Un Lenguaje de Consulta Intuitivo para Textos Estructurados

Jesús Vegas Pablo de la Fuente
Dpto. de Informática, Universidad de Valladolid
Campus Miguel Delibes, 47011 Valladolid, España
{jvegas,pfuente}@infor.uva.es

Ricardo Baeza-Yates Gonzalo Navarro
Dpto. de Ciencias de la Computación, Universidad de Chile
Blanco Encalada 2120, Santiago, Chile
{rbaeza,gnavarro}@dcc.uchile.cl

Resumen

El aumento de la cantidad de información disponible hace que los sistemas de búsqueda basados en el contenido de los documentos no satisfagan las necesidades de los usuarios. En este informe se presenta un modelo de lenguaje de consulta que permite realizar consultas en bases de datos documentales considerando tanto su estructura como su contenido. Los aspectos más destacables del modelo son: permite incluir la estructura de los documentos en la consulta; el álgebra de consulta es potente, intuitiva y se puede implementar fácilmente en un sistema de consulta visual; el lenguaje es eficiente desde el punto de vista de su implementación y puede ser construido sobre índices tradicionales o incluso funcionar sin índice; está fuertemente orientado al usuario final, no hace uso de la lógica booleana; y permite una visualización flexible de los resultados en función de los términos contenidos en la consulta.

1 Introducción

En los últimos años la cantidad de información a la que se puede tener acceso electrónicamente ha aumentado de manera impresionante. Este incremento de la información accesible a través de la red ha propiciado la aparición de un gran número de motores de búsqueda. Una de las estrategias más recientes para conseguir que las consultas sean más precisas es la de intentar utilizar la estructura de los documentos. Esto es especialmente importante, ya que existen varios formatos estándares para representar documentos estructurados, por ejemplo SGML [7] y su caso particular HTML, cuyo uso está difundido ampliamente.

Ya existen varios modelos de bases de datos textuales que permiten consultar sobre la estructura y el contenido de los documentos. En la bibliografía se pueden encontrar algunos trabajos han intentado relacionar la eficiencia de las consultas con la posibilidad de tratar la estructura de los documentos [3]. Sin embargo, se han empleado menos esfuerzos en considerar el interés que lenguajes tan potentes tienen para el usuario.

La principal aportación de este trabajo es intentar cubrir esa carencia. Nosotros hemos definido un modelo de lenguaje de consulta que permite realizar consultas en bases de datos textuales preguntando sobre contenido y sobre estructura, que es intuitivo y, al mismo tiempo, expresivo. Este modelo puede ser implementado de varias maneras,

respondiendo a un subconjunto razonable de consultas en un tiempo lineal con respecto al tamaño de los resultados intermedios con índice, o sin él. Nuestro lenguaje de consulta no está basado en la lógica booleana, y utiliza pesos para ayudar en la ordenación de los documentos recuperados según su relevancia. Para facilitar la interacción con el usuario se complementa con un lenguaje de consulta visual.

Existen varias alternativas diferentes para construir índices de textos estructurados; y la mayoría son construídos *ad–hoc*. En cambio, nosotros, en este trabajo, hemos utilizado el tradicional fichero invertido con unas pocas mejoras para soportar consultas sobre contenido y estructura. Este tipo de índices ha sido bien estudiado en la literatura y su comportamiento es bien conocido [16, 6]. No en vano, son los índices más comunes en las bases de datos textuales. Esto hace que el modelo considerado sea más simple de añadir a un sistema de recuperación de información existente, para dotarle de la capacidad de tratar la estructura de los documentos en sus consultas.

El resto del trabajo se organiza como sigue. Primero se revisan los trabajos precedentes que ya se han desarrollado sobre el tema. Segundo, presentamos el modelo elegido, incluyendo el lenguaje tanto en la versión texto como en la visual. Tercero, se estudia un ejemplo que ilustre el trabajo realizado. Por último, se revisan las conclusiones obtenidas y se presenta el trabajo futuro.

