View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Servicio de Difusión de la Creación Intelectual

Bases de Datos Métricas *

Cristian Bustos, Verónica Ludueña, Nora Reyes
Departamento de Informática, Universidad Nacional de San Luis.
{cjbustos,vlud,nreyes}@unsl.edu.ar
y
Gonzalo Navarro
Departamento de Ciencias de la Computación, Universidad de Chile.
gnavarro@dcc.uchile.cl

Resumen

Con la evolución de las tecnologías de información y comunicación, han surgido almacenamientos no estructurados de información. No sólo se consultan nuevos tipos de datos tales como texto libre, imágenes, audio y vídeo; sino que además, en algunos casos, ya no se puede estructurar más la información en claves y registros. Estos escenarios requieren un modelo más general tal como *bases de datos métricas*.

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. En particular, nos vamos a dedicar a cómo resolver eficientemente no sólo las búsquedas, sino también a aplicaciones y algunas otras cuestiones de interés en el ámbito de las bases de datos métricas. Así, la investigación apunta a poner estas nuevas bases de datos a un nivel de madurez similar al de las tradicionales.

Palabras Claves: bases de datos, espacios métricos, consultas, algoritmos, estructuras de datos.

1. Introducción y Motivación

Con la evolución de las tecnologías de información y comunicación, han surgido almacenamientos no estructurados de información. No sólo se consultan nuevos tipos de datos tales como texto libre, imágenes, audio y vídeo; sino que además, en algunos casos, ya no se puede estructurar más la información en claves y registros. Aún cuando esta estructuración sea posible, nuevas aplicaciones tales como la minería de datos requieren acceder a la base de datos por cualquier campo y no sólo por los marcados como "claves".

Los escenarios anteriores requieren un modelo más general tal como *bases de datos métricas* (es decir, bases de datos que incluyan objetos de un espacio métrico), entre otros; y contar con herramientas que permitan realizar búsquedas eficientes sobre estos tipos de datos. Las técnicas que emergen desde estos campos muestran un área de investigación propicia para el desarrollo de herramientas que resuelvan eficientemente los problemas involucrados en la administración de estos tipos de bases de datos no convencionales.

La búsqueda por similitud es un tema de investigación que abstrae varias nociones de las ya mencionadas. Este problema se puede expresar como sigue: dado un conjunto de objetos de naturaleza desconocida, una función de distancia definida entre ellos, que mide cuan diferentes son, y dado otro objeto, llamado la consulta, encontrar todos los elementos del conjunto suficientemente similares a la consulta. El conjunto de objetos junto con la función de distancia se denomina espacio métrico [6].

En algunas aplicaciones, los espacios métricos resultan ser de un tipo particular llamado "espacio vectorial", donde los elementos consisten de D coordenadas de valores reales. Existen muchos trabajos que explotan las propiedades geométricas sobre espacios vectoriales (ver [10] para más detalles); pero normalmente éstas no se pueden extender a los espacios métricos generales.

Por otra parte, una base de datos de texto es un sistema que debe proveer acceso eficiente a grandes volúmenes de texto no estructurado, donde se necesitan construir índices que permitan realizar búsquedas eficientes de patrones ingresados por el usuario. El texto se puede ver como una secuencia de símbolos y el patrón a buscar como otra más breve, y así un problema de búsqueda puede consistir en encontrar todas las apariciones del patrón en el texto admitiendo un número pequeño de errores, lo que se puede encuadrar en el contexto de las bases de datos métricas.

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. En particular, nos vamos a dedicar a optimizar las estructuras y los algoritmos de búsqueda en esos ámbitos. Además, como las estructuras y algoritmos para búsquedas en espacios métricos sufren de la así llamada "maldición de la dimen-

*Este trabajo ha sido financiado parcialmente por la red RITOS2 de CYTED (todos los autores) y Núcleo Milenio Centro de Investigación de la Web, Proyecto P04-067-F, Mideplan, Chile (último autor).

sionalidad", nos interesa también estudiar la manera de estimar la dimensión intrínseca de un espacio métrico con el fin de elegir el índice que más convenga en cada aplicación particular.

2. Bases de Datos Métricas

El planteo general del problema en bases de datos métricas 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. El conjunto $\mathcal{U}$ 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 $d : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}^+$ cumple con las propiedades propias de una función de distancia.

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

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

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 en la mayoría de los casos la cantidad de evaluaciones de distancias necesarias para resolver la búsqueda, se usa como medida de complejidad.

