Reducción del Tamaño del Índice en Búsquedas por Similitud sobre Espacios Métricos*

Luis G. Ares, Nieves R. Brisaboa, María F. Esteller,
Óscar Pedreira y Ángeles S. Places

Laboratorio de Bases de Datos, Facultade de Informática, Universidade da Coruña
Campus de Elviña s/n, 15071 A Coruña
{lgares,brisaboa,mfesteller,opedreira,asplaces}@udc.es

Resumen En problemas de recuperación de información se plantea frecuentemente la realización de búsquedas por similitud sobre espacios métricos. Se utilizan funciones de distancia costosas sobre grandes bases de datos, que requieren un equilibrio entre la cantidad de información almacenada en el índice y el número de evaluaciones de la función de distancia. Las soluciones adoptadas son métodos basados en pivotes y métodos basados en clustering. Los primeros obtienen menor número de comparaciones, pero presentan necesidades de espacio mucho mayores, por lo que se han propuesto alternativas que pretenden reducir esa cantidad de espacio. En este trabajo analizamos la utilidad de los pivotes en función de su proximidad al objeto, y como consecuencia, proponemos un nuevo método basado en pivotes que requiere una cantidad de espacio semejante a la que necesitan los métodos basados en clustering. Ofrecemos resultados experimentales que demuestran su idoneidad cuando se utiliza la misma cantidad de memoria.

1 Introducción

La búsqueda por similitud es una operación que está cada vez más presente en un gran número de aplicaciones que realizan el tratamiento de tipos de datos complejos, como es el caso de datos semiestructurados, información biológica y bases de datos multimedia. La recuperación basada en contendido es un problema habitual en estos escenarios, siendo la búsqueda por similitud una de las estrategias más adecuadas para solucionarlo. Un ejemplo de búsqueda por similitud consiste en obtener la imagen más semejante a una dada, pero su aplicación se extiende a una gran variedad de áreas, como los editores de texto y los grandes buscadores, donde se requiere encontrar palabras que se asemejan a otras para obtener resultados ante criterios de búsquedas o para efectuar comprobaciones sintácticas, el reconocimiento de patrones donde se realiza una detección y catalogación de documentos, imágenes o sonidos que presentan diversas similitudes, y la bioinformática donde se necesita reconocer y estudiar secuencias de ADN.

* Este trabajo ha sido parcialmente financiado por el Ministerio de Educación y Ciencia (PGE y FEDER, ref. TIN2006-15071-C03-03) y por la Xunta de Galicia (ref. 2006/4)

El problema de la búsqueda por similitud puede formalizarse mediante el concepto de *espacio métrico* que es un par (X, d) formado por un *universo* de objetos válidos X y una función de distancia $d : X \times X \longrightarrow \mathbb{R}^+$ que cumple tres propiedades fundamentales: ser *estrictamente positiva* ($d(x, y) \geq 0$, y $d(x, y) = 0 \Leftrightarrow x = y$), ser *simétrica* ($d(x, y) = d(y, x)$) y verificar la *desigualdad triangular* ($d(x, y) \leq d(x, z) + d(z, y)$). La *base de datos* o *colección de objetos* sobre la que se realizan las búsquedas es un subconjunto finito $U \subseteq X$ de tamaño $n = |U|$.

Una *consulta* se expresa mediante un objeto a consultar $q \in X$ y un criterio de proximidad a ese objeto. El *conjunto resultado* es el conjunto de objetos de la colección que cumplen el criterio. La operación principal es la *búsqueda por rango* en la que se recuperan los objetos que están a una distancia de q menor o igual que un valor dado r. Otras operaciones pueden implementarse a partir de la anterior[1], destacando la *búsqueda de los k–vecinos más próximos*, que recupera los k objetos más similares a q.

Un espacio vectorial es un caso particular de espacio métrico en el que cada objeto está formado por un número de coordenadas reales. En este caso, como función de distancia d podría utilizarse alguna de la familia L_p definida sobre $\mathbb{R}^l$ como $L_p(x, y) = (\sum_{1 \leq i \leq l} |x_i - y_i|^p)^{1/p}$. Por ejemplo L_2 es la distancia euclídea. En una colección de cadenas de caracteres se puede utilizar la distancia de edición, que es el número de inserciones, borrados o sustituciones necesarias para transformar una cadena en la otra, teniendo así otro ejemplo de espacio métrico.

