Uso de Grafos Geométricos para Calcular Join por Similitud en Espacios Métricos

Nora Reyes, Patricia Roggero
Departamento de Informática
Universidad Nacional de San Luis
(5700), San Luis, Argentina
Tel (+54 2652) 420823 – Fax 430224
{nreyes, proggero}@unsl.edu.ar

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

1. Introducción

A diferencia de la aproximación para bases de datos tradicionales, la comunidad de Recuperación de Información siempre ha considerado los resultados de las búsquedas como una lista *rankeada* de objetos. Dada una consulta, algunos objetos son más relevantes a la especificación de la consulta que otros y los usuarios habitualmente están interesados en los objetos más relevantes, es decir los objetos cuyo ranking es más alto. Este paradigma de búsqueda recientemente se ha generalizado en un modelo en el cual un conjunto de objetos pueden sólo compararse de a pares a través de una medida de distancia que satisface las propiedades de un *espacio métrico* [CBNM2001], [Samet2005], [ZADB2006].

Por ejemplo, considerar los datos de textos como el tipo más común de datos usado para recuperación de la información. Como el texto es habitualmente representado como una secuencia de caracteres, los pares de secuencias se pueden comparar y decidir la *coincidencia exacta*. Sin embargo, a medida que las secuencias son más largas es menos significativa la coincidencia exacta: las secuencias pueden contener errores de cualquier tipo y aún las secuencias correctas pueden tener pequeñas diferencias. De acuerdo a [Kukich1992], los textos típicos contienen cerca del 2% de errores de tipeo y ortográficos. Esto motiva una búsqueda que permita errores, o *búsqueda aproximada* o *búsqueda por similitud*, la cual requiere una definición del concepto de *similitud*, además de un algoritmo para evaluarla.

Otras numerosas aplicaciones modernas de sistemas de bases de datos, tales como multimedia, biología molecular, medicina y análisis de series de tiempo, entre otras, necesitan también poderosas herramientas de búsqueda. Los objetos de datos en estas aplicaciones son comúnmente representados por vectores de alta dimensión. Una operación común entre esos datos es encontrar los *k* objetos más cercanos o más similares a un objeto de consulta dado, lo cual se traduce en una consulta de los *k* vecinos más cercanos en espacios de alta dimensión.

Dentro del conjunto de operaciones de búsqueda por similitud, sin duda, son necesarios los *joins* o *ensambles por similitud* [DGZ2003], [DGSZ2003]. Por ejemplo, considerar una colección de documentos de libros y una colección de documentos de discos compactos. Una posible consulta podría requerir encontrar *todos los pares de libros y discos compactos con títulos similares*. Sin embargo, el join por similitud no es sólo aplicable a textos. En nuestro caso nos interesa el problema desde una perspectiva más amplia y suponemos que las medidas de distancia son métricas, de forma tal que extendemos el rango de los posibles tipos de datos a la

dimensión multimedia, lo que es más habitual para sistemas de recuperación de información modernos.

Para resolver el join o ensamble por similitud con rango *r* de una base de datos *X* respecto de otra *Y*, podemos pensar en que queremos emparejar cada elemento de *X* con aquellos elementos de *Y* que sean suficientemente similares, a distancia a lo más *r*. Este emparejamiento se puede representar por un *grafo de disco unitario* (GDU) *bipartito* en el cual los vértices son objetos del universo que pertenecen a *X* o a *Y*. Dos nodos, uno desde *X* y otro desde *Y*, estarían conectados por un arco si la distancia entre los objetos es menor que una unidad dada, donde la unidad representa un radio de similitud *R*, el cual esperamos que sea mayor que los radios de similitud habituales con los que luego pretenderemos resolver el join.

Es claro que no se puede mantener completo el grafo GDU que se genere, se necesita mantener un subgrafo, llamado *subgrafo geométrico*, que nos sea útil desde el punto de vista de la resolución del join. Existen numerosos tests para obtener subgrafos geométricos con alguna característica particular, que dado el grafo GDU determinan para cada arco si éste se mantendrá o no en el subgrafo. De todos estos tests se planea utilizar el test de *Half-Space Proximal* (*HSP*) [CDKOSTU2006], pero dado que la idea es trabajar en espacios métricos y no en el espacio euclidiano, se adaptará HSP a espacios métricos con el objetivo de resolver join por similitud.

