Un Sistema de Recuperación de Información Estructurada

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

Resumen

En este trabajo se presenta un lenguaje de consulta sobre estructura y contenido, EC, que trabaja con documentos XML y toma los datos de la estructura de la misma DTD XML. El lenguaje EC está orientado a la facilidad de uso e incluye sólo dos operadores: presencia e inclusión y no toma una forma booleana, sino que utiliza pesos asociados a cada término de la consulta. También se propone una medida de la relevancia de cada documento con respecto a una consulta. Todo esto utilizando sólo ficheros invertidos como índice tanto para los elementos de contenido como para los de estructura.

Palabras Clave: Búsqueda en Documentos Estructurados, Bases Documentales, XML, Sistemas de Recuperación de Información.

Abstract

This work show a query language wich considers the structure and the content of XML documents, called EC. The EC language is user friendly and includes two operators: presence and inclusion, and is not a boolean language but used aweight with each query element. Also, this ork presents a new form of document relevance. All this is done using inverted files to index the structure and content elements.

Keywords: Searching on Structured Documents, Bases Documentales, XML, Information Retrieval Systems.

1 Introducción

Cada vez toma más importancia el intercambio y manejo de documentos estructurados según alguno de los estándares al uso (SGML, HTML, XML, etc). El interés en los documentos estructurados y en los lenguajes de formateo de documentos reside en gran medida en el uso de documentos hipermedia, que necesitan estructurar los documentos en partes que delimiten las distintas informaciones que los componen, y que identifiquen la naturaleza de cada una de ellas. Dado el esfuerzo necesario para confeccionar un documento, y más si es hipermedia, hay que hacer algo para que ese esfuerzo perdure en el tiempo lo más posible. Así, si se utiliza un lenguaje de formateo de documentos estándar es posible que el documento pueda ser tratado por más herramientas, indexadores, visualizadores, etc., durante más tiempo. Además, siempre puede existir la posibilidad de traducir un documento que utilice un formato estándar que se haya quedado obsoleto a otro más moderno.

Cuando hablamos de documentos con estructura nos estamos refiriendo a documentos cuya estructura es declarada explícitamente de algún modo, ya sea asociando etiquetas a elementos de estructura o mediante la sintaxis con la que se escribe el documento, como hacen los lenguajes de programación. No se pueden entender como documentos estructurados aquellos escritos utilizando cualquier procesador de textos, ya que en ellos la estructura (por ejemplo el título de un capítulo) se denota a través de la forma que adopta el contenido.

De la misma manera que sucede en los sistemas de recuperación de información clásicos, al considerar los documentos estructurados surge la necesidad de buscar en ellos, pero no sólo por su contenido, sino también por su estructura. A esto podríamos llamarlo Recuperación de Información Estructurada, Structured Information Retrieval, SIR.

En este trabajo vamos a presentar un modo de indización y un lenguaje consulta para documentos estructurados, que se caracteriza por utilizar un solo índice tanto para la estructura como para el contenido de los documentos en el que el lenguaje de consulta no es de tipo booleano y los documentos son XML.

2 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, vídeo, 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 construidos sobre elementos de contenido C y de estructura de E.

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.

Para diferenciar los elementos de estructura del contenido del documento y determinar las características de la estructura del mismo (jerarquía, inclusión, identificadores, etc.) se utiliza la definición de la metaestructura del documento. Para ello nosotros utilizamos las Document Type Definition, DTD, del Extensible Markup Language, XML [3]. Así, la estructura de los documentos estará definida por una DTD.

3 El Lenguaje de Consulta

Este lenguaje de consulta cumplirá con los siguientes objetivos:

- Permitir consultar sobre estructura y contenido.
- Resultar más cercano al usuario que al sistema de recuperación.

En el trabajo [1] se presentan varios de los más importantes lenguajes de consulta que permiten la consideración de la estructura de los documentos. Casi todos ellos incluyen una gran cantidad de operadores que tratan los elementos de estructura y contenido de los documentos recogidos en la base, pero analizando los operadores hemos llegado a la conclusión que se pueden clasificar en dos clases de consultas: *incluido* y *presente*, tomando formas más o menos sofisticadas en cada uno de los sistemas estudiados. Así, estas dos operaciones son las que hemos considerado en nuestro sistema.

El segundo aspecto que hacen que los sistemas IR tradicionales no sean fáciles de utilizar por usuarios no expertos es el uso de los operadores lógicos AND, OR y NOT para construir las consultas [5]. Por ello, nosotros hemos basado nuestro sistema de consulta en la utilización de pesos asociados a los elementos incluidos en las consultas.