La búsqueda por similitud puede implementarse de forma trivial comparando el objeto a consultar con la totalidad de los objetos de la base de datos. Pero esta alternativa no es viable en términos prácticos, debido a que comparar los objetos supone un elevado coste computacional, máxime si la base de datos tiene un gran número de objetos. Por ello se han desarrollado métodos que construyen índices sobre la colección, y reducen el número de comparaciones entre objetos, utilizando la desigualdad triangular. De esta forma se pueden descartar elementos, sin compararlos directamente con la consulta.

Aunque la reducción en el número de evaluaciones de la función de distancia es el principal objetivo de estos métodos, hay otros factores de interés que afectan al rendimiento global del coste de la búsqueda, como son las necesidades de espacio para almacenar el índice (ya sea en memoria primaria o secundaria) y el tiempo de CPU adicional requerido para la carga y procesamiento del mismo. En unos casos los métodos son estáticos, en los que la colección no puede crecer una vez creado el índice; en otros son dinámicos y se adaptan a operaciones de inserción en una base de datos inicialmente vacía.

Los métodos de acceso en espacios métricos pueden clasificarse en dos grupos: los basados en clustering y los basados en pivotes[1]. Los métodos basados en clustering particionan el espacio y tratan de descartar regiones enteras durante la búsqueda. Los métodos basados en pivotes almacenan las distancias precalculadas de cada objeto en la base de datos a un conjunto de pivotes, y estas distancias se utilizan durante la búsqueda para descartar objetos del conjunto resultado. En los métodos basados en clustering, el índice suele ser una lista o un

árbol que representa la partición del espacio. Necesitan una pequeña cantidad de memoria para almacenar el índice, siendo el tiempo extra de CPU también muy pequeño. Los métodos basados en pivotes superan a los métodos basados en clustering en términos de cálculos de distancia, a cambio de necesitar cantidades de espacio muy significativas para almacenar las distancias precalculadas.

En trabajos anteriores se han estudiado diferentes estrategias para reducir la cantidad de espacio utilizado por los métodos basados en pivotes, al tratar de conservar su eficacia para descartar objetos del conjunto resultado. Puede hacerse, fundamentalmente, almacenando menos información en el índice o utilizando menor precisión. En ambos casos, la pérdida de información implica una reducción de eficacia en el proceso de descarte de objetos.

En este trabajo, analizamos la eficacia de la selección de pivotes para cada objeto en la base de datos, y la utilidad de almacenar las distancias asociadas. Estudiamos también el efecto del conjunto de pivotes iniciales en el rendimiento del índice. Basándonos en este análisis, se propone un nuevo método basado en pivotes que necesita una cantidad de espacio muy cercana o incluso menor, que la utilizada en algoritmos basados en clustering. Nuestra evaluación experimental muestra que la propuesta representa una estrategia competitiva, cuando se usa la misma cantidad de espacio.

El resto del documento está organizado de la siguiente manera: la sección 2 describe las propuestas anteriores para reducir la cantidad de espacio en los métodos basados en pivotes, y las técnicas para la selección efectiva de los pivotes. La sección 3 se centra en los objetivos de este trabajo y describe nuestra configuración experimental. La sección 4 presenta nuestro análisis de la eficacia de los pivotes. La sección 5 presenta el nuevo método que proponemos en este artículo y su evaluación experimental. Por último, la sección 6 presenta las conclusiones de este trabajo y posibles líneas de trabajo futuro.

2 Antecedentes y Trabajos Relacionados

Sea (X, d) un espacio métrico y $U \subseteq X$ una base de datos o colección de objetos de tamaño $n = |U|$. Los métodos basados en pivotes seleccionan un conjunto de m objetos $P = \{p_1, \ldots, p_m\}$ de la base de datos, denominados pivotes. En la fase de indexación, se calculan las distancias $d(x_i, p_j)$ de los objetos en la base de datos a los pivotes, y se almacenan en una estructura de datos apropiada. Dada una consulta (q, r), el objeto de consulta q se compara con los pivotes para obtener las distancias $d(q, p_j)$. Por abuso de lenguaje se utiliza indistintamente *distancia de un elemento al objeto de consulta* y *distancia de un elemento a la consulta*. Para cada elemento $x_i \in U$, podemos obtener una cota inferior de la distancia a la consulta, usando la desigualdad triangular. El objeto se descarta del conjunto resultado, sin compararlo con la consulta, si $d(x_i, q) \geq |d(x_i, p_j) - d(p_j, q)| > r$ para algún pivote $p_j \in P$. La complejidad de la búsqueda es la suma de la complejidad interna (comparaciones de la consulta con los pivotes) y de la complejidad externa (comparación de la consulta con los objetos que no resultan descartados).

