Búsquedas en Espacios Métricos

Norma Herrera ; Nora Reyes - {nherrera,nreyes}@unsl.edu.ar
Departamento de Informática - Universidad Nacional de San Luis

Ricardo Baeza-Yates ; Gonzalo Navarro - {rbaeza,gnavarro}@dcc.uchile.cl
Departamento de Ciencias de la Computación - Universidad de Chile

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

1 Introducción y motivación

La búsqueda es un problema fundamental en Ciencias de la Computación, presente virtualmente en cada aplicación. Las bases de datos tradicionales se construyen basándose en el concepto de búsqueda exacta. Las consultas a la base de datos retornan todos aquellos registros cuyas claves coinciden con la aportada en la búsqueda. Las búsquedas mas sofisticadas como búsqueda de rangos sobre claves numéricas, o búsqueda de prefijos sobre claves alfabéticas, se basan en la existencia de un orden lineal sobre las claves de búsqueda.

Actualmente las bases de datos han incluido la capacidad de almacenar nuevos tipos de datos tales como imágenes, sonido, video, etc. Estructurar este tipo de datos en registros, para adecuarlos al concepto tradicional de búsqueda exacta, es sumamente dificultoso y hasta imposible en algunos casos. Aún cuando pudiera hacerse las consultas a la base serán por objetos similares a uno dado, y no por aquellos exactamente iguales. Este tipo de búsqueda se conoce con el nombre de ***búsqueda aproximada o búsqueda por similitud***, y surge en áreas tales como reconocimiento de voz, reconocimiento de imágenes, etc.

La necesidad de una respuesta rápida y adecuada, y un eficiente uso de memoria, hace necesaria la existencia de estructuras de datos especializadas que incluyan estos aspectos.

El planteo general del problema es: dado un conjunto $\mathcal{S} \subseteq \mathcal{U}$, recuperar los elementos de $\mathcal{S}$ que sean similares a uno dado, donde la similitud entre elementos es modelada mediante una función de distancia positiva d. $\mathcal{U}$ denota el universo de objetos válidos y $\mathcal{S}$, un subconjunto finito de $\mathcal{U}$, denota la base de datos en donde buscamos. El par $(\mathcal{U}, d)$ es llamado ***espacio métrico***. La función $d : \mathcal{U} \times \mathcal{U} \rightarrow R^+$ cumple con las propiedades propias de una función de distancia:

1. $d(x, y) = 0 \Leftrightarrow x = y$ (positividad estricta)
2. $d(x, y) = d(y, x)$(simetría)
3. $d(x, y) \leq d(x, y) + (d(y, z)$ (desigualdad tringular)

Básicamente, existen tres tipos de búsquedas de interés en espacios métricos:

Búsqueda por rango: recuperar todos los elementos de $\mathcal{S}$ que están a distancia r de un elemento q dado.

Búsqueda del vecino más cercano: dado q, recupera el (o los) elemento(s) más cercano(s) a q en $\mathcal{S}$.

Búsqueda de los k vecinos más cercanos: dado q, recuperar los k elementos más cercanos a q

En muchas aplicaciones, la evaluación de la función d, suele ser una operación costosa. Es por esta razón, que se usa como medida de complejidad, la cantidad de evaluaciones de distancias hechas para resolver la búsqueda.

Un caso particular de espacios métricos son los espacios vectoriales K-dimensionales, con las funciones de distancia L_p $(p = 1, 2, \cdots, \infty)$. Para estos casos existen soluciones bien conocidas. Sin embargo, éste no es el caso general.

Las investigaciones en la actualidad tienden al estudio de algoritmos en espacios métricos generales, donde existen varias técnicas conocidas para resolver el problema en un número sublineal de cálculos de distancia, con la condición del preprocesamiento de $\mathcal{S}$. En general las estructuras sobre las que trabajan dichos algoritmos suponen que los espacios son estáticos.

Nuestro objetivo es estudiar principalmente dos estructuras: ***Geometric Near-neighbor Access Tree (GNAT)*** y ***Spatial Aproximation Tree (SAT)***, a fin de optimizar tanto su construcción, como el proceso de búsqueda.

En el caso particular del SAT, nos hemos centrado en lograr a partir de él, una estructura totalmente dinámica. Y en el caso del $GNAT$, la atención ha sido puesta en buscar alternativas para su construcción, de forma tal que el árbol resultante tenga un mejor comportamiento en tiempo de búsqueda.

2 Árbol de aproximación Espacial (SAT)

Para mostrar la idea general de la aproximación espacial se utilizan las *búsquedas del vecino más cercano*.

En este modelo, dado un punto $q \in \mathcal{U}$ y estando posicionado en algún elemento $a \in \mathcal{S}$ el objetivo es moverse a otro elemento de $\mathcal{S}$ que esté más cerca "espacialmente" de q que a. Cuando no es posible realizar más este movimiento, se está posicionado en el elemento más cercano a q de S.

