FQTrie Desbalanceado

Carina Ruano, Ana Villegas, Norma Herrera
Departamento de Informática
Universidad Nacional de San Luis
San Luis, Argentina
{cmruano,anaville, nherrera}@unsl.edu.ar

Edgar Chávez
Escuela de Ciencias Físico-Matemáticas
Universidad Michoacana
Morelia - México
elchavez@fismat.umich.mx

Resumen

El modelo de Espacios Métricos permite formalizar el concepto de búsqueda por similitud en bases de datos no tradicionales. El objetivo es construir *índices* que permitan reducir el tiempo necesario para resolver una búsqueda por similitud. Uno de los enfoques para la construcción de índices es el usado por los algoritmos basados en pivotes. Sobre bases de datos tradicionales se sabe que, mientras más balanceado sea un índice, mejor será su desempeño durante una búsqueda. Los supuestos que llevan a esta conclusión no son ciertos en espacios métricos y por lo tanto un enfoque desbalanceado suele ser la mejor opción para los espacios de alta dimensión. En este trabajo nos proponemos estudiar técnicas para lograr desbalancear el Fixed Queries Trie.

Palabras claves: Bases de Datos, Espacios Métricos, Índices, Pivotes.

1. Introducción

El concepto de *búsquedas por similitud* o *por proximidad*, es decir buscar elementos de una base de datos que sean similares o cercanos a uno dado, aparece en diversas áreas de computación, tales como reconocimiento de voz, reconocimiento de imágenes, compresión de texto, biología computacional, inteligencia artificial, minería de datos, entre otras.

En [4] se muestra que el problema se puede expresar como sigue: dado un conjunto de objetos $\mathcal{X}$ y una función de distancia d definida entre ellos que mide cuán diferentes son, el objetivo es recuperar todos aquellos elementos que sean similares a uno dado. Esta función d cumple con las propiedades características de una función de distancia:

Positividad Estricta: $d(x, y) = 0 \Leftrightarrow x = y$.

Simetría: $d(x, y) = d(y, x)$.

Desigualdad Triangular: $d(x, z) \leq d(x, y) + d(y, z)$.

El par $(\mathcal{X}, d)$ se denomina *espacio métrico*. La base de datos será un subconjunto finito $\mathcal{U} \subseteq \mathcal{X}$ de cardinalidad n.

En este nuevo modelo de bases de datos, una de las consultas típicas que implica recuperar objetos similares es la *búsqueda por rango*, que denotaremos con $(q, r)_d$. Dado un elemento $q \in \mathcal{X}$, al que llamaremos *query* y un radio de tolerancia r, una búsqueda por rango consiste en recuperar los objetos de la base de datos cuya distancia a q no sea mayor que r, es decir:

$$(q, r)_d = \{u \in \mathcal{U} : d(q, u) \leq r\}$$

Se han logrado avances importantes sobre búsquedas por similitud en espacios métricos en torno al concepto de construir un *índice*, es decir, una estructura de datos que permita reducir el tiempo necesario para resolver una consulta. En este tiempo influyen tres factores, a saber: la cantidad de evaluaciones de la función de distancia d hechas, la cantidad de operaciones adicionales realizadas que implican un tiempo extra de CPU y la cantidad de accesos a páginas de disco que implican un tiempo extra de I/O. En consecuencia, el tiempo total de resolución de una búsqueda puede ser calculado como:

$$T = \#d \times \text{complejidad}(d) + \text{Tpo } CPU + \text{Tpo } I/O$$

En muchas aplicaciones la evaluación de la función d es tan costosa que las demás componentes pueden ser despreciadas. Éste es el modelo de complejidad usado en la mayoría de los trabajos de investigación hechos en esta temática. Sin embargo, hay que prestar especial atención al tiempo extra de *CPU*, dado que reducir este tiempo produce que en la práctica la búsqueda sea más rápida, aún cuando estemos realizando la misma cantidad de evaluaciones de la función d. De igual manera, el tiempo de *I/O* puede jugar un papel importante en algunas aplicaciones.

En [4] se presenta un desarrollo unificador de las soluciones existentes en la temática. En dicho trabajo se muestra que todos los enfoques para la construcción de índices en espacios métricos consisten en:

- particionar el espacio en clases de equivalencia.
- indexar las clases de equivalencia.

- durante la búsqueda, usando el índice y la desigualdad triangular, descartar algunas clases y buscar exhaustivamente en las restantes.

La diferencia entre los distintos algoritmos radica en cómo construyen estas clases de equivalencia. Básicamente se pueden distinguir dos enfoques: *algoritmos basados en pivotes* y *algoritmos basados en particiones compactas*.

