad del espacio, el número de pivotes seleccionados debería crecer rápidamente con los primeros objetos de la colección, crecer más lentamente después y llegar a estabilizarse. En la sección 4 presentamos resultados experimentales que confirman nuestra hipótesis.

3.3. El parámetro α

Si bien es cierto que en nuestro método no es necesario prefijar de antemano el número de pivotes a seleccionar como en [6], también es cierto que hay que prefijar el valor de α y que a su vez dicho valor condiciona el número de pivotes. Sin embargo, el valor de α debe estar entre $0{,}35$ y $0{,}40$, dependiendo de la dimensionalidad de la colección. La figura 1 muestra el número de evaluaciones de la función de distancia realizados en función del parámetro α para espacios vectoriales de dimensión 8, 10, 12 y 14. En la gráfica se puede observar que el mejor resultado se obtiene siempre para valores de α situados entre $0{,}35$ y $0{,}40$, aunque la eficiencia del método es prácticamente la misma para todos los valores de α incluidos en ese intervalo. También se puede observar que cuando $\alpha > 0{,}40$, el número de evaluaciones de la función de distancia aumenta más en los espacios de mayor dimensionalidad. Este resultado se debe a que un aumento de α supone una reducción del número de pivotes, y esto se hace notar más en espacios con una dimensionalidad mayor.

Estas pruebas demuestran algunas de las ventajas más importantes de nuestro método. Nuestra estrategia de selección de pivotes es computacionalmente más sencilla y eficiente que otras existentes. Además, nuestros pivotes están distantes entre sí, pero no están distantes de los objetos de la colección (es decir, no son *outliers*). Sin embargo, hemos logrado alcanzar una eficiencia similar a las mejores técnicas existentes hasta este momento sin tener que prefijar de antemano el número de pivotes a utilizar. Nuestro método determina por sí sólo el número de pivotes adecuado a la complejidad del espacio métrico, basándose sólo en la distancia máxima entre dos objetos cualesquiera de la colección.

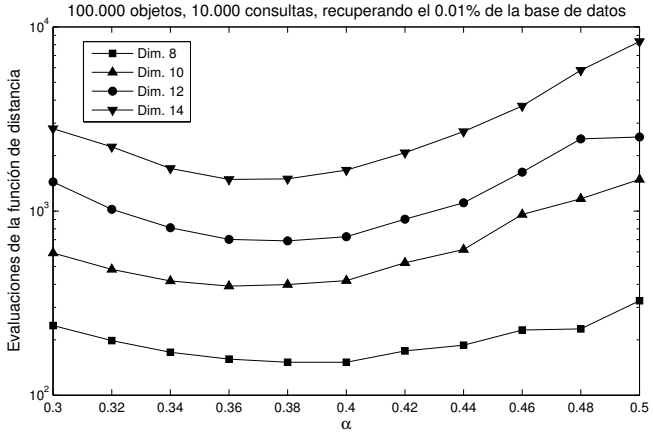


Figura 1: Número de evaluaciones de la función de distancia para espacios vectoriales de distinta dimensión, en función de α.

3.4. Construcción del índice y crecimiento de la colección

Una de las principales ventajas de nuestro método es su carácter dinámico. Así, para describir el proceso de construcción del índice partimos de que la colección está vacía inicialmente. El primer objeto insertado en la base de datos, u_1, es el primero en ser seleccionado como pivote, p_1. Cuando se inserta un nuevo objeto en la base de datos, se calcula y almacena su distancia a todos los pivotes que ya forman parte del índice. Si su distancia a todos ellos es mayor o igual que $M \times \alpha$, se añade al conjunto de pivotes. En este caso se calcula y almacena la distancia del nuevo pivote a todos los objetos de la base de datos. De esta forma, el conjunto de pivotes no tiene que ser prefijado de antemano sobre una colección inicial de objetos, sino que crece al mismo tiempo que crece la base de datos. Esta implementación permite que la estructura sea totalmente dinámica, y que el conjunto de pivotes se adapte a los objetos que se van insertando. Además, asegura que aunque la colección crezca, los pivotes están bien distribuidos en el espacio métrico. De esta forma, el método conservará su eficiencia a pesar del crecimiento de la colección.