3.1 Las Consultas EC

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

$$Q = (s_1 \; x \; q_1, \; s_2 \; x \; q_2, \ldots, \; s_m \; x \; q_m)$$

en la que la consulta *Q* está formada por una unión de subconsultas q_i.
Cada subconsulta puede ser

1. una expresión de presencia,
2. 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.

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

3.1.1 El Operador Presencia

Una expresión de presencia tiene la forma general siguiente

$$(s_1 \; x \; e_1, \; s_2 \; x \; e_2, \ldots, \; s_n \; x \; e_n)$$

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 y valdrá 1 por defecto.
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 $e \in E$.

Las expresiones de presencia no tienen, en principio, asociado ningún orden ni obligatoriedad. Si se quiere incluir el orden y la obligatoriedad de la presencia de los elementos en una expresión de presencia, debemos transformarla en una expresión de secuencia de elementos utilizando los corchetes como delimitadores, en lugar de los paréntesis.

Así, un ejemplo de expresión de presencia puede ser `(1*"a", -1*"b", -1*"c")` que está preguntado por los documentos que contienen el elemento a, y no tienen b o c.

3.1.2 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.

Una expresión de inclusión tiene la forma general siguiente

$$(e_0(s_1 \times e_1, s_2 \times e_2, \ldots, s_n \times e_n))$$

con la secuencia de elementos de inclusión y coeficientes habitual, en la que $e_0 \in E$ y e_i con $0 < i \leq n$

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.

Con una expresión de inclusión se consulta sobre los documentos en los 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 ejemplo de consulta de inclusión puede ser la siguiente, `(S["b","c"])` en la que se pregunta por los documentos conteniendo b seguido de c, ambos dentro de S.

3.2 Cálculo de la Relevancia

Cada documento d 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:

$$Rel(d_j,Q) = (x, y)$$

$$x = \sum_{i=1}^{m} \nu(d_j, q_i).s_i \quad \forall s_i > 0$$

$$y = -\sum_{i=1}^{m} \nu(d_j, q_i).s_i \quad \forall s_i < 0$$

donde la función $\nu\ (d_j,q_i)$ calcula el número de veces que un documento d_j responde a la subconsulta q_i.

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. En nuestro sistema hemos utilizado ficheros invertidos [4] para construir el índice.

3.2 Otra Forma de Lenguaje Booleano

El principal problema que plantea el uso de los 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 que conducen al desánimo de los usuarios no experimentados. 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. Una consulta EC 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 "," . Además, los coeficientes son una representación del operador `NOT`, cuando son negativos.

La aparente falta de expresividad de nuestro lenguaje se ve compensada por la utilización de diferentes funciones de cálculo de la relevancia `Rel(d_j,Q)`, y será posible experimentar con diferentes funciones de representación de `Rel()` para satisfacer diferentes necesidades de evaluación de la relevancia de los documentos con respecto a una consulta.

Este es uno de los aspectos a destacar en este trabajo, ya que se hace independiente la representación de la función de relevancia del sistema de indexación y recuperación.

4 Indización y Recuperación

En esta sección vamos a describir y explicar los algoritmos y las estructuras de datos asociadas que hemos utilizado para construir tanto el índice como el motor de búsqueda.

De entre las características de nuestro sistema de indexación y búsqueda se pueden destacar las siguientes:

- Se utilizan índices convencionales, ficheros invertidos.
- El índice es único tanto para el contenido de los documentos como para la estructura.
- Se representan las consultas como árboles.

El hecho de que sólo se traten documentos XML no se ha de considerar como una restricción en cuanto al modo de expresar la estructura de los documentos, ya que se pueden tratar documentos cuya estructura se exprese de otro modo, previo paso por un filtro que los etiquete convenientemente.

4.1 Un Solo Índice para Estructura y Contenido

En la literatura se pueden encontrar descripciones de sistemas de indexación de documentos con estructura en la que se mantienen dos tipos de índices distintos, uno para los elementos de contenido, palabras, y otro para cada uno de los elementos que conforman la estructura de los documentos. Esto, se vuelve impracticable en cuanto la estructura de los documentos se complica y el número de elementos de estructura a indexar crece y se diversifica. Además en la mayoría de los casos, aunque se mantenga un solo índice para todos los elementos de estructura de los documentos, éste puede ser de naturaleza totalmente distinta que el utilizado para el contenido de los mismos, lo que complica su uso y mantenimiento.