2. Conceptos Previos

Un tipo de consulta, que combina elementos de dos bases de datos *X* e *Y*, donde $X, Y \subseteq U$, es el *join por similitud* [DGZ2003], [DGSZ2003]. En este caso nos concentraremos en resolver el *join por rango*, ya que es posible obtener a partir de él los otros considerando un valor de *r* decreciente.

Join por rango $X \rhd\lhd_r Y$: obtener todos los pares desde el producto cartesiano de *X* x *Y* tales que se encuentran a lo sumo a distancia *r* entre sí; es decir, obtener el conjunto $\{ (x, y) / x \in X \wedge y \in Y \wedge d(x, y) \leq r \}$.

La Figura 1 muestra un ejemplo de un join por rango entre *X* e *Y*, considerando que ambos son conjuntos de puntos de $\mathbf{R}^2$, en el que mediante flechas bidireccionales mostramos qué pares se formarían entre elementos de *X* e *Y*, y donde para mayor claridad mostramos circunferencias de radio *r* alrededor de cada elemento de *X* que evidencien qué elementos de *Y* aparecen dentro del radio y los elementos de *X* e *Y* en diferentes colores y formas.

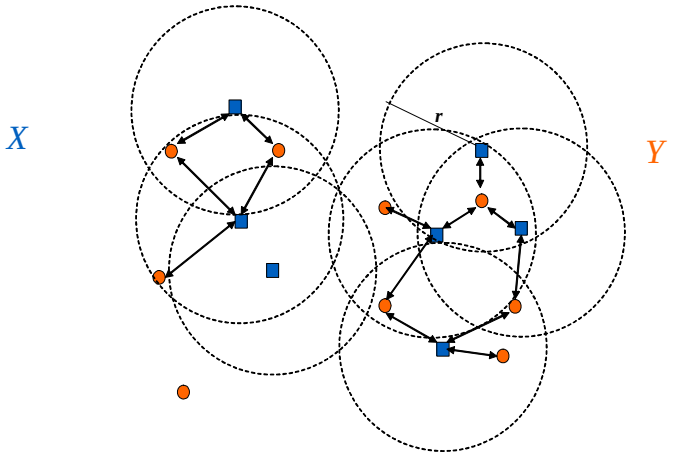


Figura 1: Ejemplos de un join por rango $X \rhd\lhd_r Y$ entre dos bases de datos *X* e *Y*, como conjuntos de $\mathbf{R}^2$.

3. Half-Space Proximal

Half-Space Proximal (HSP) es un nuevo test, propuesto en [CDKOSTU2006] para construir un *t*-spanner de un GDU. Sin embargo, a diferencia del test de Yao [Yao1982], el test HSP aplicado a una rotación del grafo GDU *G* produce una rotación del grafo geométrico (*spanner*) HSP de *G*. Por lo tanto, las propiedades del grafo geométrico HSP son independientes de la orientación del GDU en el plano.

Test HSP

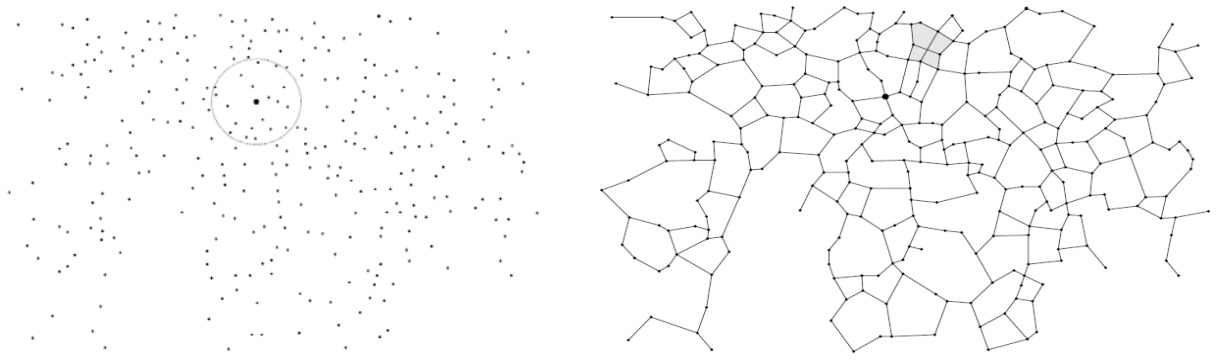
Para definir el test HSP se asume que el grafo $G = (V, E)$ es un GDU donde cada nodo *v* tiene coordenadas v_x , v_y en el plano Euclidiano y cada vértice está asignado a un rótulo entero único.

Entrada: un vértice *u* de un grafo geométrico y una lista L_1 de los arcos incidentes con *u*.