2 Precedentes

Tradicionalmente, las bases de datos textuales han permitido la realización de búsquedas por su contenido (palabras, frases, etc.) o por su estructura (navegando por una tabla de contenidos o un índice), pero no ambas cosas a la vez. Recientemente, han aparecido muchos modelos que permiten mezclar ambos tipos de consultas.

Mezclando contenido y estructura en las consultas se puede conseguir que éstas sean muy potentes, siendo mucho más expresivas que si se construyeran utilizando el contenido o la estructura por separado. La utilización de un lenguaje que permita integrar ambos tipos de consulta, puede potenciar la calidad de la recuperación en las bases de datos textuales.

Sin embargo, a pesar de la cantidad de trabajo invertido en conseguir lenguajes potentes y expresivos para consultar sobre textos estructurados [13, 1, 15, 4, 11, 8, 5, 12], casi nada se ha hecho para abordar el problema desde el punto de vista del usuario.

En un sistema de Recuperación de Información clásico, RI, el proceso de consulta está bien definido. El usuario hace la consulta, el sistema responde y el usuario evalúa el resultado. Si las necesidades del usuario han sido satisfechas, el proceso acaba. En otro caso, el usuario puede proponer otra consulta, filtrando previamente la respuesta recibida en la consulta anterior o comenzando de nuevo.

Varios aspectos de un sistema de RI basado en el contenido varían con respecto a los orientados a la estructura y al contenido. En los primeros, el usuario no necesita conocer donde se encuentra la información, ya que la estructura no es importante. Además, los documentos son más o menos relevantes según contengan más o menos veces el contenido buscado. Pero cuando se añade la estructura a la consulta, el conocimiento que el usuario tiene de la organización del documento se convierte en esencial, y la relevancia de un documento no está sólo relacionada con el número de veces que satisfaga la consulta sobre el contenido, sino que también depende de dónde se encuentre la información. Existe muy poco trabajo hecho sobre el estudio de la relevancia para textos estructurados [14],

y tampoco se pueden encontrar demasiados trabajos sobre expansión de consultas y otros aspectos que en los sistemas de RI tradicionales están siendo estudiados desde hace tiempo.

Por lo tanto, es importante que el usuario conozca la estructura de los documentos sobre los que quiere consultar para que la pueda utilizar en la consulta. Así, la interfaz de consulta debe mostrar la estructura de los documentos y asistir el proceso de consulta. Sin embargo, casi todos los lenguajes de consulta que incluyen estructura [3, 10] suelen ser demasiado complejos para el usuario final. Esta es una de las principales razones por la que la estructura de los documentos no se utiliza, en general, para realizar búsquedas. Una de las posibles soluciones a este problema es utilizar un lenguaje de consulta visual, ya que este tipo de lenguajes es más cercano al usuario que los tradicionales tipo comando. Además, la utilización de lenguajes visuales permite expresar de manera más fácil las relaciones de inclusión entre objetos. Por otro lado, este tipo de lenguajes limita la expresividad de las consultas, aunque su principal baza es la simplicidad y la facilidad de uso. Existen muy pocos lenguajes que dispongan de estas características [9].

Otro problema es cómo mostrar la estructura de los documentos presentes en la respuesta. Una solución es la utilización de una metáfora visual que represente cada ocurrencia de la consulta en los documentos. Si fuera posible, esta metáfora debería ser similar a la utilizada para formular las consultas. La interfaz debería ayudar también al usuario para que ordenara los documentos recuperados y evaluara la precisión de la respuesta [14].

3 El Lenguaje de Consulta

Vamos a definir un lenguaje de consulta sobre Estructura y Contenido, EC, que cumpla con los siguientes objetivos:

- Permita consultar sobre estructura y contenido.
- Resulte más cercano al usuario que al sistema de recuperación.