3.5. Búsqueda

Para finalizar esta sección describimos cómo utilizar el índice en la operación de búsqueda por rango, ya que los demás tipos de búsqueda pueden implementarse en términos de ésta. Sea (q, r) una consulta. El primer paso consiste en calcular la distancia de q a todos los pivotes del índice. Hecho esto, se descartan todos los elementos $x_i \in \mathbb{U}$ tales que $|d(x_i, p_j) - d(q, p_j)| > r$, siendo p_j un pivote, ya que por la desigualdad triangular ($d(x, y) \leq d(x, z) + d(z, y)$), si se da esta condición su

distancia a q será $d(x_i, q) > r$. Los objetos que no pueden descartarse mediante esta condición constituyen la lista de candidatos, $\{u_1, u_2, \ldots, u_m\} \subseteq \mathbb{U}$ y deben compararse directamente con la consulta.

La complejidad de la búsqueda se mide por el número de evaluaciones de la función de distancia. Por una parte están las comparaciones de q con los pivotes, que constituyen la *complejidad interna* del algoritmo. Hecho esto hay que comparar la consulta con los objetos de la lista de candidatos. Estas comparaciones constituyen la *complejidad externa* del algoritmo. La *complejidad total* viene dada por la suma de las dos anteriores [8].

4. RESULTADOS EXPERIMENTALES

Para probar el comportamiento del algoritmo se han utilizado varias colecciones de datos en distintas situaciones. En primer lugar se realizaron pruebas con conjuntos sintéticos de puntos aleatorios en espacios vectoriales de dimensión $k = 8$, $k = 10$, $k = 12$ y $k = 14$. Aunque son objetos de un espacio vectorial, no se ha utilizado esta información en las pruebas. Se les ha considerado objetos de un espacio métrico general. La ventaja de probar el algoritmo con este tipo de datos es que podemos estudiar su comportamiento en espacios de diferente dimensionalidad intrínseca. La función de distancia utilizada con estos conjuntos de datos fue la distancia euclídea. Además, se ha probado nuestro método con espacios métricos reales: diccionarios de palabras en inglés y español utilizando la distancia de edición como función de distancia.

4.1. Número de pivotes generados en función de la dimensionalidad

En la sección anterior destacábamos que nuestro método genera de forma dinámica un número de pivotes que está en función de la dimensionalidad del espacio, y no del número de elementos de la base de datos. Para comprobar la validez de esta hipótesis utilizamos colecciones de $1{,}000{,}000$ de vectores de dimensiones $k = 8$, $k = 10$, $k = 12$ y $k = 14$. Para cada una de ellas se obtuvo el número de pivotes seleccionados en función del número de objetos insertados en la colección, fijando el parámetro α a $0{,}5$.

k	n, tamaño de la colección ($\times 10^3$)									
	100	200	300	400	500	600	700	800	900	1000
8	16	17	19	20	21	22	22	22	22	22
10	20	24	28	29	30	30	30	30	30	30
12	44	50	53	54	55	57	58	58	58	58
14	56	62	69	71	73	79	80	80	82	82

Tabla 1: Número de pivotes seleccionados en espacios vectoriales de dimensión $k = 8$, $k = 10$, $k = 12$ y $k = 14$, en función del tamaño de la colección.

La tabla 1 muestra los resultados obtenidos en las pruebas. En primer lugar podemos observar que el número de objetos seleccionados como pivotes es mayor al aumentar la dimensión del espacio vectorial. Esto demuestra que el número de pivotes depende de la dimensionalidad intrínseca del espacio métrico. Fijémosnos ahora en el número de pivotes en función del tamaño de la colección. En todos los espacios de prueba el número de pivotes aumenta muy rápido con los primeros objetos de la base de datos. A partir de ahí, crece mucho más lentamente hasta llegar a estabilizarse. Evidentemente, cuando la colección tiene pocos elementos, el número de pivotes depende del tamaño de

ésta. Sin embargo, cuando la colección alcanza un tamaño dado, al insertar elementos en la base de datos llega un momento en que, aunque sigamos insertando objetos, no se seleccionarán más pivotes. Esto es así porque el conjunto de pivotes existentes es capaz de cubrir todo el espacio y capturar su dimensionalidad. Con estos resultados se demuestra que el número de pivotes está en función de la dimensionalidad intrínseca del espacio, y no del tamaño de la colección.