En este trabajo nos centraremos en los algoritmos basados en pivotes, más específicamente el Fixed Queries Trie (FQTrie). Comenzamos dando una introducción a este índice; en las secciones siguientes describimos el concepto de dimensión y analizamos el efecto de desbalancear el índice sobre espacios de alta y de baja dimensión. Luego, proponemos una técnica para lograr un desbalance en el FQTrie. Finalizamos dando el trabajo futuro.

2. Fixed Queries Trie

Esta estructura fue presentada en [2] y forma parte de los familia de estructuras basadas en pivotes. La idea subyacente de los algoritmos basados en pivotes es la siguiente. Se seleccionan k pivotes $\{p_1, p_2, \ldots, p_k\}$, y se le asigna a cada elemento a de la base de datos, el vector o firma $\Phi(a) = (d(a, p_1), d(a, p_2), \ldots, d(a, p_k))$. Ante una búsqueda $(q, r)_d$, se computa $\Phi(q) = (d(q, p_1), d(q, p_2), \ldots, d(q, p_k))$. Luego, se descartan todos aquellos elementos a, tales que para algún pivote p_i, $\mid d(q, p_i) - d(a, p_i) \mid > r$, es decir $max_{1 \leq i \leq k} \mid d(q, p_i) - d(a, p_i) \mid = L_\infty(\Phi(a), \Phi(q)) > r$. Los elementos no descartados por la condición anterior, se comparan directamente con la query q.

Esto significa que, todos los algoritmos basados en pivotes, proyectan el espacio métrico original en un espacio vectorial k dimensional con la función de distancia L_∞. La diferencia entre todos ellos, radica en cómo implementan la búsqueda en el espacio mapeado.

La familia de estructuras *FQ (FQT, FHQT, FQA, FQtrie)* [1, 3] forman parte de las estructuras basadas en pivotes; cada una de ellas fue presentada como una mejora de la anterior

En el FQtrie [2], se utiliza un *Árbol Digital o Trie* [5] para indexar las firmas de los elementos de la base de datos.

Un *Trie* es un árbol m-ario para búsqueda lexicográfica. En esta estructura, cada elemento se considera como una secuencia de caracteres sobre un alfabeto Σ; la cardinalidad del alfabeto determina la aridad del árbol, es decir $m = |\Sigma|$. Un nodo en un trie o es un nodo externo y contiene un elemento; o es un nodo interno y contiene m punteros a subtries. Dada una cadena, se usan los caracteres que la conforman para direccionar la búsqueda en el árbol. Estando en un nodo de nivel i, la selección del subtrie que le corresponde se realiza en función del i-ésimo caracter de la cadena. El nodo raíz usa el primer caracter, los nodos hijos de la raíz usan el segundo caracter, y así sucesivamente.

En un trie m-ario la búsqueda toma un tiempo que es proporcional a la longitud de la cadena, independientemente de cual sea el tamaño de la base de datos.

3. La maldición de la dimensión

Uno de los principales obstáculos en el diseño de buenas técnicas de indización es lo que se conoce con el nombre de *maldición de la dimensionalidad*. El concepto de dimensionalidad está relacionado a la dificultad o facilidad de buscar en un determinado espacio métrico. La dimensión intrínseca de un espacio métrico se define en [4] como $\rho = \frac{\mu^2}{2\sigma^2}$, siendo μ y σ^2 la media y la varianza respectivamente de su histograma de distancias. Es decir que, a medida que la dimensionalidad intrínseca crece, la media crece y su varianza se reduce. Esto significa que el histograma de distancia se concentra más alrededor de su media, lo que influye negativamente en los algoritmos de indización.

La figura 1 da una idea intuitiva de por qué el problema de búsqueda se torna más difícil cuando el histograma es más concentrado. Consideremos una búsqueda $(q, r)_d$, y un índice basado en pivotes elegidos aleatoriamente. Los histogramas de la figura representan posibles distribuciones de distancias entre q y algún pivote p.

La regla de eliminación dice que podemos descartar aquellos puntos y tales que $y \notin [d(p, q) - r, d(p, q) + r]$. Las áreas sombreadas muestran los puntos que no podrán descartarse. A medida que el histograma se concentra más alrededor de su media, disminuye la cantidad de puntos que pueden descartarse usando como dato $d(p, q)$. Este fenómeno es independiente de la naturaleza del espacio métrico, y nos brinda una forma de cuantificar cuán dura es una búsqueda sobre el mismo.