Otros lenguajes de consulta sobre estructura y contenido aportan un conjunto de operadores más o menos completo que responden a necesidades planteadas desde el punto de vista de la expresividad, la eficiencia y el aprovechamiento óptimo de las posibilidades del sistema de indexación y recuperación [13, 5]. Además, el lenguaje básico de consulta es de tipo booleano, en el que se mezclan los operadores lógicos con los específicos del sistema para componer consultas más o menos sofisticadas. Nosotros, en cambio, hemos planteado un conjunto reducido de operadores que se utilizarán para componer consultas utilizando como enlace operadores lógicos pseudo-booleanos.

3.1 El Modelo de Documento

El modelo de documento que vamos a considerar es el que se define a continuación.

Sea C el conjunto de *elementos de contenido* que pueden encontrarse en un documento. Estará formado por cada una de las distintas formas en las que se puede presentar la información y que el usuario es capaz de captar: texto, imágenes, sonido, video, etc.

Sea E el conjunto de *elementos de estructura* que forman un documento. En él se definen tanto las diferentes estructuras que forman el documento como las relaciones jerárquicas que pueden existir entre ellas.

Así, se puede definir el *tipo de documento* $D(E, C)$ como el conjunto de todos aquellos documentos que contienen la estructura E y los elementos de contenido C.

Cuando se dice que $d \in D(E, C)$ se quiere expresar que d es un documento de tipo D con una estructura E y unos elementos de contenido C.

3.2 Las Consultas EC

El objetivo final es definir un lenguaje de consulta visual, pero éste ha de estar basado en un lenguaje formal. Siguiendo el trabajo realizado en [13], se define un lenguaje que puede ser utilizado como intermediario entre la interfaz de usuario y el motor de búsqueda. Este es el lenguaje EC.

Teniendo todo esto en cuenta se proponen sólo 2 operadores generales: *presencia* e *inclusión*. Estos operadores responden a la necesidad de preguntar por la presencia de un elemento y por la inclusión de un elemento en otro, respectivamente.

El primero de los operadores es el mismo que aparece en todos los demás lenguajes de consulta. El segundo viene justificado porque al añadir los elementos de estructura a la consulta, resulta natural preguntar por la inclusión. Y estos dos operadores generales son tan intuitivos y naturales que sirven para satisfacer muchas de las necesidades de información del usuario, como vamos a ver a continuación.

La forma básica de una consulta en el lenguaje EC será la siguiente:

$$Q = (s_1 \times q_1, s_2 \times q_2, \ldots, s_m \times q_m) \tag{1}$$

en la que la consulta Q está formada por una unión de subconsultas q_i.

Cada subconsulta puede ser

- una expresión de presencia,
- una expresión de inclusión.

Cada subconsulta está ponderada por un coeficiente s_i que le da un peso en la consulta. Así, un coeficiente positivo aumentará la relevancia los documentos que cumplen esa subconsulta, y un coeficiente negativo disminuirá la relevancia de los documentos que se ajustan a esa subconsulta. Por defecto, se puede entender que la ausencia de coeficiente implica un valor del mismo igual a 1.

Además, se puede determinar la obligatoriedad o no de la presencia de todos los elementos de una consulta y en el orden en que se especifican, según se explica más adelante.

Cada documento recibirá una puntuación relacionada con su relevancia respecto a la consulta $Rel(d, Q)$ que estará compuesta por un par de valores. Estos valores reflejarán el grado de relación del documento con las subconsultas con coeficientes positivos y negativos, respectivamente. Esta información se utilizará para presentar al usuario una relación de documentos ordenados por su relevancia con respecto a la consulta.

Este proceso se puede ver en la siguiente expresión:

$$\begin{aligned} Rel(d_j, Q) &= (x, y) \\ x &= \sum_{i=1}^{m} \nu(d_j, q_i) s_i \ \forall s_i > 0 \\ y &= \sum_{i=1}^{m} \nu(d_j, q_i) |s_i| \ \forall s_i < 0 \end{aligned} \tag{2}$$