4.2. Eficiencia de la búsqueda en espacios vectoriales

En esta sección mostramos los resultados obtenidos en las pruebas realizadas para evaluar la eficiencia del algoritmo en la operación de búsqueda. En el primer conjunto de pruebas se han utilizado espacios vectoriales de dimensiones 8, 10, 12 y 14, cada uno de ellos con $100{,}000$ objetos distribuidos de manera uniforme en un hipercubo de lado 1. Se obtuvo el número medio de evaluaciones de la función de distancia en un total de $10{,}000$ búsquedas. El número medio de elementos recuperado en cada una de ellas es el $0{,}01\,\%$ de la base de datos. Para poder evaluar el comportamiento del algoritmo, hemos comparado los resultados con los obtenidos con las técnicas de selección de pivotes propuestas en [6].

Método	$k = 8$		$k = 10$		$k = 12$		$k = 14$	
	pivotes	eval. d	pivotes	eval. d	pivotes	eval. d	pivotes	eval. d
Random	85	213	190	468	460	998	1000	2077
Selection	85	204	200	446	360	986	800	2038
Incremental	65	157	150	335	300	714	600	1458
Loc. Opt. A	70	155	150	333	300	708	600	1448
Loc. Opt. B	60	157	150	369	300	881	760	1930
SSS	57	151	148	389	258	689	598	1452

Tabla 2: Número mínimo de evaluaciones de d con distintas estrategias de selección de pivotes en espacios vectoriales.

La tabla 2 muestra el número mínimo de evaluaciones de d al que se ha llegado con cada estrategia de selección de pivotes, y el número de pivotes utilizados. Se puede observar que el número de evaluaciones de la función de distancia obtenidos con nuestro método está siempre en torno al mejor resultado obtenido con las estrategias propuestas en [6]. En algunos casos llegamos a menos evaluaciones de d con un número menor de pivotes. En cambio en otros, nuestro método realiza más evaluaciones, aunque con un número también menor de pivotes. Estos resultados demuestran que la estrategia de selección de pivotes propuesta presenta una eficiencia similar a otras más complejas. En los resultados de las pruebas también se puede observar que el número de pivotes que selecciona nuestro método por sí sólo es muy similar al número óptimo de pivotes para otras estrategias de selección de pivotes.

4.3. Eficiencia de la búsqueda en espacios métricos

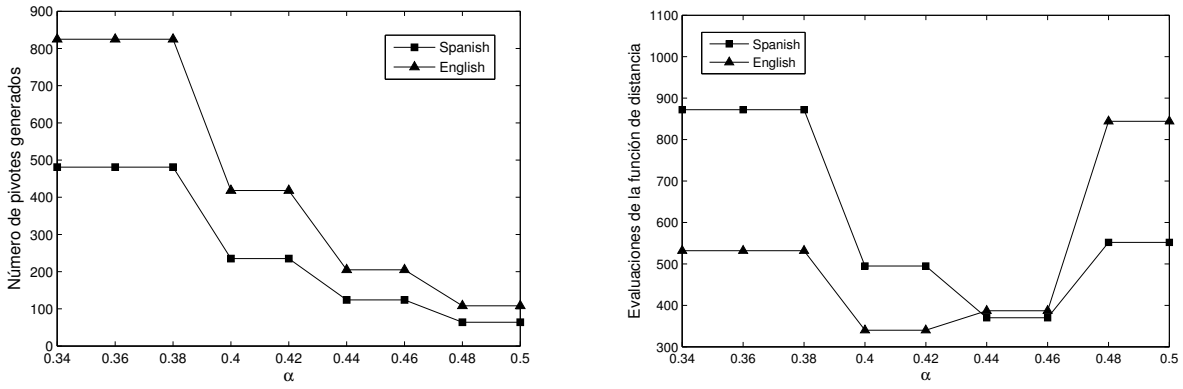
Además de realizar pruebas con espacios vectoriales distribuidos de manera uniforme, también comparamos nuestro método con las estrategias existentes utilizando espacios métricos "reales". En concreto, se han utilizado dos colecciones de palabras. La primera de ellas contiene $69{,}069$ palabras