Los métodos basados en pivotes se diferencian en la información que almacenan y en las estructuras de datos que utilizan. Así AESA [2] y LAESA [3] usan una tabla para almacenar las distancias de los objetos a los pivotes, mientras que otros como FQT [4], FQA [5] o VPT [6] usan estructuras de árbol.

Dos factores relacionados con estos métodos son de gran importancia en el rendimiento global de la búsqueda. El primero se refiere a los requerimientos de espacio del índice, frecuentemente muy elevados, que conllevan además un tiempo de CPU adicional, necesario para la carga y procesamiento de la información almacenada en el índice. Este es uno de los principales inconvenientes de los métodos basados en pivotes. El otro factor es la elección efectiva de los pivotes. La mayoría de los métodos seleccionan los pivotes al azar, pero se ha demostrado que el conjunto de pivotes determinado afecta al rendimiento [7], ya que la distribución de cada pivote con respecto a los demás, y al resto de los objetos en la base de datos, determina la capacidad del índice para descartar elementos del conjunto resultado. Ambos factores interactúan al considerar el número de pivotes elegidos, puesto que cuanto mayor sea el número de pivotes, mayores serán las necesidades de espacio del índice.

Los trabajos relacionados se han centrado en mejorar estos factores, intentado mantener la eficacia en el proceso de descarte de objetos.

2.1 Reducción de los Requerimientos de Espacio del Índice

Se han desarrollado varias estrategias para reducir el espacio del índice en los métodos basados en pivotes:

- Degradar el rango: Consiste en almacenar las distancias de los objetos a los pivotes con una precisión menor, con lo que se reduce el espacio necesario para el índice. Trabajando con distancias continuas -en el caso de distancias discretas es directo-, el rango de posibles valores se divide en varios intervalos y el índice almacena el intervalo en el que se encuentra cada distancia. Las distancias son ahora menos precisas, con lo que la cantidad de objetos desechados será menor. Este es el planteamiento utilizado por VPT [6], y puede aplicarse fácilmente en índices como FQA [5]. BAESA [8] propone algo semejante sobre un índice AESA [2] en el que el rango de distancias está dividido en 2^k intervalos, por lo que cada una de las $n \times n$ distancias de AESA se almacena utilizando k bits.
- Reducir el nivel de granularidad del cubo: Está orientada a índices con una estructura arbórea. Supone interrumpir la creación de subárboles, cuando se ha alcanzado un número reducido pero suficiente de elementos. El tamaño de la lista de candidatos aumenta, ya que determinadas áreas no forman parte del índice, pero este almacena menos distancias, con la consiguiente reducción de espacio.
- Reducir el ámbito de acción de los pivotes: Frente a métodos que almacenan las distancias de todos los objetos de la base de datos a cada uno de los pivotes, como AESA [2], LAESA [3] y SSS [9], esta alternativa propone guardar las distancias de cada pivote a un subconjunto de objetos en la base

de datos. De esta forma se reduce el ámbito de actuación del pivote. El índice almacena así menos distancias, pero se reducen las posibilidades de descartar un objeto del conjunto resultado. KVP [10] es un ejemplo de esta estrategia.

2.2 Elección Efectiva de los Pivotes

La elección de un conjunto de pivotes que sea realmente efectivo, junto a determinar el número más favorable de pivotes, son factores que influyen en el rendimiento de la búsqueda. Si incrementamos el número de pivotes, aumentan las posibilidades de descartar objetos, pero a costa de aumentar la complejidad interna y también los requerimientos de espacio del índice. Por otra parte, los datos experimentales muestran que el número óptimo de pivotes depende de las características del espacio métrico.

Diversos trabajos realizan propuestas diferentes para determinar la eficacia de un conjunto de pivotes. En [3] y [6] se indica que los pivotes más adecuados deberían estar lejos unos de otros, y también del resto de objetos de la base de datos. En [7] se propone un criterio formal para comparar la eficacia de dos conjuntos de pivotes del mismo tamaño.

Estos trabajos están orientados a determinar la eficacia de un conjunto de pivotes, para la totalidad de los objetos de una base de datos, y no estudian cuál es el mejor pivote para un objeto determinado. Celik [10] muestra empíricamente que dado un objeto y una consulta, los mejores pivotes son aquellos que están o muy cerca o muy lejos del objeto, aunque en [11] menciona que es posible encontrar distribuciones para las que los pivotes cercanos y lejanos, no serían la mejor elección.

