Nuevos Índices para la Búsqueda por Proximidad en Espacios Métricos. [*]

Karina Figueroa Gonzalo Navarro [**] Edgar Chávez [***]

1. Motivación

La búsqueda por proximidad consiste en recuperar de una base de datos (*BD*) los elementos más semejantes a uno de consulta dado. Éste tipo de búsquedas son indispensables en aplicaciones que requieren nuevos formatos de datos, conocidas como *BD multimediales (BDM)*, (voz, imágenes, ADN, etc.), pues son inútiles las búsquedas por igualdad en este tipo de objetos. Existen muchas aplicaciones que utilizan el concepto de búsqueda por proximidad. Entre ellas: reconocimiento de patrones, recuperación de información y biología computacional.

En *BD* con miles o millones de objetos y/o una comparación costosa entre ellos (en tiempo y/o recursos), hacer una búsqueda secuencial resulta impensable. Una alternativa para localizar rápidamente los elementos *relevantes* a la consulta, y reducir el número de comparaciones necesarias entre objetos para responder una consulta es usar un índice.

Muchos de los tipos de objetos de *BDM* son modelados como vectores en múltiples dimensiones[4] y después tratados con *índices para espacios vectoriales*. Éstos usan las coordenadas de cada vector pero se degradan rápidamente cuando la dimensión de los datos aumenta, conocido como *la maldición de la dimensión*[2]. En un intento por evitar esa maldición se han propuesto otro tipo de índices, como los basados en las distancias entre objetos: *índices para espacios métricos*. Éstos, además, pueden tratar objetos que no pueden ser modelados como vectores.

2. Conceptos Básicos

Formalmente, sea $(\mathbb{X}, d)$ un espacio métrico, donde $\mathbb{X}$ es el universo de objetos válidos, y d es una función de distancia, $d : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}^+$ que denotará la medida de semejanza entre los elementos de $\mathbb{X}$. Cuanto más pequeo es el valor de d, más parecidos entre sí son los elementos. La función de distancia debe satisfacer las siguientes propiedades: positividad estricta, $d(x, y) > 0$ si $x \neq y$; simetría, $d(x, y) = d(y, x)$; reflexividad, $d(x, x) = 0$; desigualdad triangular, $d(x, z) \leq d(x, y) + d(y, z)$. Nuestra *BD* será $\mathbb{U} \subseteq \mathbb{X}$.

Las consultas típicas pretenden encontrar al elemento más similar (vecino más cercano), o bien, dado un radio de interés (r) se quieren encontrar los objetos a distancia menor igual que r.

3. Trabajos previos en el área

Las búsquedas por proximidad en espacios métricos usualmente son resueltas en dos partes: preprocesamiento y consulta. Durante el preprocesamiento se construye el índice que se ocupará durante la consulta para reducir comparaciones de distancias. Básicamente el estado del arte de este tipo de índices está dividido en dos familias: las basadas en pivotes y las basadas en particiones compactas [2].

Algoritmos basados en pivotes (*ABPI*). En esta familia su índice consiste en elegir un conjunto (*pivotes*) $\mathbb{P} = \{p_1, p_2, \ldots, p_k\} \subseteq \mathbb{U}$, $k = |\mathbb{P}|$, calcular y almacenar (con alguna estructura, árboles usualmente) $\{d(p_1, u), d(p_2, u), \ldots, d(p_k, u)\}, \forall u \in \mathbb{U}$.

Algoritmos basados en particiones compactas (*ABPC*). Esta familia divide el espacio en zonas tan compactas como sea posible, seleccionando un conjunto de objetos (*centros*) $p_1, \ldots, p_k \subseteq \mathbb{U}$ y distribuyendo los demás elementos entre las k zonas pertenecientes a cada p_i. El índice está compuesto por los centros, los elementos que pertenecen a cada zona y, en algunos casos, información adicional sobre las distancias. Un criterio de distribución, entre varios, es $\forall\, u \in \mathbb{U} \in$ a la zona de p_i, si p_i es su centro más cercano. El proceso es recursivo en cada zona hasta que no sea posible o deseable dividir el espacio.

En ambas familias, dada una consulta q, de acuerdo al índice usado, q se compara contra los pivotes o los centros (según sea el caso). Con ayuda de la desigualdad triangular y el índice, es posible acotar inferiormente la distancia entre q y cualquier $u \in \mathbb{U}$. De esta forma, es posible reducir cálculos de distancia para responder la consulta.

[*]Núcleo Milenio, Centro de investigación de la Web. Proyecto P04-067-F, Mideplan, Chile (primer y segundo autor).

[**]Departamento de Computación. Universidad de Chile, Chile (primero y segundo autor). {kfiguero,gnavarro}@dcc.uchile.cl

[***]Facultad de Físico Matemáticas. Universidad Michoacana, México. elchavez@fismat.umich.mx

4. Metodología

La propuesta en este trabajo (algoritmo basado en permutaciones *ABPE*) plantea un enfoque distinto al estado del arte para enfrentar el problema de búsqueda por proximidad en espacios métricos, lo que permite tener nuevas alternativas y nuevos horizontes para este problema. En este nuevo enfoque cada elemento ordena el espacio de acuerdo a cómo lo *percibe* (por cercanía en orden ascendente) [1].