4. Desbalanceando el Índice

Sobre bases de datos tradicionales se sabe que mientras más balanceado sea un índice mejor será su desempeño durante una búsqueda. Estructuras tales como AVL, Skip List y Árbol B han sido diseñadas persiguiendo este objetivo. Esta idea de balancear una estructura de datos se basa en dos supuestos:

- el tipo de búsqueda que se quiere resolver es la búsqueda exacta.
- existe un orden lineal en el conjunto sobre el que se realizarán las búsquedas.

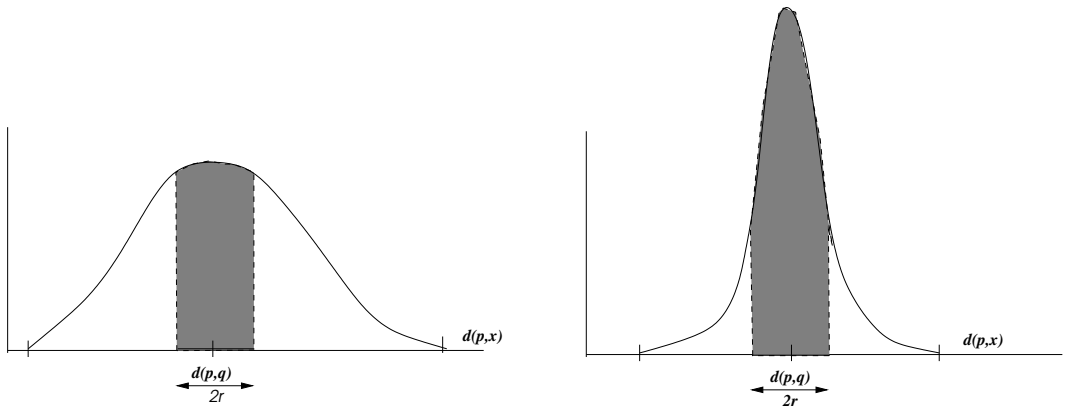

Figura 1: *Histogramas de distancias de baja dimensionalidad (izquierda), y de alta dimensionalidad (derecha)*

Ninguno de estos dos supuestos es válido sobre espacios métricos. La única información con la que se cuenta al momento de organizar un índice (generalmente un árbol) en espacios métricos es la distancia entre elementos. Se impone un orden lineal a los elementos ordenándolos de acuerdo a su distancia a la raíz del árbol. Sin embargo, la mayoría de los autores tratan de balancear estos árboles dividiendo los rangos de distancia en partes uniformes, con el fin de lograr que cada subárbol tenga aproximadamente la misma cantidad de elementos.

Los problemas con este enfoque surgen cuando consideramos el tipo de búsqueda que realizaremos y la distribución de los elementos en el espacio métrico.

En primer lugar, la búsqueda no es exacta sino que existe un radio de tolerancia r que se conoce en el momento de realizar la búsqueda y no cuando se construye el índice. En segundo lugar, la distribución de los elementos en el espacio no se corresponde necesariamente con una distribución uniforme. Como vimos en la sección anterior, los espacios métricos de baja dimensionalidad presentan un histograma más uniforme que los espacios de alta dimensionalidad.

Como el histograma de espacios de baja dimensionalidad no es concentrado, una partición donde cada subárbol tenga la misma cantidad de elementos produce franjas de aproximadamente el mismo tamaño y, en consecuencia, la búsqueda entra en un número razonable de subárboles.

Pero si el espacio es de alta dimensionalidad, el histograma está concentrado en un pequeño rango de valores, a donde probablemente también pertenecerá la query q. Esto produce que una proporción grande de elementos no puedan descartarse cuando se compara la query q con la raíz del árbol. Si balanceamos el árbol hacemos aún más ineficiente la búsqueda. Como el histograma es más concentrado, particionar en partes con la misma cantidad de elementos produce que las zonas sean cada vez más pequeñas; pero como el radio de búsqueda es el mismo, una mayor cantidad de zonas serán intersectadas por la query, lo que producirá que una mayor cantidad de subárboles deban ser revisados. Por esta razón balancear los árboles en espacios de alta dimensionalidad no es una buena opción.

Si armamos un árbol particionando de manera tal que cada parte tenga el mismo ancho evitamos este problema y obtenemos algunos beneficios adicionales. Dado que el ancho es independiente de la dimensión, no necesitamos revisar más subárboles cuando la dimensión crece. Además, los subárboles que contienen aquellas partes que conforman el núcleo del espacio tendrá mayor cantidad de elementos logrando así cada vez un mayor desbalance.