Salida: Una lista de arcos dirigidos L_2 que se retienen para el grafo $\vec{HSP}(G)$.

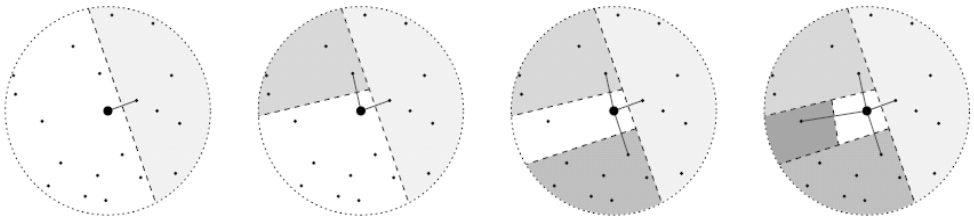
1. Colocar el área prohibida $F(u)$ como $\varnothing$
2. Repetir lo siguiente mientras L_1 no sea vacía
   a. Eliminar desde L_1 el arco más corto, digamos $[u,v]$, (cualquier empate es resuelto por el rótulo del vértice final) e insertar en L_2 un arco dirigido (u, v) con *u* siendo el vértice inicial.
   b. Agregar a $F(u)$ el semiplano abierto determinado por la línea perpendicular a un arco $[u,v]$ en el medio del arco y conteniendo el vértice *v* (Notar que los puntos sobre la línea no pertenecen al área olvidada)
   c. Revisar la lista L_1 y eliminar desde ella cualquier arco cuyo vértice esté en $F(u)$.

Una ilustración del test HSP aplicado a un GDU, con puntos en el espacio Euclidiano, se muestra en la Figura 2, se hace un acercamiento a un nodo seleccionado y el área prohibida aparece sombreada.



(a) Conjunto de puntos originales, notar el acercamiento realizado en (b).

(b) El grafo geométrico HSP



(c) Primer Vecino (d) Segundo Vecino (e) Tercer Vecino (f) Cuarto Vecino

Figura 2: Aplicación del test HSP a un GDU. El conjunto original de puntos en el espacio Euclidiano (a), y el grafo no dirigido resultante al aplicar el test (b). Un acercamiento alrededor de la vecindad de un nodo seleccionado (c) … (e). Observar el área sombreada en (b).

4. Join por Similitud usando HSP

Como se ha mencionado se adaptará el test HSP para aplicarlo a espacios métricos y además para que sea útil al momento de resolver el problema del join por rango entre dos bases de datos $X, Y \subseteq U$: $X \triangleright\triangleleft_r Y$.

Primero se podría fijar un radio R considerado la distancia unidad y el cual se debería elegir de manera tal que cubra los radios de join habituales entre X e Y. Luego, se construiría el grafo GDU bipartito de forma tal que los arcos vayan de elementos de X a elementos de Y siempre que su distancia d sea menor o igual a R.

Aunque en principio el test HSP se plantea para espacios Euclídeos, se puede observar que la eliminación que lleva a cabo el test también se puede aplicar en espacios métricos, hablando de hiperplanos en lugar de semiplanos entre dos elementos.

Entonces, luego de haber obtenido el grafo GDU bipartito G se aplicaría el siguiente test HSP, adaptado a espacios métricos y a nuestro problema, a cada elemento de X para quedarnos con el grafo $\vec{HSP}(G)$ que se obtendría con el siguiente algoritmo. Además se debería mantener alguna información extra, con el fin de poder resolver más eficientemente los joins y aprovechar más aún las distancias calculadas [WS1990].

Test HSP Adaptado a Espacios Métricos

Se asume que el grafo $G = (V, E)$ es un GDU bipartito, donde $V = X \cup Y$, donde cada nodo v está asignado a un rótulo entero único.

Entrada: un vértice u de un grafo geométrico y una lista L_1 de los arcos incidentes con u.

Salida: Una lista de arcos dirigidos L_2 que se retienen para el grafo $\vec{HSP}(G)$, donde además con cada arco se guarda su distancia y un conjunto de elementos de Y, junto con una distancia.