del diccionario inglés, y la segunda $51{,}589$ palabras del diccionario español. Como función de distancia se ha utilizado la distancia de edición. Se ha utilizado el $10\,\%$ de la base de datos como consultas, con un radio $r = 2$, que recupera una media del $0{,}02\,\%$ de la colección. La tabla 3 muestra el número mínimo de evaluaciones de d al que se ha llegado con nuestra estrategia de selección de pivotes y las propuestas en [6], para la colección de palabras del diccionario inglés. En este caso el resultado obtenido con nuestra estrategia de selección de pivotes es mejor que el obtenido con cualquier otra. De la misma forma que en el caso de los espacios vectoriales, el número de pivotes que genera nuestro método es casi igual al número óptimo de pivotes usado en otras estrategias, que lo han obtenido por prueba y error.

Método	pivotes	eval. d
Random	200	443
Good pivots	200	389
Outliers	200	429
SSS	205	370

Tabla 3: Número mínimo de evaluaciones de d en una colección de palabras.

Finalmente se ejecutaron las mismas pruebas con otra colección de palabras, formada por $51{,}589$ palabras del diccionario español. Como en el caso del diccionario inglés, se utilizaron como consultas aproximadamente el $10\,\%$ de la base de datos ($5{,}200$ consultas), con un radio de búsqueda $r = 2$ para recuperar aproximadamente el $0{,}02\,\%$ de la base de datos en cada una de ellas. La figura 2 muestra los resultados obtenidos en estas pruebas.



(a) Número de pivotes generados en función de α

(b) Número de evaluaciones de d en función de α

Figura 2: Número de pivotes generados (izquierda) y número de evaluaciones de la función de distancia para distintos valores de α (derecha), en colecciones de palabras del diccionario inglés y español.

En la figura 2 se pueden apreciar diferencias importantes tanto en el número de pivotes seleccionados como en el número de evaluaciones de la función de distancia. En el caso del diccionario español se selecciona un número menor de pivotes que en el caso del diccionario inglés. Aún así, el número óptimo de evaluaciones de d es muy similar en los dos casos, y se obtiene para valores muy

próximos de α. Este resultado podría parecer extraño en un primer momento ya que, al fin y al cabo, los dos conjuntos de prueba son colecciones de palabras. Sin embargo, los dos espacios tienen una complejidad diferente. La colección de palabras inglesas tiene un número un poco mayor de elementos. Además, la distribución de distancias de palabras es diferente en los dos diccionarios, algo que tiene una influencia importante en el resultado de una consulta. Pese a esas diferencias, nuestro método ha seleccionado un número de pivotes adecuado para cada espacio, obteniendo la misma eficiencia de búsqueda en ambos. Este resultado es una prueba más de la capacidad del método propuesto de adaptarse a la complejidad del espacio métrico con que estamos trabajando.

5. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo hemos propuesto un método dinámico de búsqueda basado en pivotes. La aportación más importante del método es la estrategia dinámica de selección de pivotes, capaz de adaptarse al crecimiento de la base de datos sin que esto suponga una reducción en la eficiencia del método. Además, la estructura del índice permite que éste pueda almacenarse de forma eficiente en memoria secundaria. El objetivo que persigue nuestra forma de generar los pivotes es que éstos estén bien distribuidos por todo el espacio métrico. Esto va más allá de la idea de que los pivotes estén distantes entre sí, característica necesaria según muchos autores para que el conjunto de pivotes consiga descartar un gran número de objetos en la búsqueda.