Integrar objetos de un espacio métrico en un ambiente de bases de datos requiere principalmente poder resolver consultas por similaridad eficientemente. En aplicaciones reales es posible tener grandes volúmenes de datos, ya sea por la cantidad de objetos o por el tamaño de cada objeto. Ello implica que debemos contar con estructuras adecuadas y eficientes para memoria secundaria.

Además en un escenario real de un ambiente de base de datos es necesario contar con herramientas que permitan trabajar con grandes volúmenes de datos y usualmente serían *dinámicas*. Es decir, deberían permitir insertar, o eliminar objetos dinámicamente, por lo tanto los índices deberían soportar estas operaciones eficientemente. Un objeto de un espacio métrico puede ser una imagen, una huella digital, un documento, o cualquier otro tipo de objeto. Esta es una de las razones por las que los elementos de una base de datos métrica no se pueden almacenar en bases de datos relacionales, las cuales tienen tamaño fijo de tupla. Esto también implica que las operaciones sobre datos de un espacio métrico son, en general, más costosas que las operaciones relacionales estándar.

Existen índices que, en principio, resuelven estos tipos de problemas; pero aún están muy inmaduros para ser usados en la vida real por dos motivos importantes: *falta de dinamismo* y *necesidad de trabajar en memoria principal*. Estas características son sobreentendidas en los índices para bases de datos tradicionales, y la investigación apunta a poner los índices para estas nuevas bases de datos a un nivel de madurez similar.

3. Optimización de Estructuras

Hemos desarrollado una estructura para búsqueda por similitud en espacios métricos llamado *Árbol de Aproximación Espacial Dinámico (SATD)* [13] que permite realizar inserciones y eliminaciones, manteniendo una buen desempeño en las búsquedas. Muy pocos índices para espacios métricos son completamente dinámicos. Esta estructura se basa en el *Árbol de Aproximación Espacial* [12], el cual había mostrado un muy buen desempeño en espacios de mediana a alta dimensión, pero era completamente estático.

3.1. Optimización del *SAT*

Para mostrar la idea general de la aproximación espacial se utilizan las *búsquedas del vecino más cercano*. Las aproximaciones son efectuadas sólo vía los "vecinos". Cada elemento $a \in \mathcal{S}$ 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 posicionándose sobre un nodo arbitrario a y considerando todos sus vecinos. Si ningún nodo está más cerca de q que a, entonces a es 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. La construcción del árbol se hace también de manera recursiva. De la definición se observa que se necesitan de antemano todos los elementos para la construcción y que queda completamente determinado al elegirle una raíz, lo cual actualmente se realiza al azar. Por lo tanto, es posible pensar en que se podría mejorar su comportamiento en las búsquedas si se eligiera la raíz usando el conocimiento sobre el espacio métrico particular a ser indexado.

Actualmente se están analizando diversas maneras de elegir la raíz, prepocesando el conjunto de elementos, y se ha observado que alguna de ellas consigue mejorar significativamente los costos de las búsquedas, a costa de aumentar en algunos casos los costos de construcción. Entonces, se deberá analizar para la aplicación particular cuál de los métodos obtiene el mejor tradeoff entre búsqueda y construcción.

3.2. Actualización de *SATD*

Para el desarrollo del *SATD* [13] se han estudiado distintas maneras de realizar incorporaciones de nuevos elementos sobre el árbol, pero se optó por un

método que inserta un nuevo elemento en un punto determinado del árbol, manteniendo la aridad acotada. Gracias a esos trabajos, quedó demostrado que existen otros posibles puntos de inserción válidos, aunque no se analizó cuál de ellos sería el mejor.

Por lo tanto, tiene sentido considerar si los otros puntos posibles de inserción para un elemento consiguen mejorar aún más los costos de búsqueda. Para ello, cada vez que un elemento elige un punto de inserción, evaluamos cómo se comportaría la estructura durante las búsquedas si insertáramos al elemento más profundo en el árbol. De esta manera es posible que el árbol resultante sea menos "ancho" y más profundo que el que se hubiera obtenido con el *SATD* original. Cabe destacar que en el *SATD*, a diferencia de los árboles de búsqueda tradicionales, se ha mostrado que mantener aridad baja en los primeros niveles en algunos casos mejora las búsquedas.

Como resultado se espera brindar una estructura que mejore el comportamiento durante las búsquedas gracias a elegir adecuadamente los puntos de inserción de los nuevos elementos.

3.3. Combinando *SATD* con Clustering

El *SATD* es una estructura que realiza la partición del espacio considerando la proximidad espacial, pero si el árbol lograra agrupar los elementos que se encuentran muy cercanos entre sí, lograría mejorar las búsquedas al evitarse el recorrerlo para alcanzarlos. Así, nos hemos planteado el estudio de una nueva estructura de datos que realice la aproximación espacial sobre clusters o grupos de elementos, en lugar de elementos individuales.