donde la función $\nu(d_j, q_i)$ calcula el número de veces que la subconsulta q_i aparece en el documento d_j. De hecho, la función ν es el único enlace entre el sistema de indexación y el lenguaje de consulta EC, y resulta una interfaz clara y bien definida entre ambos. El sistema de indexación debe ser capaz de resolver los dos tipos de operadores que componen las consultas EC, considerando los diferentes tipos de información de los documentos. El modo en que se trate la información en el índice y otros aspectos relacionados con la indexación no deben afectar al lenguaje considerado ni al modo de resolver las consultas. Podría darse el caso de que varios sistemas de indexación diferentes colaboraran para resolver las consultas planteadas, de modo que cada uno estaría especializado en indexar y responder preguntas sobre un tipo de datos específico (texto, imágen, sonido, estructura, etc), e incluso que no existiera índice y la búsquedas se resolvieran *on line*. En nuestro sistema hemos utilizado ficheros invertidos [6] para construir el índice.

Una vez explicada la lógica de consulta y la forma de evaluación de las consultas, vamos a definir y explicar los dos operadores generales que se van a utilizar: *presencia* e *inclusión*.

3.2.1 El Operador Presencia

Una expresión de presencia tiene la forma general siguiente

$$(s_1 \times e_1, s_2 \times e_2, \ldots, s_n \times e_n) \tag{3}$$

en el que cada elemento de presencia e_i está modificado por un coeficiente s_i que determina su peso en la expresión de presencia, positivo o negativo.

Un elemento de presencia puede ser cualquiera de los siguientes:

- un elemento de contenido, c, con $c \in C$.
- un elemento de estructura, e, con $e \in E$.

Así, un ejemplo de expresión de presencia puede ser `("a", -"b", -"c"))` que está preguntado por los documentos que contienen el elemento `a`, y no tienen `b` o `c`. Si quisiéramos penalizar la aparición de la la secuencia compuesta por el elemento `b` seguido por el elemento `c` la expresión quedaría `("a", -["b","c"])`.

Las operaciones de presencia se evalúan siguiendo la misma regla que en las expresiones EC generales.

3.2.2 El Operador Inclusión

Este operador añade al lenguaje la posibilidad de utilizar la estructura del documento como un factor que ayude a la localización de documentos relevantes al usuario, ya que es el que permite expresar relaciones de inclusión entre elementos de contenido y estructura. Estas relaciones no se podían expresar con los anteriores operadores, ya que hacían mención sólo a la presencia o ausencia de elementos en el documento, pero no a la relación jerárquica de inclusión entre ellos.

Una expresión de inclusión tiene la forma general siguiente

$$(e_0(s_1 \times e_1, s_2 \times e_2, \ldots, s_n \times e_n)) \tag{4}$$

con la secuencia de elementos de inclusión y coeficientes habitual, en la que $e_0 \in E$ y e_i *si* $0 < i \leq n$ son elementos de inclusión. Con ella se expresa que el elemento

de estructura e_0 contiene los elementos de inclusión e_i, cada uno multiplicado por el coeficiente s_i, a efectos de calcular la relevancia del documento considerado.

Un elemento de inclusión puede ser cualquiera de los siguientes:

- un elemento de contenido, c, con $c \in C$.
- un elemento de estructura, e, con $e \in E$.
- una expresión de inclusión, con lo que se puede tener anidamiento en las consultas de inclusión.

Un ejemplo de consulta de inclusión puede ser la siguiente, `("a"("b","c"))` en la que se pregunta por los documentos con `a` conteniendo `b` o `c`.

3.3 Otra Forma de Lenguaje Booleano

El modo de formar la consulta y de evaluarla da una idea del modo de trabajo que se ha elegido y de la lógica que se utiliza en la recuperación. La idea fundamental es evitar la presencia de operadores booleanos que dificulte la utilización de este lenguaje por parte de usuarios no experimentados. Aunque como las expresiones booleanas están ampliamente difundidas como base de los lenguajes de consulta, y muchos usuarios están familiarizados con ellas, merece la pena tener en cuenta su incorporación en nuestro lenguaje de consulta, aunque no sea de modo explícito. El principal problema que plantea el uso de operadores booleanos `AND` y `OR` es que su significado no se corresponde exactamente con el significado de la conjunción *y* y la disyunción *o*, del lenguaje común. Esto supone una fuente de malentendidos y errores. También el operador `NOT` puede generar errores cuando nos encontramos con una doble negación –igual a una afirmación–, cuyo significado lógico no siempre se corresponde con el significado en los lenguajes comunes.