Se han propuesto varias técnicas para seleccionar pivotes, entre ellas MaxMin [12] que maximiza la distancia mínima entre pivotes, Spacing [13] que selecciona pivotes con una correlación mínima y que maximizan la distancia entre objetos, Incremental [7] que elige de forma incremental un conjunto de pivotes que maximizan un criterio para comparar dos conjuntos de pivotes. Sparse Spatial Selection (SSS) [9] selecciona un conjunto de pivotes de forma dinámica, determinando si un elemento será o no un pivote, en el momento en el que se introduce en la base de datos. Considera que un nuevo elemento está lo suficientemente lejos, si su distancia a los pivotes es mayor que $M\alpha$, siendo M la distancia máxima entre dos objetos cualesquiera, y α una constante relacionada con la densidad de pivotes en el espacio. Con este método adaptativo, se garantiza que el número de pivotes depende de la complejidad del espacio, y no del número de objetos.

3 Objetivos de este Trabajo

En este artículo nos planteamos los siguientes objetivos, referentes a los métodos basados en pivotes para la búsqueda por similitud en espacios métricos:

- Analizar la eficacia de los pitoves para cada objeto de la base de datos. Para ello realizamos un análisis empírico con diferentes colecciones de datos.

– Basándonos en los resultados del análisis anterior, reducir en lo posible el número de distancias almacenadas en el índice, para cada objeto de la base de datos.

En los experimentos hemos utilizado varias colecciones de datos, tanto reales como sintéticos, disponibles en la *Metric Spaces Library* [14]: UV08, UV10, UV12 y UV14 son colecciones de 100.000 vectores sintéticos con distribución uniforme y de dimensión 8, 10, 12 y 14 respectivamente. English es una colección de 69.069 palabras extraídas del diccionario de inglés, que se comparan utilizando la distancia de edición. Nasa es una colección de 40.150 imágenes extraídas de los archivos de la NASA, representadas por vectores de dimensión 20 y comparadas usando la distancia euclídea.

Las evaluaciones se centraron en la búsqueda por rango. En English trabajamos con $r = 2$. Para las colecciones de vectores, el radio de búsqueda se determinó en una media del $0,01\%$ de la base de datos, para cada consulta.

4 Análisis de la Eficacia de los Pivotes

Consideramos dos cuestiones relativas a la selección de los pivotes más eficaces para cada uno de los objetos de la base de datos: en primer lugar, el conjunto inicial de pivotes, entre los que se eligen los que resultan más prometedores, o sea, los que se espera que obtengan un mejor resultado, de cara a los descartes, en la evaluación de una consulta; en segundo lugar, la eficacia de los diferentes pivotes para cada objeto.

Celik [10] muestra que los pivotes más cercano y más lejano a un objeto son los más prometedores. Esto nos lleva a necesitar un método de selección de pivotes que garantice que los pivotes estén bien distribuidos en el espacio, lo que hará que para cada objeto haya pivotes tanto cercanos como lejanos. Estas condiciones las cumple SSS [9], mientras que otros métodos, en particular los que eligen los pivotes de manera aleatoria, no garantizan que cada objeto disponga de pivotes realmente cercanos o lejanos. En el peor de los casos, todos los pivotes pueden estar a la mitad de la distancia máxima al objeto. En este trabajo utilizamos SSS como alternativa de selección de pivotes.

La siguiente cuestión a dilucidar es que, entre los pivotes seleccionados con SSS, ¿cuáles son las características de los que resultan más útiles para descartar un objeto? A raíz de los resultados de Celik [10], sabemos que los pivotes deben ser cercanos o lejanos al objeto. Pero, ¿cuántos pivotes cercanos y lejanos necesitamos para cada objeto, y que a la vez sean eficaces?

La tabla 1 muestra el número de pivotes ("Piv."), espacio ("Espac.", en MB), y evaluaciones de la función de distancia ("Eval. d") obtenidas al utilizar:

– SSS, almacenando todas las distancias de cada objeto a cada pivote ("SSS (α óptimo)").
– Una selección aleatoria de pivotes, almacenando todas las distancias de cada objeto a cada uno de los pivotes ("Random").

– SSS, almacenando solo las distancias a los dos pivotes más cercanos y a los dos más lejanos a cada objeto ("SSS ($\alpha = 0,25; 4\ dist$)").
– SSS, almacenando solo las distancias al pivote más cercano y al más lejano a cada objeto ("SSS ($\alpha = 0,25; 2\ dist$)").