Podemos pensar entonces que construimos un *SATD*, con la diferencia que cada nodo representa un grupo de elementos muy cercanos ("clusters") y relacionamos los clusters por su proximidad en el espacio. La idea sería que en cada nodo se mantenga un elemento, al que se toma como el centro del cluster correspondiente, y se almacenen los k elementos más cercanos a él; cualquier elemento a mayor distancia del centro que los k elementos, pasaría a formar parte de otro nodo en el árbol, que podría ser un nuevo vecino en algunos casos.

La búsqueda de un elemento q con radio r debería proceder de manera similar al *SATD*, es decir realizando aproximación espacial entre los centros de los nodos. Sin embargo, al tener clusters en los nodos debemos además verificar si hay o no intersección entre la zona consultada y el cluster. Más aún, si no la hay se pueden descartar todos los elementos del cluster, sin necesidad de compararlos contra q. Si el cluster no se pudo descartar para la consulta, es posible usar al centro de cada nodo como un pivote para los elementos x_i que se encuentran en el cluster, ya que además mantenemos las distancias $d(a, x_i)$ respecto del centro a. De esta manera es posible que, evitemos algunos cálculos de distancia entre q y los x_i, si $|d(q, a) - d(a, x_i)| > r$. Cabe destacar que si la zona consultada cae completamente dentro de un cluster, podemos estar seguros que en ninguna otra parte del árbol encontraremos elementos relevantes para esa consulta [1].

3.4. *SATD* en Memoria Secundaria

Estamos actualmente desarrollando una versión del *SATD* que funcione adecuadamente en memoria secundaria, manteniendo su buen desempeño en las búsquedas y el dinamismo. Así, sería posible pensar en extender apropiadamente el álgebra relacional y diseñar soluciones eficientes para los nuevos operadores, teniendo en cuenta aspectos no sólo de memoria secundaria, sino también de concurrencia, confiabilidad, etc. Algunos ejemplos de las operaciones que podrían ser de interés resolver son: join espacial, operaciones de conjuntos y otras operaciones de interés en bases de datos espaciales tales como los operadores topológicos. Algunos de estos problemas ya poseen solución en las bases de datos espaciales, pero no en el ámbito de los espacios métricos.

En este caso no sólo se busca minimizar la cantidad de cálculos de la función de distancia, sino también la cantidad de operaciones de entrada/salida, lo que permitiría utilizarla en aplicaciones reales.

4. Estimadores de Dimensión Intrínseca

En los espacios de vectores, la "maldición de la dimensionalidad" describe el fenómeno por el cual el desempeño de todos los algoritmos existentes se deteriora exponencialmente con la dimensión. En espacios métricos generales la complejidad se mide como el número de cálculos de distancias realizados, pero la ausencia de coordenadas no permite analizar la complejidad en términos de la dimensión.

En los espacios vectoriales existe una clara relación entre la dimensión (*intrínseca*) del espacio y la dificultad de buscar. Se habla de "intrínseca", como opuesta a "representacional", porque un conjunto finito de puntos puede tener una alta dimensión representacional (la dimensión del espacio de vectores) pero ellos se pueden transformar en vectores de dimensión más baja, sin afectar demasiado sus distancias. Los algoritmos más ingeniosos se comportan más de acuerdo a la dimensión intrínseca que a la representacional.

Existen varios intentos de medir la dimensión intrínseca en espacios de vectores, tales como la transformada de *Karhunen-Loève (KL)* y otras relacionadas, más fáciles de computar; medidas tales como *Fastmap* [9]. Otro intento útil para medir la dimensión intrínseca, de espacios de vectores no uniformemente distribuidos, es la *dimensión fractal* [2].

Existen sólo unas pocas propuestas diferentes sobre cómo estimar la dimensión intrínseca de un espacio métrico tales como el *exponente de la distancia* (basada en una ley de potencias empírica observada en mu-

chos conjuntos de datos) [11], y la medida de dimensión intrínseca como una medida cuantitativa basada en el histograma de distancias [3]. Además, también parece posible adaptar algunos de los estimadores de distancia para espacios de vectores para aplicarlos a espacios métricos generales, como por ejemplo *Fastmap* y *dimensión fractal*.

En aplicaciones reales de búsqueda en espacios métricos, sería muy importante contar con un buen estimador de la dimensión intrínseca porque esto nos permitiría decidir el índice adecuado a utilizar en función de la dimensión del espacio. Además, al conocer una buena estimación de la dimensión nos permitiría, en algunas ocasiones, elegir la función de distancia de manera tal que se obtenga una menor dimensión.

