Bibliotecas digitales con documentos comprimidos: una arquitectura*

Joaquín Adiego[1], Pablo de la Fuente[1], Jesús Vegas[1], Miguel Villarroel[1], and Alberto Pedrero[2]

Grupo de Recuperación de Información y Bibliotecas Digitales (GRINBD)
[1]Departamento de Informática. Universidad de Valladolid, España
[2]Escuela Universitaria de Informática. Universidad Pontificia de Salamanca, España
[1]`[jadiego,pfuente,jvegas,miguelv]@infor.uva.es`
[2]`apedrero@upsa.es`

Resumen Las bibliotecas digitales deben almacenar de forma eficiente grandes cantidades de información y además deben disponer de mecanismos que permitan realizar búsquedas y recuperar información, a veces multimedia, en el menor tiempo posible. Aunque en la actualidad el coste del almacenamiento no se considera un problema, ciertos entornos de ejecución (CD-ROMs) imponen restricciones que se deben cumplir utilizando técnicas de compresión apropiadas. La realidad nos muestra que estos dos requisitos están en conflicto: la necesidad de realizar una descompresión incrementa el tiempo de acceso y la necesidad de disponer de un índice aumenta el espacio necesario.
Tradicionalmente, las funcionalidades de recuperación que ofrecen las bibliotecas digitales están orientadas a sus contenidos, pero en ciertos entornos —como es el caso de los sistemas jurídicos de información— también es necesario que estas funciones se puedan aplicar a la estructura de esa información. Este artículo propone una arquitectura de tres capas para una biblioteca digital manteniendo comprimidos todos los datos de los que consta y obteniendo buenos tiempos de respuesta ante búsquedas por contenido y por estructura.

Palabras clave: Recuperación de Información, Documentos Semiestructurados, Compresión de Datos.

1. Introducción

Muchas bibliotecas digitales contienen documentos que tienen una estructura compleja, a pesar de ello, la mayoría de las bibliotecas digitales no hacen uso de dicha estructura. Es habitual que las funciones de búsqueda y recuperación estén orientadas al tratamiento de los datos con contenido plano sin considerar otros aspectos relacionados con esa información (tal y como cuando el usuario busca un conjunto de términos y el sistema devuelve una lista ordenada de

* Trabajo parcialmente subvencionado por los pryectos CICYT (TEL 1999-0335-C04-04) y CYTED VII.19 RIBIDI

documentos). En algunos contextos el usuario necesita más funcionalidades de búsqueda como, por ejemplo, en las bibliotecas digitales jurídicas, puede ser muy útil proporcionar a los usuarios la posibilidad de incluir restricciones de estructura en sus consultas [HMQS96]. El dominio de la información jurídica se compone de leyes, reglas, códigos de leyes, ect., cada uno con una estructura particular. La necesidad de manipular estructura está presente en el contexto jurídico y en otros contextos de bibliotecas digitales [BVNF98].

La inclusión de la estructura de un documento en los procesos de indexación y recuperación afecta de varias maneras al diseño e implementación de una biblioteca digital. En primer lugar, el proceso de indexación debe de considerar la estructura de forma adecuada para permitir al usuario realizar búsquedas por contenido o estructura. En segundo lugar, el proceso de recuperación debe usar el contenido y la estructura para estimar la relevancia de los documentos. Por último, la interfaz debe permitir al usuario una completa utilización de la estructura del documento [VdC02] ya que la consulta por contenido y estructura sólo se puede llevar a cabo si el usuario es capaz de especificar en la consulta *qué* está buscando y *dónde* se debe encontrar dentro de los documentos. El *qué* hace referencia a la especificación del contenido y el *dónde* hace referencia a la estructura de los documentos.

En líneas generales, las bibliotecas digitales contienen documentos de texto (estructurados) e información multimedia (imágenes, vídeos, sonidos, etc.). Esto implica que el espacio de almacenamiento que se precisa sea considerable y que el simple hecho de acceder a la biblioteca en respuesta a alguna necesidad de información conlleva un tiempo que puede no ser aceptable. En aplicaciones convencionales estos dos problemas se han resuelto mediante la utilización de compresión de datos y estrategias de indexación respectivamente. La compresión de datos reduce la cantidad de espacio en disco necesario, almacenando la información de una manera adecuada y las estrategias de indexación se utilizan en los sistemas de recuperación de información para proporcionar un acceso rápido a los documentos almacenados, de la misma forma que un índice de un libro convencional proporciona un acceso rápido a las páginas del libro en las que aparecen los conceptos mencionados.