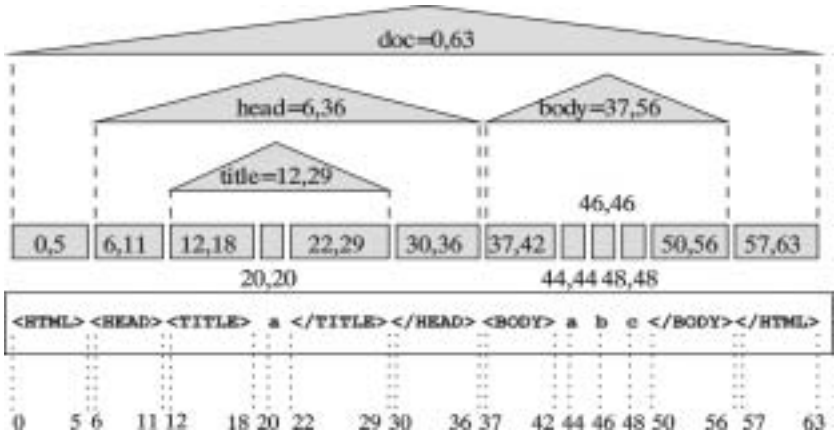

Figura 1: Relaciones de jerarquía entre segmentos en un documento.

Por todo ello, nosotros, a la hora de plantearnos el tipo de índice y su arquitectura, nos decidimos por un índice único para elementos de contenido y de estructura, y que fuera un tipo de índice clásico y bien conocido: los ficheros invertidos. De este modo podremos asentarnos en sistemas que ya estuvieran en funcionamiento y aprovechar su índice, con pocas o ninguna modificaciones.

4.2 El Modelo de Segmentos

Para nosotros, los documentos están constituidos por segmentos, cadenas de caracteres con significado que tienen una posición de inicio y otra de finalización. Un segmento se va a corresponder tanto con una palabra de contenido como con una etiqueta de estructura.

Sólo vamos a considerar una jerarquía de estructura. De modo que un elemento de estructura o está contenido en otro, (que será de rango superior) o es disjunto con los demás.

En la Figura 1 se puede ver un ejemplo concreto de la jerarquía de segmentos en un documento y de cómo se puede utilizar el modelo de segmentos para representar también las relaciones de jerarquía entre los elementos de estructura y de contenido. Se puede observar como los segmentos correspondientes a las hojas, corresponden con las palabras del documento, mientras que cuando vamos ascendiendo hacia la raíz se van considerando nuevos segmentos, que agrupan a segmentos de nivel inferior, hasta considerar la raíz, un único segmento $S_{0,63}$ que abarca todo el documento.

4.3 La Indexación

En el fichero invertido se almacenan las cadenas asociadas a los segmentos, el documento en que aparece y su posición de inicio y final en cada documento. Como segmentos vamos a considerar las palabras en lo que respecta a los elementos de contenido, y las etiquetas de marcado de los elementos de estructura (tanto las de apertura como las de cierre). Con esto se determina que nuestra unidad de indexación y recuperación será la palabra, en lo que respecta a los elementos de contenido, y la porción de documento incluida entre las etiquetas de apertura y cierre, en los elementos de estructura.

4.4 Representación y Resolución de Consultas

Dada la naturaleza de las consultas EC el modo utilizado para representar las consultas EC ha sido un árbol n-ario, en el que la raíz representa la consulta EC, y los nodos del primer nivel a cada una de las subconsultas.

Los nodos hoja serán siempre elementos EC, pudiendo corresponder los nodos interiores tanto a expresiones de inclusión o de presencia como a elementos EC.

Una vez que el árbol con la consulta ha sido construido, se recorre en postorden, resolviendo cada uno de los nodos hijos antes que el padre. De este modo se obtendrán los segmentos asociados a los hijos para poder componer a partir de ellos los segmentos que corresponderán al padre.

Un ejemplo de árbol de consulta en proceso de resolución se puede ver en la Figura 2.