5. Desbalanceando el FQTrie

Como todos los elementos del espacio tiene el mismo tamaño de firma, el trie utilizado por el FQTrie queda necesariamente balanceado. Este trie puede verse como un índice sobre U* (el conjunto de firmas). Lo que nosotros en realidad debemos desbalancear es el índice sobre U (el espacio métrico).

El índice base sobre el que se construye el FQTrie es el FQT. *Fixed Height Fixed Queries Tree (FHQT)*[1]. Si miramos el FHQT subyacente, desbalancearlo significa conseguir pivotes que distribuyan los elementos en forma no uniforme dentro del índice.

Para lograr esto formaremos la firma de los elementos de manera tal que no todos los pivotes contribuyan a la firma de todos los elementos. El procesa a utilizar es el siguiente.

- Se selecciona un pivote p_1 como raíz.
- Los M elementos más cercanos a p_1 forman el primer grupo de elementos.
- Con los restantes se procede recursivamente: se elije un pivote p_2 y los M elementos más cercanos a p_2 forman el segundo grupo de elementos.

Este proceso continúa hasta que no queden más elementos para clasificar. Como resultado final tendre-

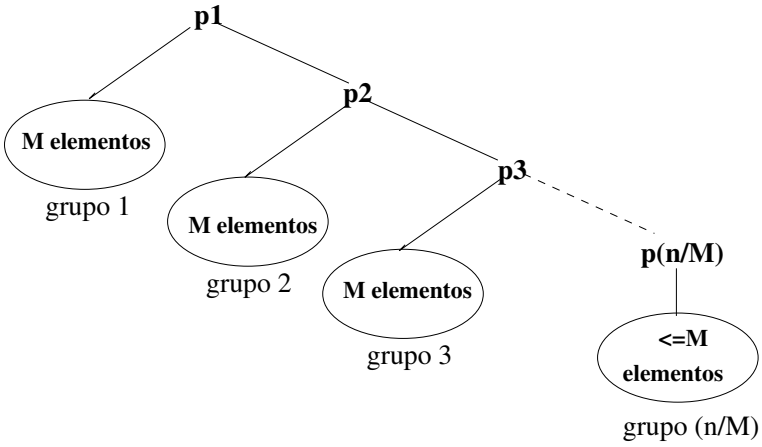

Figura 2: *Proceso de creación de grupos*

mos los n elementos del espacio clasificados en n/M grupos (ver figura 2).

La firma de cada elemento x se forma ahora teniendo en cuenta el grupo al que pertenece x. Si se utilizan k pivotes para la creación de cada firma, entonces:

- los elementos del primer grupo forman su firma con los pivotes $p_2, \cdots, p_{k+1}$;
- los elementos del segundo grupo forman su firma con los pivotes $p_3, \cdots, p_{k+2}$;
- en general los elementos del grupo i forman su firma con los pivotes $p_{i+1}, \cdots, p_{k+i}$.

Cabe señalar que se necesitarán elegir k pivotes adicionales a los ya utilizados a fin de poder completar la firma de los $(k-1)$ últimos grupos.

El proceso de búsqueda debe ahora tener en cuenta esta situación a fin de descartar correctamente los grupos que correspondan. En el peor caso la query q deberá compararse con $k + n/M$ pivotes.

6. Trabajo Futuro

Nos proponemos estudiar cuál es el efecto de la técnica de desbalance propuesta sobre la performance de las búsquedas. El trabajo a realizar puede resumirse en los siguientes puntos:

- Establecer empíricamente el valor más adecuado para M.
- Estudiar el efecto de esta técnica de desbalance experimentando sobre espacios de alta y también de baja dimensionalidad.
- Comparar los resultados obtenidos con el FQTrie original.
- Analizar otras técnica posibles para el desbalance del índice.

Referencias

[1] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixed-queries trees. In *Proc. 5th Combinatorial Pattern Matching (CPM'94)*, LNCS 807, pages 198–212, 1994.

[2] E. Chávez and K. Figueroa. Faster proximity searching in metric data. In *Proceedings of MICAI 2004. LNCS 2972, Springer*, Cd. de México, México, 2004.

[3] E. Chávez, J. Marroquín, and G. Navarro. Fixed queries array: A fast and economical data structure for proximity searching. *Multimedia Tools and Applications (MTAP)*, 14(2):113–135, 2001.

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

[5] K. Melhorn. *Data Structures and Algorithms*, volume III - Multidimensional Searching and Computational Geometry. Springer, 1984.