2. Trabajo relacionado

Tradicionalmente las bibliotecas digitales tratan los documentos como entidades atómicas, indexándolos y recuperándolos como objetos indivisibles. Las bibliotecas digitales modernas deben de ser capaces de manipular representaciones de documentos más elaboradas, como, por ejemplo, documentos escritos en HTML, SGML o XML. Los nuevos estándares de representación de documentos estructurados obligan a diseñar e implementar modelos y herramientas para trabajar con la estructura de los documentos. Esto implica que los documentos no serán considerados como entidades atómicas sino como un conjunto de objetos agregados e interrelacionados que necesitan ser indexados, recupera-

dos y presentados conjunta o separadamente de acuerdo con las necesidades del usuario.

Existen dos grandes familias de algoritmos de compresión: (1) Los algoritmos sin pérdida en los que la entrada al algorimo de codificación es exáctamente la misma que la salida obtenida por el algoritmo de decodificación. (2) Los algoritmos con pérdida en los que la entrada al algoritmo de codificación no es exactamente igual a la salida decodificada. Generalmente la compresión con pérdida es importante porque al aceptar una pequeña pérdida de información se puede obtener una enorme ganancia en las tasas de compresión. Por este motivo, los algoritmos con pérdida se suelen aplicar para comprimir material multimedia, existendo estándares ampliamente extendidos como, por ejemplo, JPEG, MPEG y MP3 para comprimir imágenes, vídeo y audio respectivamente. En general, para textos se utilizan métodos sin pérdida que tratan al texto como una secuencia de caracteres [Wel84] o como una sucesión intercalada de palabras y separadores [MT97], pero ninguno de ellos tiene en cuenta la estructura de los documentos.

Un método de compresión que tiene en cuenta la estructura de los documentos cuando comprime y descomprime es XMill [LS00] que es un compresor específico para datos XML y agrupa objetos de datos relacionados semánticamente en contenedores. Otro compresor específico para XML es XGrind [TH02] que soporta directamente consultas sobre los documentos comprimidos.

Por otro lado, muchas veces es necesario localizar de forma eficiente una palabra concreta en la colección de documentos. Para ello es habitual indexar la colección de documentos, el método más generalizado y simple son los índices invertidos [BYRN99,WMB99]. Generalmente, un índice invertido se compone de un vector que conteniene todas las palabras distintas de la colección (*vocabulario*) en orden lexicográfico y para cada palabra del índice se almacena una lista con todas las posiciones del texto o documentos en los que aparece la palabra correspondiente (*lista de ocurrencias*).

Navarro y Baeza-Yates proponen en [NBY97] un modelo genérico en el que se utilizan *nodos proximales* para realizar consultas por contenido y estructura, y evitan el problema restringiendo la expresividad del lenguaje de consulta.

En la actualidad existen bibliotecas digitales que mantienen sus datos comprimidos, como es el caso de la biblioteca digital de Nueva Zelanda, NZDL[1], construida con Greenstone. Greenstone es un entorno de desarrollo para construir bibliotecas digitales con compresión que utiliza MG [WMB99] como motor principal para la compresión, búsqueda y recuperación. MG es un software de dominio público, versátil y de propósito general que comprime e indexa textos, documentos escaneados e imágenes, pero no tiene en consideración otros tipos de datos multimedia ni tampoco considera la estructura de los documentos en el momento de la indexación y recuperación.

[1] www.cs.waikato.ac.nz/~nzdl/

3. Descripción de la arquitectura

En esta sección se describe de forma abreviada la arquitectura propuesta para bibliotecas digitales con compresión. La Figura 1 muestra los elementos principales que intervienen en una biblioteca digital con compresión, los detalles relacionados con la recuperación por contenido y estructura principalmente están relacionados con la indexación y nuestra propuesta se expondrá en la siguiente sección. No se hace referencia al proceso de construcción de la biblioteca ya que éste es un proceso independiente a los procesos de recuperación y se puede realizar concurrentemente a los mismos, cuando el bibliotecario lo estime conveniente se pueden bloquear todos los accesos a la biblioteca para efectuar la actualización de los contenidos de la misma.