Colec.	Random			SSS (α óptimo)			SSS (α = 0,25; 4 dist.)			SSS (α = 0,25; 2 dist.)		
	Piv.	Espac.	Eval. d	Piv.	Espac.	Eval. d	Piv.	Espac.	Eval. d	Piv.	Espac.	Eval. d
UV08	85	29,1828	211,78	53	18,1963	141,43	494	2,7485	1231,66	494	1,3752	3094,13
UV10	190	65,2321	468,23	176	60,4255	367,14	1461	2,7522	3011,61	1461	1,3789	5891,64
UV12	460	157,9303	998,13	250	85,8317	645,08	4303	2,7630	6803,09	4303	1,3897	9739,49
UV14	1000	343,3266	2077,44	491	168,5734	1381,64	10000	2,7848	14229,20	10000	1,4115	18160,91
English	200	27,5696	443,85	212	45,0553	354,89	3100	1,9089	7931,30	3100	0,9604	12373,45
Nasa	77	10,6143	276,34	55	7,4438	168,62	871	1,1061	1308,28	871	0,5547	1958,34

Tabla 1. Coste de la búsqueda con diferentes estrategias al seleccionar un conjunto inicial de pivotes.

Lo obtenido es casi una reproducción de los resultados de Celik [10], aunque como podemos observar, el método utilizado para seleccionar el conjunto inicial de pivotes tiene una gran incidencia en la eficacia, mientras que el espacio se ha reducido considerablemente. Por ejemplo, en el caso de UV12, el número de evaluaciones al utilizar dos pivotes es de aproximadamente 15 veces el número de comparaciones cuando se utilizan todos los pivotes, pero el espacio necesario al utilizar todos los pivotes es de aproximadamente 61 veces el espacio necesario cuando se utilizan solo dos. En el caso de UV14, una colección más compleja, la diferencia es aún mayor: el número de comparaciones cuando se utilizan dos pivotes es de aproximadamente 13 veces el número de evaluaciones necesarias cuando se utiliza la totalidad de los pivotes, pero el espacio utilizando todos los pivotes es de aproximadamente 119 veces el necesario utilizando solo dos. Podemos observar que en las colecciones más complejas, como UV14, English o Nasa, al pasar de 4 a 2 pivotes la eficacia se reduce un 25%, mientras que el espacio se reduce un 50%. Es decir, la pérdida de eficacia en evaluaciones de la función de distancia, es mucho menor, concretamente la mitad, que la reducción de espacio. Por lo tanto, pensamos que reducir drásticamente el número de pivotes a solo 2, es factible si se desean rebajar las necesidades de espacio del índice.

Una segunda cuestión que debemos preguntarnos es si es más eficaz utilizar el pivote más cercano o el más lejano para cada objeto. La figura 1 muestra, para cada colección, los porcentajes de los objetos que no son descartados ("No descartados"), los descartados solo por su pivote más cercano ("Solo por el más cercano"), los descartados solo por su pivote más lejano ("Solo por el más lejano"), y los descartados por ambos a la vez ("Por ambos"). En la figura podemos ver que la utilización del pivote más cercano, origina que se descarte la mayoría de los objetos, aunque un gran porcentaje puede ser descartado por la utilización, tanto del pivote más cercano como del más lejano. Solo en unos pocos casos el objeto resulta descartado por su pivote más lejano y no por el más cercano.

Con el fin de analizar estos datos y entender en qué casos se descarta el objeto solo por su pivote más lejano, calculamos la media y la desviación estándar de la distribución de distancias entre los objetos de cada colección. Posteriormente, para los casos en el que los objetos solo fueron descartados por su pivote más cercano ("Más cercano"), solo por el más lejano ("Más lejano") y por ambos ("Ambos"), calculamos la distancia media de los objetos en cada grupo, a los pivotes más cercano y más lejano. Los resultados se muestran en la tabla 2. Las distancias de los objetos a los pivotes se muestran en puntuaciones estándar (z) para poder realizar su comparación. Los pivotes más cercano y más lejano, se han obtenido utilizando SSS con $\alpha = 0,25$.