1. Colocar el conjunto prohibido $F(u)$ como $\varnothing$
2. Repetir lo siguiente mientras L_1 no sea vacía
   a. Eliminar desde L_1 el arco más corto, digamos (v, u) con d_{vu} (cualquier empate es resuelto por el rótulo del vértice final) e insertar en L_2 un arco dirigido (v, u) con u siendo el vértice final, junto con la distancia d_{vu}.
   b. Agregar a $F(u)$ los objetos y de Y tales que $d(u, y) < d(v, y)$, es decir aquéllos que se encuentran en el hiperplano que separa los elementos más cercanos a u de los más cercanos a v (Notar que los puntos que se encuentran a igual distancia de v que de u no pertenecen a la región olvidada) y registrar la distancia máxima desde u a un elemento en $F(u)$ como $dmáx(u)$.
   c. Guardar junto con el arco (v,u) con d_{vu} en L_2 el conjunto $F(u)$ y la distancia máxima entre u y los elementos en $F(u)$ $dmáx(u)$.
   d. Revisar la lista L_1 y eliminar desde ella cualquier arco cuyo vértice esté en $F(u)$.

Ahora que ya se ha determinado el grafo geométrico sobre el cual vamos a trabajar, falta analizar cómo usaríamos este grafo para resolver el join por rango $X \triangleright\triangleleft_r Y$. Por lo tanto el trabajo propuesto es el de determinar cómo realizar el join por rango $X \triangleright\triangleleft_r Y$, entre las bases de datos X e Y, usando el grafo $\vec{HSP}(G)$ obtenido desde el GDU con radio unitario R, donde consideramos que $R \geq r$.

5. Conclusiones

La búsqueda por similitud es un concepto importante en la recuperación de información moderna. Sin embargo los costos computacionales de las funciones de similitud son generalmente altos, considerar por ejemplo la complejidad computacional del cálculo de la distancia de edición entre cadenas de caracteres que es cuadrática.

Aunque existen numerosas técnicas de indexación para soportar las búsquedas por rango o de los *k* vecinos más cercanos, existen sólo unos escasos estudios de índices para los joins por similitud.

Aquí hemos iniciado el estudio de un nuevo método, que a través de conceptos de teoría de grafos y de una adaptación del test HSP, recientemente presentado, nos permitirá diseñar un algoritmo que resuelva el $X \triangleright\triangleleft_r Y$ entre dos bases de datos *X* e *Y*.

Como trabajo futuro, además hay que analizar el desempeño tanto teórico como experimental del algoritmo y demostrar que resuelve el join por rango $X \triangleright\triangleleft_r Y$. También nos resta analizar cómo podemos usar nuestro grafo para resolver el join por rango cuando $r > R$ y qué sucede si $X = Y$, porque es posible que en ese caso algo se pueda simplificar en nuestro proceso.

Para el análisis experimental, deberíamos no sólo probar el algoritmo sobre distintos tipos de espacios métricos, de alta y baja dimensión, sino también compararlo contra la solución trivial de calcular las distancias de todos contra todos, contra la solución de realizar una búsqueda por rango $(q, r)_d$ de cada elemento de *X* dentro de la base de datos *Y* y contra algunas de las soluciones existentes para resolverlo.

6. Referencias

[CDKOSTU2006] E. Chávez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho and J. Urrutia: Half-Space proximal: a new local test for extracting a bounded dilation spanner of a unit disk graph. *Proceedings of OPODIS* 2005, LNCS 3974, 235-245, 2006. Springer.

[CNBM2001] E. Chávez, G. Navarro, R. Baeza-Yates, and J. Marroquín. Searching in metric spaces. ACM Computing Surveys, 33(3):273 - 321, September 2001.

[DGSZ2003] Dohnal, V., Gennaro, C., Savino, P., and Zezula, P. 2003. D-index: Distance searching index for metric data sets. Multimedia Tools and Applications 21, 1, 9 - 33.

[DGZ2003] V. Dohnal, C. Gennaro, and P. Zezula. Similarity join in metric spaces using eD-index. In Proc. 14th Intl. Conf. on Database and Expert Systems Applications (DEXA'03), LNCS 2736, pages 484–493, 2003.

[Kukich1992] K. Kukich: Techniques for automatically correcting words in text. *ACM Computing Surveys*, 1992, 24(4) : 377 – 439.

[Samet2005] Samet, H. 2005. Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[WS1990] T. Wang, D. Shasha: Query Processing for Distance Metrics, Proceedings of the 16th VLDB Conference Brisbane. 1990, 602 - 613.

[Yao1982] A.C.-C. Yao, On constructing minimum spanning trees in k- dimensional spaces and related problems, *SIAM Journal on Computing* 11(4) (1982) 721-736.

[ZADB2006] Zezula, P., Amato, G., Dohnal, V., and Batko, M. 2006. Similarity Search: The Metric Space Approach. Advances in Database Systems, vol. 32. Springer.