5. Búsqueda por Similaridad en Textos

Una *base de datos de texto* es un sistema que provee acceso eficiente a amplias masas de datos textuales. El requerimiento más importante es que desarrolle búsquedas rápidas para patrones ingresados por el usuario. En general el escenario más simple aparece es como sigue: el texto $T_{1...u}$ se ve como una única secuencia de caracteres sobre un alfabeto Σ de tamaño σ, y el patrón de búsqueda $P_{1...m}$ como otra (breve) secuencia sobre Σ.

Existen muchas aplicaciones en las cuales tiene sentido buscar todas las ocurrencias de un patrón P, pero admitiendo que nuestras respuestas contengan algunos errores. Esto es un problema abierto en la búsqueda combinatoria de patrones, por lo tanto hay que diseñar nuevos índices que permitan responder eficientemente esta clase de consulta.

El problema de búsqueda aproximada es: dado un gran texto T de longitud n y un patrón P de longitud m (comparativamente más corto) y un valor r, devolver todas las ocurrencias del patrón, es decir, todos los substrings cuya *distancia de edición* (o *distancia de Levenshtein*) a P es a lo sumo r. La *distancia de edición* entre dos strings se define como la mínima cantidad de inserciones, supresiones o substituciones de caracteres necesaria para que los strings sean iguales. Esta distancia es usada en muchas aplicaciones, pero cualquier otra distancia puede ser de interés.

Recientemente ha recibido mayor atención la posibilidad de indexar el texto para la búsqueda aproximada de strings. Sin embargo, la mayoría de los esfuerzos se orientan a la búsqueda en textos de lenguaje natural y las soluciones no se pueden extender a casos generales como ADN, proteínas, símbolos orientales, etc.

La aproximación [4] considera que la distancia de edición satisface la *desigualdad triangular* y así define un *espacio métrico* sobre el conjunto de substrings de T. Así se puede replantear el problema como uno de búsqueda por rango sobre un espacio métrico.

Una propuesta de indexación del texto para búsqueda aproximada de strings, usando técnicas de espacios métricos, tiene el problema que hay $O(n^2)$ diferentes substrings en un texto y además habría que indexar $O(n^2)$ objetos, lo cual es inaceptable. En esta aproximación la idea es indexar los n sufijos del texto. Cada sufijo $[T_{j...}]$ representa todos los substrings que comienzan en la posición j. Existen diferentes opciones de cómo indexar este espacio métrico formado por $O(n)$ conjuntos de strings, la elegida aquí es una propuesta basada en *pivotes*. Entonces seleccionamos k pivotes y para cada conjunto de sufijos $[T_{j...}]$ del texto y cada pivote p_i, computamos la distancia entre p_i y todos los strings representados por $[T_{j...}]$. De ese conjunto de distancias desde $[T_{j...}]$ a p_i, se almacena sólo la mínima obtenida.

Notar que no necesitamos construir ni guardar los sufijos, ya que ellos se pueden obtener e indexar directamente del texto. Así, la única estructura necesaria sería el *índice métrico*.

Si se considera la búsqueda de un patrón dado P con a lo sumo r errores se está ante un query por rango de radio r en el espacio métrico de sufijos. Como en todos los algoritmos basados en pivotes comparamos el patrón P contra los k pivotes y obtenemos una coordenada k-dimensional $(ed(P, p_1), \ldots, ed(P, p_k))$.

Sea p_i un pivote dado y los sufijos $[T_{j...}]$ de T, si se cumple que $ed(P, p_i) + r < \text{mín}(ed([T_{j...}], p_i))$, por la desigualdad triangular se sabe que $ed(P, [T_{j...}])) > r$ para todo substring que esté representado por $[T_{j...}]$. La eliminación se puede hacer usando cualquier pivote p_i. Al buscar qué sufijos podrán ser eliminados se cae en un clásico problema de búsqueda por rango en un espacio multidimensional, así para esta tarea se podría usar el *R-tree*, o alguna de sus variantes. Los nodos que no puedan ser eliminados usando algún pivote deberán ser directamente comparados contra P y para aquellos cuya distancia mínima a P sea a lo sumo r, se reportarán todas sus ocurrencias.

6. Trabajos Futuros

Como trabajo futuro de esta línea de investigación se considera:

SAT: Poder evaluar empíricamente la nueva variante de construcción que se obtenga no sólo contra la versión original de la estructura, sino también sobre otras estructuras competitivas. Además se intentará ver cómo aprovechar, si es posible, la optimización propuesta para el *SAT* a su versión dinámica el *SATD*.