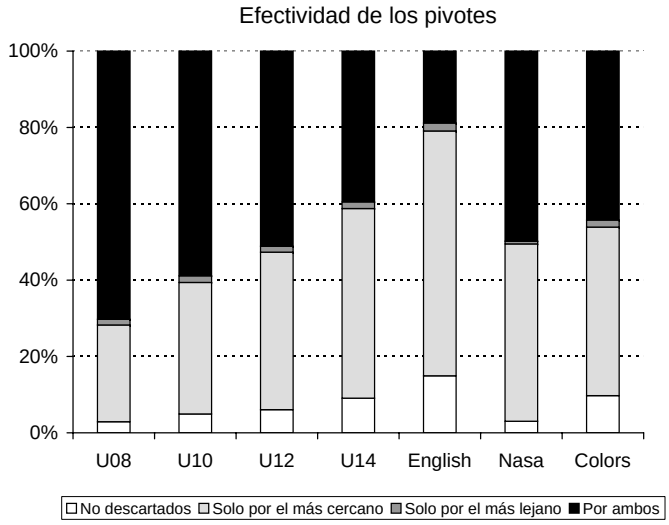

Fig. 1. Porcentajes de objetos descartados por el pivote más cercano, por el más lejano y por ambos.

Colec.	μ	σ	Más cercano		Más lejano		Ambos	
			$zd_{cercano}$	zd_{lejano}	$zd_{cercano}$	zd_{lejano}	$zd_{cercano}$	zd_{lejano}
UV08	1,5086	0,2452	-4,3989	0,9845	-4,1949	1,3923	-4,3989	1,4331
UV10	1,4032	0,2456	-3,7182	2,2671	-3,5147	2,6743	-3,7182	2,6743
UV12	1,2652	0,2450	-3,0008	3,5298	-2,7151	3,9788	-3,0416	3,9380
UV14	1,1244	0,2469	-2,3669	4,6804	-1,9214	5,0855	-2,4480	5,0450
English	8,3176	2,0260	-2,3335	4,6113	-1,9040	5,1591	-2,2742	4,9617
Nasa	1,2342	0,3424	-2,2611	3,1419	-2,0859	2,9083	-2,1735	2,7623

Tabla 2. Puntuaciones estándar de las distancias a los pivotes más cercano y más lejano.

Tomando la colección English como ejemplo, observamos que la distancia media entre dos palabras es $\mu = 8,31$, y la desviación típica es $\sigma = 2,02$. Hay que

tener en cuenta que, aunque la distribución de distancias es normal, las puntuaciones estándar se calculan sobre distancias de los objetos descartados a sus pivotes correspondientes. Un objeto será descartado por su pivote más cercano, cuando la puntuación estándar de su distancia a ese pivote, es de aproximadamente $-2,33$. Sin embargo, cuando esa puntuación está cercana o es inferior a $-1,9$ y la puntuación estándar de la distancia al pivote más lejano es aproximadamente $5,1$, el objeto va a ser descartado únicamente por el más pivote más lejano. En las dos últimas columnas mostramos las puntuaciones estándar de las distancias a los pivotes más cercano y más lejano, para los objetos que pueden ser descartados por ambos.

Resulta curioso observar que, a medida que la complejidad de las colecciones sintéticas crece, hay pocas posibilidades de descartar el objeto mediante su pivote más lejano. Estos resultados son acordes con los obtenidos en la figura 1.

Estas puntuaciones nos dan un umbral para cada colección, que nos permite decidir, para cada objeto, si será el pivote más cercano o el más lejano, el que tenga más posibilidades de descartarlo.

Como conclusión, el pivote más cercano es el que tiene mayor capacidad para descartar un objeto, y solo en aquellos casos en los que no está suficientemente cerca, vale la pena tratar de utilizar el pivote más lejano, si realmente está lo suficientemente lejos. Las puntuaciones estándar de la distancia al pivote más cercano y al pivote más lejano, pueden darnos para cada colección, una guía de cuando utilizar el pivote más lejano. En caso de duda, porque la distancia de ambos podría estar por encima o por debajo de estos umbrales, la opción más fiable es usar el pivote más cercano.

5 Índice de Espacio Mínimo Basado en Pivotes

Basándose en los resultados de nuestro análisis empírico acerca de la eficacia de los pivotes, proponemos construir un índice en el que, para cada objeto almacenamos el identificador del pivote que se va a utilizar y la distancia del objeto a ese pivote. Dicho pivote será el más cercano, o el más lejano en aquellos casos en los que la distancia del objeto al más cercano no es lo suficientemente pequeña, y a la vez existe un pivote suficientemente lejano.

Podríamos ver el índice resultante como si se crease una partición de Voronoi sobre el espacio, almacenando para cada objeto, la partición a la que pertenece y la distancia al pivote central de la partición. Sin embargo, en nuestro caso, para los objetos que no están lo suficientemente cerca del centro de su partición (porque están distantes del resto de elementos de la colección o porque están muy cerca del límite de dos regiones), almacenamos información sobre su pivote más lejano, en lugar de sobre el más cercano.