Nuestros resultados experimentales demuestran que el método genera un número de pivotes que depende de la dimensionalidad intrínseca del espacio métrico y no del número de elementos de la colección. Además ese número de pivotes está muy próximo al número óptimo con otras estrategias. Esto hace que no sea necesario especificar a priori el número de pivotes necesarios para la estructura, algo que ningún método existente consideraba. El número de pivotes seleccionados se adapta a la complejidad del espacio, evitando seleccionar pivotes innecesarios que puedan llegar a reducir la eficiencia de la búsqueda. La eficiencia de nuestro método en espacios vectoriales es similar a los resultados obtenidos en trabajos anteriores. Sin embargo, las pruebas reflejan que nuestro método es más eficiente que otros ya existentes en la búsqueda por similitud en espacios métricos generales.

Nuestra línea de trabajo todavía mantiene abiertas algunas cuestiones que abordaremos en un futuro. En primer lugar sería interesante evaluar el comportamiento de SSS con otros espacios métricos reales, como colecciones de documentos de texto o imágenes. Por otra parte, también tenemos pensado trabajar en refinamientos de la estrategia de selección de pivotes y en el desarrollo de técnicas que traten de estimar el valor óptimo de α de forma automática en función de la complejidad del espacio.

Agradecimientos. Nos gustaría agradecer a Benjamín Bustos, Gonzalo Navarro y Edgar Chávez el habernos proporcionado las colecciones de prueba y los resultados obtenidos con sus estrategias de selección de pivotes.

REFERENCIAS

[1] Ricardo Baeza-Yates. Searching: an algorithmic tour. *Encyclopedia of Computer Science and Technology*, 37:331–359, 1997.

[2] Ricardo Baeza-Yates, Walter Cunto, Udi Manber, and Sun Wu. Proximity matching using fixed-queries trees. In M. Crochemore and D. Gusfield, editors, *Proceedings of the 5th Annual Sym-*

posium on Combinatorial Pattern Matching, pages 198–212, Asilomar, C.A., 1994. Springer-Verlag, Berlin.

[3] Tolga Bozkaya and Meral Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In ACM Press, editor, *Proceedings of the ACM International Conference on Management of Data (SIGMOD 1997)*, pages 357–368, Tucson, Arizona, USA, May 1997.

[4] Sergey Brin. Near neighbor search in large metric spaces. In *21st conference on Very Large Databases*, 1995.

[5] Walter A. Burkhard and Robert M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236, April 1973.

[6] Benjamin Bustos, Gonzalo Navarro, and Edgar Chávez. Pivot selection techniques for proximity search in metric spaces. In *SCCC 2001, Proceedings of the XXI Conference of the Chilean Computer Science Society*, pages 33–40. IEEE Computer Science Press, 2001.

[7] Edgar Chávez, Jose Luis Marroquín, and Gonzalo Navarro. Overcoming the curse of dimensionality. In *European Workshop on Content-based Multimedia Indexing (CBMI'99)*, pages 57–64, 1999.

[8] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and Jose Luis Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.

[9] Iraj Kalantari and Gerard McDonald. A data structure and an algorithm for the nearest point problem. *IEEE Transactions on Software Engineering*, 9:631–634, 1983.

[10] Luisa Micó, José Oncina, and R. Enrique Vidal. A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear pre-processing time and memory requirements. *Pattern Recognition Letters*, 15:9–17, 1994.

[11] Gonzalo Navarro. Searching in metric spaces by spatial approximation. In *Proceedings of String Processing and Information Retrieval (SPIRE'99)*, pages 141–148. IEEE Computer Science Press, 1999.

[12] Jeffrey K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40:175–179, 1991.

[13] Enrique Vidal. An algorithm for finding nearest neighbors in (aproximately) constant average time. *Pattern Recognition Letters*, 4:145–157, 1986.

[14] Peter Yianilos. Data structures and algorithms for nearest-neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321. ACM Press, 1993.

[15] Peter Yianilos. Excluded middle vantage point forests for nearest neighbor search. In *Proceedings of the 6th DIMACS Implementation Challenge: Near neighbor searches (ALENEX 1999)*, Baltimore, Maryland, USA, January 1999.

[16] Pavel Zezula, Giuseppe Amato, Vlatislav Dohnal, and Michal Batko. *Similarity search. The metric space approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.