Actualización del SATD: Se espera poder adquirir un conocimiento más acabado sobre el *SATD* a través de la experimentación y proponer finalmente una nueva versión que mejore aún más los costos tanto de las operaciones de actualización, sino también la construcción y las búsquedas.

Clustering+SATD: Esperamos poder evaluar el comportamiento de la estructura propuesta experimentando sobre distintos espacios métricos.

Además pretendemos hacer comparaciones contra otras estructuras tal como la *Lista de Clusters* propuesta en [5].También queremos analizar, entre otras, cuestiones tales como: ¿esta estructura sería eficiente en memoria secundaria?, ¿es posible mantener el cluster separado del nodo para obtener una mejor opción para memoria secundaria?, ¿sería más adecuado mantener los clusters con cantidad fija de elementos o con radio fijo?, ¿existen otras maneras de combinar estas técnicas para búsquedas en espacios métricos?, entre otras.

SATD en Memoria Secundaria: Para evaluar empíricamente nuestra estructura consideramos compararla con las únicas dos estructuras de datos que actualmente permiten dinamismo y están diseñadas especialmente para memoria secundaria: *M-tree* [7] y *D-index* [8]. Con la estructura que resulte más competitiva en memoria secundaria se estudiará la manera de utilizarla como base para implementar otras operaciones de interés sobre bases de datos métricas tal como el join por similaridad.

Estimadores de Dimensión Intrínseca: Luego de adaptar algunas medidas de dimensionalidad intrínseca a espacios métricos y evaluarlas se espera evaluar cuál de ellas permite predecir más adecuadamente la facilidad o dificultad que se encontraría al realizar búsquedas sobre un espacio métrico. Esto nos permitiría no sólo predecir de alguna manera los costos de búsqueda sobre un espacio métrico, sino también elegir el índice más adecuado a su dimensionalidad.

Aplicación a Bases de Datos de Texto: Como ya se ha realizado la implementación de la aproximación presentada, estamos actualmente realizando experimentos para analizar su competitividad. Además se pretende mejorar esta propuesta basándonos en el conocimiento de que los pivotes con distancias mínimas grandes permiten la eliminación de una mayor cantidad de sufijos. Entonces se podrían almacenar para cada pivote p_i sólo los s valores más grandes de las distancias $\mathbf{min}(ed(p_i, [T_{j...}]))$; siendo s fijado de antemano, permitiendo descartar más elementos utilizando el mismo espacio. Otra posible mejora basada en tomar un primer pivote, determinar sus s sufijos más lejanos, almacenar esos sufijos y sus distancias mínimas en una lista en forma ordenada y luego descartar esos sufijos para próximas consideraciones. Esta idea puede ser eficiente en una implementación en memoria secundaria.

Referencias

[1] M. Barroso, G. Navarro, and N. Reyes. Combinando clustering con aproximación espacial para búsquedas en espacios métricos. En *Proc. del XI Congreso Argentino de Ciencias de la Computación (CACIC 2005)*, pág. 447–458, 2005.

[2] Francesco Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36(12):2945–2954, 2003.

[3] E. Chávez and G. Navarro. Towards measuring the searching complexity of metric spaces. En *Proc. International Mexican Conference in Computer Science (ENC'01)*, volume II, pág. 969–978, 2001.

[4] E. Chávez and G. Navarro. A metric index for approximate string matching. En *Proc. of the 5th Latin American Symposium on Theoretical Informatics (LATIN'02)*, LNCS 2286, pág. 181–195, 2002.

[5] E. Chávez and G. Navarro. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9):1363–1376, 2005.

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

[7] P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. En *Proc. of the 23rd Conference on Very Large Databases (VLDB'97)*, pág. 426–435, 1997.

[8] Vlastislav Dohnal, Claudio Gennaro, Pasquale Savino, and Pavel Zezula. D-index: Distance searching index for metric data sets. *Multimedia Tools Appl.*, 21(1):9–33, 2003.

[9] Christos Faloutsos and King-Ip Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. En *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pág. 163–174. ACM Press, 1995.

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

[11] Caetano Traina Jr., Agma J. M. Traina, and Christos Faloutsos. Distance exponent: A new concept for selectivity estimation in metric trees. En *ICDE*, pág. 195, 2000.

[12] G. Navarro. Searching in metric spaces by spatial approximation. *The Very Large Databases Journal (VLDBJ)*, 11(1):28–46, 2002.

[13] G. Navarro and N. Reyes. Fully dynamic spatial approximation trees. En *Proc. of the 9th International Symposium on String Processing and Information Retrieval (SPIRE 2002)*, LNCS 2476, pág. 254–270. Springer, 2002.