5.1 Almacenamiento de Distancias con Menor Precisión

Ya que nuestro objetivo es reducir el espacio del índice, podemos almacenar los identificadores de los pivotes con $\log_2(m)$ bits, siendo m el número de pivotes,

por lo que necesitarán menos de un byte en la mayoría de los casos. La distancia al pivote puede almacenarse con un número variable de bits, si permitimos perder alguna precisión.

Para evaluar nuestro método, nos referiremos a él como Índice de Pivote Único (UPI). La tabla 3 muestra dos configuraciones diferentes del método. En la primera ("UPI (4 bytes)"), los identificadores de los pivotes se almacenaron utilizando el número mínimo de bits, y para las distancias se usaron los 4 bytes necesarios para precisión flotante. En el segundo ("UPI (2 bytes)"), para los identificadores de los pivotes se utilizó también el número mínimo de bits, pero las distancias se almacenaron utilizando 2 bytes, es decir, con la mitad de precisión. Para cada opción se muestra el espacio utilizado por el índice en MB ("Espac.") y el número de evaluaciones de la función de distancia ("Eval. d"). Podemos ver en la tabla que cuando las distancias se almacenan con pérdida de precisión, la desventaja en el número de comparaciones es casi inexistente, mientras que el espacio se reduce aproximadamente a la mitad del necesario cuando se usa precisión flotante para las distancias.

Colec.	UPI (4 bytes)		UPI (2 bytes)	
	Espac.	Eval. d	Espac.	Eval. d
UV08	0,4311	3094,13	0,2594	3095,72
UV10	0,4348	5891,64	0,2631	5893,84
UV12	0,4456	9739,49	0,2740	9741,91
UV14	0,4673	18160,91	0,2957	18164,03
English	0,3083	12373,45	0,1897	12373,45
Nasa	0,1757	1958,34	0,1068	1958,72

Tabla 3. Espacio y evaluaciones de la distancia cuando se utiliza menor precisión.

El número de bits necesarios para almacenar las distancias, depende del rango de valores obtenido por la función de distancia. Por lo tanto, nuestro método puede ser parametrizado para guardar las distancias con un determinado número de bits, en función de la definición de la función de la distancia y de la tolerancia a la pérdida de precisión.

Colec.	SSS (α óptimo)		KVP (k=2)		UPI		Lista de Clusters	
	Espac.	Eval. d	Espac.	Eval. d	Espac.	Eval. d	Espac.	Eval. d
UV08	18,1963	141,43	1,3752	8724,62	0,2594	3095,72	0,3643	6139,21
UV10	60,4255	367,14	1,3789	13063,63	0,2631	5893,84	0,3710	11264,98
UV12	85,8317	645,08	1,3897	18955,94	0,2740	9741,91	0,3710	17273,55
UV14	168,5734	1381,64	1,4115	28502,26	0,2957	18164,03	0,3710	28253,04
English	45,0553	354,89	0,9604	18305,43	0,1897	8872,37	0,2651	7885,79
Nasa	7,4438	168,62	0,5547	2676,93	0,1068	1958,72	0,1427	2027,08

Tabla 4. Comparación con otros métodos.

5.2 Evaluación

Para comprobar la competitividad del método propuesto, lo comparamos con los siguientes métodos:

- SSS con el espacio necesario para su configuración óptima ("SSS (α óptimo)").
- KVP almacenando para objeto solo la distancia a sus pivotes más cercano y más lejano ($k = 2$) ("KVP ($k = 2$)").
- Lista de Clusters [15], puesto que se trata de un método representativo de la efectividad de los métodos basados en clustering ("Lista de Clusters").

Aunque SSS realiza un número mucho menor de evaluaciones, lo comparamos con nuestro método porque constituye una buena base de referencia para el número de evaluaciones de la distancia y para la utilización de espacio.

La tabla 4 muestra los resultados obtenidos. Nuestro método obtiene mejores resultados que KVP en todas las colecciones utilizando menos espacio para almacenar el índice. Frente a Lista de Clusters, nuestro método obtiene mejores resultados en términos de evaluaciones de la función de distancia, en todas las colecciones de vectores sintéticos y en Nasa, usando además menos espacio para el índice. En el caso de English, nuestro método necesita más evaluaciones de la función de distancia, pero requiere menos espacio para almacenar el índice. El número de evaluaciones de la función de la distancia es, por supuesto, superior a los obtenidos con SSS, pero la reducción del espacio necesario para el índice es muy importante. Por ejemplo, en el caso de UV14, SSS crea un índice de cerca de 168 MB, mientras que nuestro método solo necesita $0,37$ MB para ello. Estos resultados demuestran que nuestro método es una estrategia competitiva con las propuestas anteriores, cuando se usa la misma cantidad de espacio.