Por todo ello se ha tomado la decisión de eliminar los operadores booleanos del lenguaje, pero manteniendo su sentido. Esto se puede ver en la forma general de las consultas EC, y en el modo de evaluar las mismas. Así, una consulta se puede ver como una serie de elementos delimitada por paréntesis o corchetes que determinan la obligatoriedad de su presencia o no, y separados por una coma. Por lo tanto, una consulta puede ser interpretada como una serie de elementos unidos o por el operador booleano `AND`, cuando estamos considerando obligatoriedad, o unidos por el operador booleano `OR` cuando no se considera obligatoriedad; ambos representado por la coma ','. Así, en una consulta se expresan el conjunto de condiciones que hacen relevante a un documento enumerándolas y asociando un grado de importancia a cada una de ellas, el coeficiente. Además, este coeficiente es una representación del operador `NOT`, cuando es negativo.

3.4 El lenguaje de Consulta Visual

Se ha comentado anteriormente que un lenguaje de consulta que permita considerar la estructura y el contenido de los documentos debe poder expresarse de manera visual para facilitar su uso. A continuación se van a describir las caraterísticas que debería tener ese entorno visual de consulta así como presentar un primer prototipo de lenguaje de consulta visual.

Lo primero de todo es diseñar el entorno visual del sistema de recuperación. En él deben encontrarse las siguientes zonas:

- Una zona en la que se exprese la consulta. Esta debe ser como un tablero de dibujo, en el que se van colocando los distintos elementos que conforman la consulta.
- Una zona de elementos. En esta zona se muestran los distintos elementos que pueden elegirse para componer las consultas. En ella se han de mostrar tanto los elementos de contenido como de estructura, además de las relaciones jerárquicas de contención entre ellos.
- Una zona de resultados, en los que se pueda ver diferentes representaciones de los documentos recuperados. Ya que una forma de visualización no siempre es suficiente, debe proveerse de otras formas de ver los resultados [2].

Figura 1: Elemento de consulta.

En lo que respecta al lenguaje de consulta visual, cada uno de los elementos, ya sean de contenido o de estructura va a ser representado por una caja con un título que corresponde al identificador del elemento de contenido o de estructura y con una barra desplazadora en un lado, que permita especificar el coeficiente que afecta a dicho elemento. Un ejemplo de esto se puede observar en la Figura 1, en el que se ve una caja de texto en la que se ha escrito la palabra `hola`, afectada por un coeficiente igual a 1.

Dentro de este marco de trabajo, se han elegido dos formalismos visuales para representar las operaciones de presencia y de inclusión de que está compuesto el lenguaje desarrollado:

- La operación de presencia se expresa mediante la colocación de un elemento en la zona de consulta.
- La operación de inclusión se expresará colocando la caja del elemento contenido dentro de la caja del elemento que lo contiene.

Estos dos formalismos visuales son intuitivos y reflejan de una manera clara las consultas.

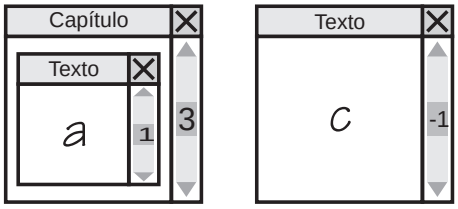

Figura 2: Ejemplo de consulta.

Un ejemplo de las consultas se pueden ver en la figura 2, en la que se expresa la consulta `(3*cap("a"),-1*"c")`, que pregunta por los documentos que contienen `a` dentro de un capítulo, y no tienen `c`.