En *ABPE*, inicialmente, se selecciona un conjunto (*permutantes*) $\mathbb{P} = \{p_1, \ldots, p_k\} \subseteq \mathbb{U}, |\mathbb{P}| = k$. En el indexamiento, cada elemento $u \in \mathbb{U}$ define un preorden $\leq_u$ en los elementos en $\mathbb{P}$ dado por la distancia a u, como: $p_1, p_2 \in \mathbb{P}$: $p_1 \leq_u p_2 \Leftrightarrow d(u, p_1) \leq d(u, p_2)$. Asociamos[1] $\leq_u$ a una *permutación* Π_u de $p_1, p_2, \ldots, p_k$, donde $p_i \leq_u p_{i+1}$. Los elementos a la misma distancia toman un orden arbitrario pero consistente. Las permutaciones de la base de datos conformarán nuestro índice. Denotaremos $\Pi^{-1}(p)$ a la posición del permutante p en Π.

Para una consulta q, se genera su permutación. De la diferencia entre las permutaciones Π_q y Π_u, $\forall\, u \in \mathbb{U}$, se puede inferir si u es relevante para q, y descartarlo sin haber calculado $d(q, u)$ (algoritmo exacto), o sólo inferir semejanza entre u y q (algoritmo probabilistico).

El algoritmo exacto de *ABPE*, consiste en encontrar *inversiones* entre dos elementos en dos permutaciones que cumplan ciertas condiciones. Por ejemplo: si $r \geq 0$ es un radio de interés, entre $\Pi_q = p_1, p_2$ y $\Pi_u = p_2, p_1$, ocurrió una inversión, siempre y cuando $d(p_2, q) - d(p_1, q) > r$ y $d(p_1, u) - d(p_2, u) > r$. En este caso u puede ser descartado sin compararse por distancia contra la q.

El algoritmo probabilístico es imbatible en la práctica. Éste requiere medir sólo la similaridad entre permutaciones, para esto se usó Spearman Rho (aunque existen otras), definida como:

$$S_\rho(\Pi_q, \Pi_u) = \left(\sum_{i=1}^{k} |\Pi_u^{-1}(p_i) - \Pi_q^{-1}(p_i)|^2 \right)^{\frac{1}{2}}$$

En la figura 1 se muestra un ejemplo de las dos fases de *ABPE* (probabilístico). En la fase de preprocesamiento se calculan las permutaciones para los elementos de la *BD*; para q, en la fase de consulta. Para saber en qué orden comparar los elementos de la *BD*, éstos son ordenados por la similaridad entre permutaciones (S_ρ). En el ejemplo q se comparará primero contra $u_{13}, u_{14}, \ldots$.

5. Resultados preliminares

Para la mayoría de aplicaciones basadas en búsquedas por proximidad, recuperar el 90 % de las respuestas es un resultado aceptable. La versión probabilística de *ABPE* ha

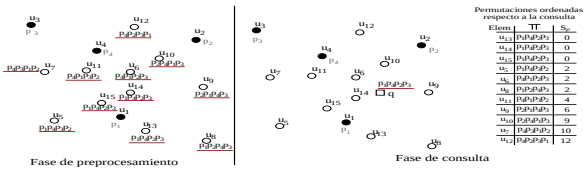

Figura 1. Ejemplo de las dos fases de un algoritmo basados en permutaciones.

sido comparada contra algoritmos probabilísticos de ambas familias (*ABPI*, *ABPC*). Algunos resultados son:

Cubo unitario[2]. En el caso de *ABPE* recupera el 90 % de la respuesta examinando menos del 10 % de la *BD*, mientras que las técnicas existentes requieren comparar al menos el 60 % de los datos.

Imágenes Faciales[3]. En esta colección, usando métodos exactos se requiere comparar al menos el 80 % de la *BD* para recuperar el vecino más cercano. En el caso *ABPE* (probabilístico) se requiere revisar menos del 10 %.

Combinando *ABPE* y *ABPI*. Con esta combinación ha sido posible mejorar el desempeo (reducir el número de cálculos de distancia) de algunos algoritmos de *ABPI* [3].

Actualmente trabajamos en: aplicar *ABPE* en *BD* con similaridades en lugar de distancias, desarrollar heurísticas para reducir el tiempo de CPU empleado y otros. La investigación se encuentra en un 95 % de avance.

Referencias

[1] E. Chávez, K. Figueroa, and G. Navarro. Proximity searching in high dimensional spaces with a proximity preserving order. In *MICAI 2005*, volume 3789 of *LNCS*, pages 405–414, 2005.

[2] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[3] K. Figueroa, E. Chávez, G. Navarro, and R. Paredes. On the lower cost for proximity searching in metric spaces. *5th International Workshop on Experimental Algorithms*, LNCS 4007:270–290, May 2006.

[4] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[5] P. Phillips, H. Wechsler, J. Huang, and P. Rauss. The FERET database and evaluation procedure for face recognition algorithms. *Image and Vision Computing Journal*, 16(5):295–306, 1998.

[1] Aprovechando que este preorden no cumple con la antisimetría, pero éste induce un orden total en el conjunto cociente.

[2] Vectores distribuidos uniformemente en el cubo unitario

[3] Colección de fotografías con rostros humanos FERET [5]