6 Conclusiones

En este trabajo, consideramos el problema de la reducción de los requerimientos de espacio en los métodos basados en pivotes, para la búsqueda por similitud en espacios métricos.

Presentamos un análisis empírico de la eficacia de los pivotes para cada objeto de la base de datos, estudiando el efecto de la elección del conjunto inicial de pivotes y la selección de los más prometedores para cada objeto.

Nuestros resultados muestran también que podemos almacenar para cada objeto, únicamente la distancia a su pivote más prometedor, siendo este el más cercano o el más lejano, y reducir la cantidad de espacio necesaria para el índice, a la que requieren los métodos basados en clustering.

Basándose en los resultados de este análisis, proponemos un nuevo método basado en pivotes que necesita una cantidad de espacio menor o muy cercana a la necesaria para los métodos basados en clustering. Nuestro método combina las estrategias de reducción de rango y de ámbito: almacenamos solo la distancia más prometedora para cada objeto, y para ello utilizamos la mitad de la precisión.

Nuestros resultados muestran que el método supone una estrategia competitiva frente a propuestas anteriores, cuando usamos la misma cantidad de espacio.

Como trabajo futuro, estamos estudiando la parametrización del número de bits usados para almacenar la distancia.

Referencias

1. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Computing Surveys **33**(3) (September 2001) 273–321 ACM Press.
2. Vidal, E.: An algorithm for finding nearest neighbors in (approximately) constant average time. Pattern Recognition Letters **4** (1986) 145–157 Elsevier.
3. Micó, L., Oncina, J., Vidal, R.E.: A new version of the nearest-neighbor approximating and eliminating search (aesa) with linear pre-processing time and memory requirements. Pattern Recognition Letters **15** (1994) 9–17 Elsevier.
4. Baeza-Yates, R., Cunto, W., Manber, U., Wu, S.: Proximity matching using fixed-queries trees. In: Proc. of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM'94), Asilomar, C.A., Springer-Verlag (1994) 198–212
5. Chávez, E., Marroquín, J.L., Navarro, G.: Fixed queries array: A fast and economical data structure for proximity searching. Multimedia Tools and Applications **14**(2) (2001) 113–135
6. Yianilos, P.: Data structures and algorithms for nearest-neighbor search in general metric spaces. In: Proc. of the fourth annual ACM-SIAM Symposium on Discrete Algorithms (SODA'93), ACM Press (1993) 311–321
7. Bustos, B., Navarro, G., Chávez, E.: Pivot selection techniques for proximity searching in metric spaces. Pattern Recognition Letters **24**(14) (2003) 2357–2366 Elsevier.
8. Figueroa, K., Fredriksson, K.: Simple space-time trade-offs for aesa. In: Proc. 6th Workshop on Efficient and Experimental Algorithms (WEA 2007). LNCS 4525, Springer-Verlag (2007) 229–241
9. Brisaboa, N.R., Fariña, A., Pedreira, O., Reyes, N.: Similarity search using sparse pivots for efficient multimedia information retrieval. In: Proc. of the 8th IEEE International Symposium on Multimedia (ISM'06), San Diego, California, USA, IEEE Press (2006) 881–888
10. Celik, C.: Priority vantage points structures for similarity queries in metric spaces. In: Proc. of EurAsia-ICT 2002: Information and Communication Technology. Volume 2510 of Lecture Notes in Computer Science., Springer (2002)
11. Celik, C.: Effective use of space for pivot-based metric indexing structures. In: Proc. of Int. Workshop on Similarity Search and Applications (SISAP 2008), Cancún, México, IEEE Press (2008) 402–409
12. Vleugels, J., Veltkamp, R.C.: Efficient image retrieval through vantage objects. Pattern Recognition **35**(1) (2002) 69–80 Elsevier.
13. van Leuken, R.H., Veltkamp, R.C., Typke, R.: Selecting vantage objects for similarity indexing. In: Proc. of the 18th International Conference on Pattern Recognition (ICPR 2006), IEEE Press (2006) 453–456
14. SISAP: Metric spaces library (`http://sisap.org/metric_space_library.html`)
15. Chávez, E., Navarro, G.: A compact space decomposition for effective metric indexing. Pattern Recognition Letters **26**(9) (2005) 1363–1376