Estas aproximaciones son efectuadas sólo vía los "vecinos". Cada elemento $a \in S$ tiene un conjunto de vecinos $N(a)$. La estructura natural para representar esto es un grafo dirigido. Los nodos son los elementos del conjunto y los elementos vecinos son conectados por un arco. La búsqueda procede sobre tal grafo simplemente posicionándose sobre un nodo arbitrario a y considerando todos sus vecinos. Si ningún nodo está más cerca de q que a, entonces se responde a a como el vecino más cercano a q. En otro caso, se selecciona algún vecino b de a que está más cerca de q que a y se mueve a b. Para obtener el SAT se simplifica la idea general comenzando la búsqueda en un nodo fijo y realmente se obtiene un árbol. El SAT está definido recursivamente; la propiedad que cumple la raíz a (y a su vez cada uno de los siguientes nodos) es que los hijos están más cerca de la raíz que de cualquier otro punto de S. En otras palabras: $\forall x \in S, x \in N(a) \Leftrightarrow \forall y \in N(a) - \{x\}, d(x, y) > d(x, a)$. La construcción del árbol se hace de manera recursiva pudiendo construirlo siguiendo una de dos estrategias posibles: best-fit y first-fit, de acuerdo a si cada elemento b que no está en $a \cup N(a)$ se ubicará en el subárbol del vecino más cercano a él de $N(a)$ o en el primero que sea más cercano a él que a.

De la definición se observa que se necesitan de antemano todos los elementos de la base de datos para la construcción. Nuestro objetivo es estudiar si es posible que dicha estructura admita la inserción y eliminación de elementos , sin degradar los tiempos de búsqueda.

2.1 Inserciones y eliminaciones

Luego de haber analizado distintas formas de realizar inserciones de elementos en el árbol [9], hemos obtenido un nuevo método que permite realizar la construcción incremental de SAT a un costo muy inferior a la versión estática y que mantiene el costo de las búsquedas en la mayoría de los casos menor, y en otros levemente mayor que la versión original.

El método obtenido ha surgido como una combinación muy adecuada de las bondades de los dos métodos que demostraron ser mejores entre los previamente analizados (Timestamp e Inserción en la Fringe)y los supera a ambos. Esta nueva técnica construye un árbol con *aridad acotada* y guarda en cada nodo un *timestamp* que marca el tiempo en que se insertó dicho elemento en el árbol.

Al incorporar un nuevo elemento x, éste se inserta en el punto más apropiado, si es que no se supera la cota de la aridad; en otro caso se continúa la búsqueda del lugar de inserción a partir del vecino más cercano. Sea b el

punto resultante para la inserción, luego x se agrega como el vecino *más nuevo* de b y no se efectúa ningún tipo de reconstrucción.

Ahora claramente, no se fuerza a un elemento a aparecer en la vecindad del más cercano. En tiempo de búsqueda alcanzaremos al elemento insertado porque los procesos de inserción y búsqueda son similares. Las búsquedas se pueden optimizar haciendo uso del *timestamp*.

Las eliminaciones son mucho más complicadas porque los cambios a realizar en la estructura son más costosos, y no son localizados, sino que se pueden propagar al resto de la estructura.

Se han analizado distintas opciones para eliminar elementos, desde esquemas muy simples a otros más sofisticados. Finalmente, el método que resultó más adecuado reinserta en el árbol los vecinos de un nodo eliminado (y los subárboles que cuelgan de ellos). En la reinserción, se trata de reinsertar subrboles completos para disminuir el número de evaluaciones de distancia realizadas. También se utiliza aquí el *timestamp* para evitar ciertas evaluaciones de distancia ya realizadas cuando se insertó el elemento.

3 Geometric Near-neighbor Access Tree

Esta estructura fue presentada por Serge Brin, en el año 1995[2]. El objetivo es que la estructura de datos actúe como un modelo geométricamente jerárquico de los mismos. Esto se logra construyendo una jerarquía basada en *Diagramas de Voronoi*

La construcción del $GNAT$ procede de la siguiente manera: en el primer nivel se seleccionan m pivotes $c_1, c_2, \cdots, c_m$ de $\mathcal{S}$, que se almacenan en el nodo raíz. A cada pivote c_i se le asocia el conjunto $\mathcal{S}_{c_i}$ definido como: $\mathcal{S}_{c_i} = \{x \in \mathcal{S} / d(c_i, x) < d(c_j, x),\ \forall j = 1 \ldots m\}$. Con cada $\mathcal{S}_{c_i}$ se construye recursivamente un $GNAT$.

En cada nodo del $GNAT$ de almacena además una tabla, a la que llamaremos $rango$, de tamaño $O(m^2)$ que mantiene información sobre las distancias mínimas y máximas, desde el centro c_i al conjunto $\mathcal{S}_{c_j}$; esto es: $rango_{i,j} = [min_{x \in \mathcal{S}_{c_j}} d(ci, x), max_{x \in \mathcal{S}_{c_j}} d(ci, x)]$, con $i, j = 1, \cdots, m$. Esta información, junto con la desigualdad triangular, se usa en el momento de la búsqueda para descartar subárboles. En tiempo de búsqueda, el punto de query q se compara con algún pivote c_i, y se descarta cualquier otro pivote c_j tal que $d(q, c_i) \pm r$ no intersecta $rango_{i,j}$.