4 Un Ejemplo de Consulta

A continuación se presenta un ejemplo de consulta utilizando nuestro lenguaje de consulta en estructura y contenido, utilizando para ello un conjunto de documentos HTML. En lugar de utilizar la versión visual del lenguaje vamos a utilizar la versión texto del mismo por cuestiones de espacio.

El conjunto de documentos sobre el que va a efectuar las consultas está compuesto por tres documentos de ejemplo, que incluyen como elementos de estructura los asociados a las etiquetas `<HTML>`, `<HEAD>`, `<TITLE>`, `<BODY>` y `<H1>`, y cadenas de caracteres como elementos de contenido. Así, según nuestra notación $E = \{\texttt{doc}, \texttt{cab}, \texttt{tit}, \texttt{cue}, \texttt{cap}\}$ y $C = \{\texttt{texto}\}$, y la correspondencia entre elementos de estructura y las etiquetas HTML correspondientes será la siguiente:

doc	<HTML> </HTML>
cab	<HEAD> </HEAD>
tit	<TITLE> </TITLE>
cue	<BODY> </BODY>
cap	<H1> </H1>

Los documentos que componen la colección seleccionada se pueden ver en la Tabla 1. Estos sirven sólo para mostrar el tipo de documentos con los que se puede trabajar y no tienen ningún significado más.

<HTML> <HEAD> <TITLE> a </TITLE> </HEAD> <BODY> <H1> a </H1> a b c </BODY> </HTML>	<HTML> <HEAD> <TITLE> b </TITLE> </HEAD> <BODY> <H1> a </H1> b c a </BODY> </HTML>	<HTML> <HEAD> <TITLE> c </TITLE> </HEAD> <BODY> c b a </BODY> </HTML>
doc 0	doc 1	doc 2

Tabla 1: Colección de Documentos.

Sobre esta colección de documentos hemos planteado una serie de consultas dirigidas a localizar los documentos que traten de 'a', y no traten sobre 'c'. Todas estas consultas, así como los resultados obtenidos se pueden ver en la Tabla 2.

num	consulta	resultados		
		doc 0	doc 1	doc 2
1	`(1*"a",-1*"c")`	(3,1)	(1,1)	(1,2)
2	`(3*tit("a"),2*cap("a"),1*"a",-1*"c")`	(8,1)	(4,1)	(1,2)
3	`(["a","b"],-1*"c")`	(3,1)	(1,1)	(0,2)
4	`(doc(cap))`	(1,0)	(1,0)	(0,0)

Tabla 2: Consultas y sus Resultados.

Estos resultados permiten observar las características de nuestro lenguaje de consulta. Así, en la consulta 1 se preguntan sobre los documentos que contienen `a`, con un coeficiente igual a `1`, y por los que contienen `c` multiplicados por un coeficiente igual a `-1`. El resultado muestra como los documentos adquieren distintas puntuaciones conforme a su relevancia con respecto a la consulta, pero no se distinguen mucho unos de otros. Así, la consulta 2 añade elementos de estructura a la misma, y considera que los documentos que contengan

`a` en el título de un capítulo del documento (`cap`) deben ser más relevantes que los que tengan `a` en un párrafo cualquiera, y más aún si `a` aparece en el título del documento (`tit`). Esto es el origen de las puntuaciones `3, 2 y 1` de la consulta. El resultado muestra que los documentos se separan entre sí y el documento 0 resulta como el más relevante, seguido del 1 y por último el 2; lo que permite pensar en la utilidad de incluir la estructura en la consulta.

La consulta 3 ilustra la posibilidad de incluir secuencias de elementos en nuestras consultas. Así, pregunta por los documentos que contienen `a` seguido de `b`, en este orden, y no contienen `c`.

La consulta 4 muestra una posibilidad más de nuestro lenguaje, la consulta sobre elementos de estructura. Así, se pregunta sobre los documentos que tienen un capítulo en ellos. Esta posibilidad no se puede encontrar en los sistemas de consulta tradicionales, ya que no consideran la estructura de los documentos.