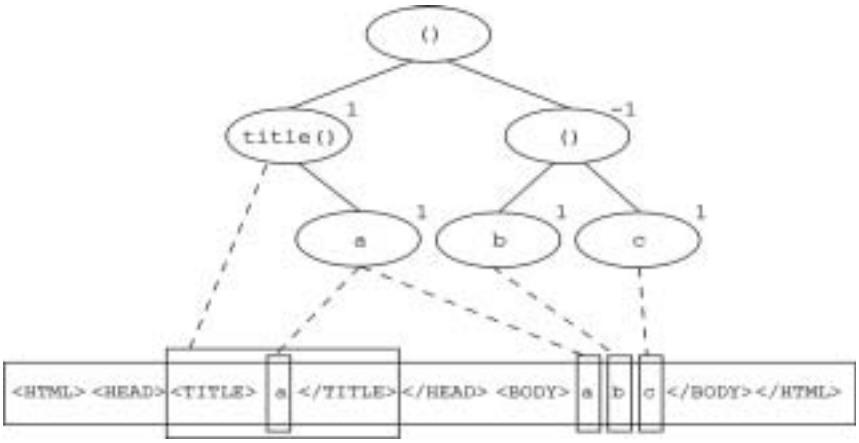

Figura 2: Resolución de una consulta EC.

Resolver un nodo significa que obtenemos un conjunto ordenado de segmentos correspondientes a partes de los documentos que son relevantes a la consulta contenida por ese nodo. Estos segmentos tendrán asociado un peso, que es el mismo que modifica a la consulta asociada a ese nodo.

Los nodos se resuelven de diferente modo según se correspondan con elementos de contenido, de estructura, o sean expresiones de inclusión o de presencia (conjunto o secuencia).

Una de las simplificaciones realizadas en esta fase es la de imponer que los documentos se encuentren bien etiquetados. Asunto este resuelto al tratar con documentos XML.

Es en esta etapa, donde se pueden implementar muchos otros tipos de búsquedas. Como por ejemplo, búsquedas aproximadas, búsquedas por raíces, búsquedas en proximidad, etc.

Al final del proceso, cada documento obtendrá una puntuación consistente en un par de valores que serán la suma de las puntuaciones obtenidas en las subconsultas con peso positivo y negativo respectivamente.

5 Conclusiones y Trabajo Futuro

Hemos discutido las propiedades que deberían tener los lenguajes de consulta con estructura con el fin de resultar útiles a los usuarios. A partir de ellas hemos definido el lenguaje de consulta EC sobre estructura y contenido, así como las implicaciones sobre el modelo de documento a considerar y se ha propuesto una nueva medida de la relevancia de un documento. Todo ello utilizando XML para definir los documentos de la base. También se han descrito tanto los algoritmos como las estructuras de datos asociadas al lenguaje EC. Entre las principales características del mismo se pueden destacar que considera los documentos según el modelo de segmentos e implementa el índice de manera unificada tanto para los elementos de consulta como para los elementos de contenido.

Analizando otros sistemas o lenguajes de consulta con estructura, se nos antoja interesante la ampliación de las posibilidades de consulta de nuestro lenguaje EC para incluir operadores como la búsqueda con distancia, búsqueda aproximadas, y otros. Esto puede aumentar la potencia de nuestro lenguaje. Además, será interesante poder distinguir entre las relaciones *incluyendo* e *incluido*, ya que en la versión actual del lenguaje EC se transforman en la relación *incluyendo*.

Estamos trabajando en la implementación de una versión visual de nuestro lenguaje de búsqueda, alguna de las claves de su diseño se pueden ver en [2].

También queda pendiente el análisis de la influencia de nuestro lenguaje de consulta EC en la precisión y la recuperación sobre bases documentales sobre las que se puedan tener juicios de relevancia. Otro problema no resuelto y cuya solución no parece trivial desde este estado de desarrollo del tema es el tratamiento de la realimentación de la relevancia cuando las consultas incluyen elementos de estructura. Esto supondría una extensión de los modelos vectoriales y probabilísticos [4] cuyas consecuencias no nos atrevemos a adelantar desde aquí.

Referencias

[1] 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.

[2] Baeza-Yates, R. Vegas, J., Navarro, G. and Fuente, P. de la *A model and a visual query language for structured text*. In Proceedings of SPIRE´98 (Sept 1998), String Processing and Information Retrieval: a South American Symposium, IEEE Computer Society, pp7-13.

[3] Bray, T., Paoli, J. et all. *Extensible Markup Language (XML)* 1.0 Tech. Report, WWW Consortium (W3C), 1998. http://www.w3.org/TR/1998/REC-xml-19980210.

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

[5]Pollock, A., and Hockely, A. *What´s wrong with Internet searching*. D-LIB Magazine (Mar. 1997).