En la actualidad, nuestro trabajo se ha concentrado en conseguir técnicas de selección de pivotes, que tengan un buen comportamiento cuando se las aplique a un diccionario de palabras como espacio métrico .

3.1 Detección de cluster

Descubrir la estructura subyacente del conjunto de datos es sumamente útil en el diseño de algoritmos de indexación. En nuestro caso particular, saber cómo se agrupan los elementos del espacio métrico nos servirá para identificar puntos, que al ser elegidos como pivotes, producirán una partición razonable del espacio.

Los algoritmos para detección de clusters en espacios métricos, usulamente son de complejidad cuadrática, dado que tiene que considerar todas las posibles relaciones entre pares de objetos de la muestra. En [4] se presenta un algoritmo de análisis de cluster, que tiene una complejidad subcuadrática, medida en términos de cantidad de evaluaciones de la función de distancia d. Para lograr esta complejidad, el método presentado se basa en la existencia de un índice sobre el espacio métrico.

La idea central es usar búsquedas por rango para la detección de cluster. Para ello se define el grafo de rango r (*Range r Graph*). En este grafo, se utilizan los elementos del espacio métrico como nodos, y se coneta el nodo s_i con el nodo s_j, si $d(s_i, s_j) \leq r$. Luego, cada componente conectada de este grafo forma un $r - cluster$.

Un aspecto importante, es que los puntos que están alejados simultáneamente de todos los demás (***outliers***) son aislados en este proceso (clusters de cardinalidad 1). Los *outliers* generalmente degradan el comportamiento de las estructuras; por consiguiente, es importante poder detectarlos para tratarlos en forma separada.

Nuestro objetivo es utilizar este algoritmo como paso previo a la creación del $GNAT$. La idea es, primero, aplicar este proceso y obtener los r-cluster del espacio métrico. Luego, aislar e indexar separadamente los puntos *outliers*. Finalmente, realizar la selección de pivotes entre los restantes cluster. La cantidad y la forma de selección

de pivotes dependerá de la cardinalidad de cada cluster. Si el cluster es pequeño, se elige, como candidato a pivote, un elemento que se encuentre en el centro del cluster(*centroide*); caso contrario se eligen elementos que particionen uniformemente el cluster (*facilities*).

4 Trabajo Futuro

En el caso del SAT se prevé analizar su comportamiento frente a otro tipo de operaciones no consideradas aún y existentes en Bases de Datos Espaciales. También se estudiará cómo modificarlo, a fin de obtener una buena estructura para memoria secundaria. Claramente en este aspecto puede ayudar el definir la aridad máxima, de manera tal de asegurar que entra todo en una página en memoria secundaria.

En el caso del $GNAT$, se prevé, en función de los resultados que se obtengan con la estrategia planteada, establecer una política eficiente de selección de pivotes y de elección de aridad de cada nodo. El objetivo también es, poder determinar cuál es el r óptimo para la construcción del *Range r Graph*, y analizar la forma más conveniente de tratar el conjunto de puntos outliers.

Referencias

[1] R. Baeza Yates, N. Herrera, N. Reyes. Búsquedas en espacios métricos: posibles optimizaciones al Geometric Near-neighbor Access Tree *In Actas, III Workshop de Investigadores en Ciencias de la Computacin (WICC'01), 2001*

[2] S. Brin. Near neighbor search in large metric spaces. In *Proc. of the 21st Conference on Very Large Databases (VLDB'95)*, pages 574–584, 1995.

[3] E. Chávez and G. Navarro. An effective clustering algorithm to index high dimensional metric spaces. In *Proc. 7th South American Symposium on String Processing and Information Retrieval (SPIRE'00)*, pages 75–86. IEEE CS Press, 2000.

[4] E. Chávez. A Subquadratic Algorithm for Cluster and Outlier Detection in Massive Metric Data. *in Proceesings of SPIRE'2001, IEEE Press, 2001.*

[5] E. Chávez and G.Navarro. A probabilistic spell for the curse of dimensionality. In *Proc. 3rd Workshop on Algorithm Engineering and Experiments (ALENEX'01)*, pages 147–160, LNCS 2153, 2001.

[6] E. Chávez, G.Navarro, R. Baeza-Yates, and J. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 2001. To appear.

[7] N. Herrera, N. Reyes and G.Navarro. Búsquedas en espacios métricos: árbol de aproximación espacial dinámico *In Actas, III Workshop de Investigadores en Ciencias de la Computacin (WICC'01), 2001*

[8] G. Navarro. Searching in metric spaces by spatial approximation. In *VLDB Journal*, 2002. To appear

[9] G. Navarro and N. Reyes. Dynamic spatial approximation trees. In *Proc. XXI Conference of the Chilean Computer Science Society (SCCC'01)*, 2001, pages 213–222. IEEE CS Press.