5 Conclusiones y Trabajo Futuro

En este trabajo se ha descrito un lenguaje de recuperación, que hemos diseñado e implementado, que permite consultar tanto por el contenido como por la estructura de los documentos. Para obtener este lenguaje se ha considerado como objetivo un acercamiento mas próximo al punto de vista del usuario que al sistema de indexación. También se ha propuesto una versión visual del lenguaje de consulta.

Actualmente se continúa trabajando para la obtención de un prototipo del modelo basado en el descrito en este trabajo. En particular, se pretenden estudiar las posibilidades de nuestro lenguaje visual, planeando la realización de pruebas que demuestren su usabilidad.

Otra cuestión interesante podría ser la comparación de la expresividad del lenguaje presentado con respecto a otros propuestos con anterioridad [3]. En particular, nosotros nos hemos limitado a una sóla jerarquía de elementos de estructura sin solapamiento, aunque esto no parece un problema pues la mayoría de los documentos tienen una sóla jerarquía de estructura sin solapamientos (por ejemplo SGML [7]). Nuestro lenguaje no permite relaciones directas de inclusión, esto es, buscar las relaciones padre–hijo, sino que todas las relaciones de inclusión son transitivas. En la mayoría de los casos, esta limitación no es importante, y los lenguajes que no presentan esta restricción son mucho más complejos, y necesitan índices especiales que no siempre son eficientes [3].

Finalmente, la ordenación de los resultados puede ser mejorada, quedando pendiente el estudio del modo de establecer la relevancia de los documentos considerando la estructura de los mismos, la realimentación de la relevancia de las consultas, y la expansión de las mismas.

Referencias

[1] Baeza-Yates, R. An hybrid query model for full text retrieval systems. Tech. rep., Department of Computer Science, University of Chile, 1994.

[2] Baeza-Yates, R. Visualization of large answers in text databases. In *Advanced Visual Interfaces, AVI'96* (May 1996).

[3] Baeza-Yates, R., and Navarro, G. Integrating contents and structure in text retrieval. In *ACM SIGMOD Record* (Mar. 1996), vol. 25, ACM press, pp. 67–79.

[4] Clarke, C., Cormack, G., and Burkowski, F. An algebra for structured text search and a framework for its implementation. *The Computer Journal* (1995).

[5] Consens, M. P., and Milo, T. Algebras for querying text regions. Tech. rep., Department of Computer Science, University of Waterloo, 1995.

[6] Frakes, W. B., and Baeza-Yates, R., Eds. *Information Retrieval: data structures and algorithms*. Prentice Hall, 1992.

[7] International Standards Organization. *Information Processing – Text and Office Systems– Standard Generalized Markup Language (SGML)*, 1986. ISO 8879–1986.

[8] Kilpeläinen, P., and Mannila, H. Retrieval from hierarchical texts by partial patterns. In *ACM SIGIR'93* (June 1993), ACM press, pp. 214–222.

[9] Kuikka, E., and Salminen, A. Two-dimensional filters for structured text. *Information Processing and Management* (1995).

[10] Loeffen, A. Text databases: A survey of text models and systems. In *ACM SIGMOD* (Mar. 1994), ACM press.

[11] MacLeod, I. A query language for retrieving information from hierarchic text structures. *The Computer Journal* (1991), 254–264.

[12] Mendelzon, A. O., Mihaila, G. A., and Milo, T. Algebras for querying text regions. Tech. rep., Department of Computer Science, University of Totonto, Mar. 1996.

[13] Navarro, G., and Baeza-Yates, R. Proximal nodes: a model to query document databases by content and structure. *ACM TOIS 15*, 4 (Oct. 1997), 401–435.

[14] Sacks-Davis, R., Arnold-Moore, T., and Zobel, J. Database systems for structured documents. In *ADTI'94* (1994), pp. 272–283.

[15] Salminen, A., and Tompa, F. Pat expressions: an algebra for text search. In *COMPLEX'92* (1992), pp. 309–332.

[16] Salton, G., and McGill, M. J. *Introduction to Modern Information Retrieval*. Computer Science Series. McGraw-Hill, 1983.