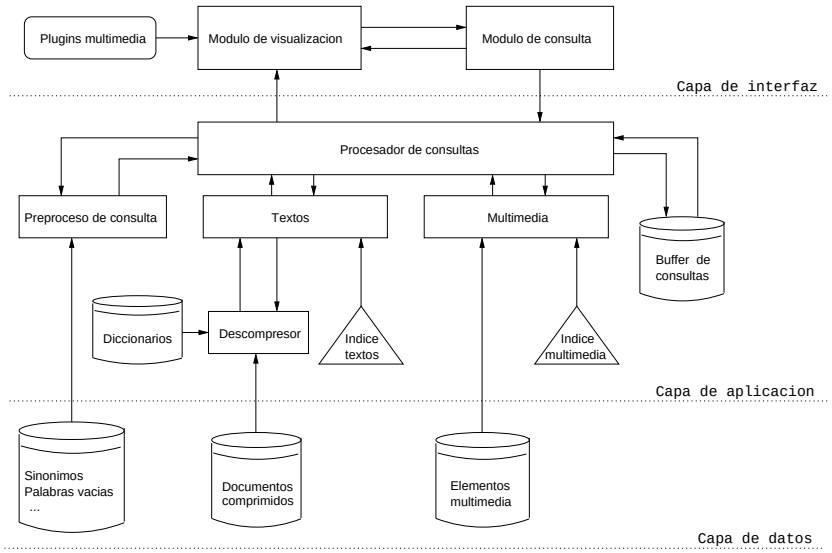

Figura 1. Arquitectura de tres capas para una biblioteca digital con compresión

A continuación se describen las capas y los elementos de los que consta dicha arquitectura:

Capa de datos En esta capa aparecen los repositorios que constienen los datos de los que consta la biblioteca:

- *Elementos multimedia*: en este repositorio se almacenan todos los elementos multimedia de la biblioteca comprimidos utilizando el método de compresión estándar que se considere más conveniente.
- *Documentos comprimidos*: en este repositorio se almacenan los textos de la biblioteca comprimidos mediante algún algoritmo de compresión sin pérdida.

- *Otros*: en este repositorio se almacenan datos adicionales inherentes a la biblioteca y que serán necesarios en el proceso de mejora de la consulta. Por ejemplo, pueden aparecer listas de sinónimos, acrónimos, la lista de palabras vacías, etc.

Capa de aplicación Es la capa principal de la biblioteca, sus funciones principales son la realización de las consultas, obtención de respuestas y descompresión de textos. Fundamentalmente consta de los siguientes elementos:

- *Procesador de consultas*: módulo encargado de recibir las consultas de usuario procedentes de la capa de interfaz y de enviar hacia esa misma capa las respuestas correspondientes a cada consulta ordenadas siguiendo algún criterio de relevancia. El módulo distingue qué tipo de información se debe de recuperar (textos o datos multimedia) y delega las tareas de búsqueda y recuperación a dos módulos especializados:
  - *Textos*: es el encargado de localizar sobre los textos comprimidos los documentos o elementos de estructura que respondan a la consulta, para ello deberá de consultar el *índice de textos* cuya granularidad (palabra, documento, elemento de estructura, etc.) dependerá de las necesidades de recuperación que precise la biblioteca. Este módulo también debe de invocar al descompresor indicándole qué documentos o qué fragmentos de documentos se deben de descomprimir a petición del usuario.
  - *Multimedia*: es el responsable de localizar los elementos multimedia que ha solicitado el usuario o que son necesarios en la consulta de la biblioteca. Para recuperar de una manera eficiente dichos elementos debe de consultar el *índice multimedia*, en el que los elementos multimedia se encuentran organizados de acuerdo a las necesidades de recuperación multimedia de la biblioteca.
- *Descompresor*: este módulo se encarga de descomprimir documentos o fragmentos de documentos, para ello debe de acceder a los documentos comprimidos y a los diccionarios.
- *Preproceso de consulta*: es el responsable de refinar la consulta mediante la eliminación de las palabras vacías que puedan aparecer en la misma, búsqueda se sinónimos o acrónimos, reorganización de los términos de la consulta para optimizar el acceso a los índices, etc.

Capa de interfaz Esta capa se utiliza para interaccionar con el usuario y entre otras cosas debe de proporcionar las funcionalidades necesarias que permitan al usuario realizar consultas por contenido y estructura. En líneas generales, esta capa puede estar formada por dos módulos intercomunicados:

- *Módulo de visualización*: encargado de mostrar los resultados recibidos del procesador de consultas. Estos resultados pueden ser rankings de respuesta, textos o material multimedia. Para reproducir/visualizar este último tipo de información el módulo puede utilizar *plugins multimedia*, aumentando de manera considerable la flexibilidad de la biblioteca.
- *Módulo de consulta*: responsable de obtener, componer y transmitir las consultas de usuario al procesador de consultas.

4. Implementación de un prototipo

A continuación se describe de forma abreviada un prototipo de biblioteca digital construido en base a la arquitectura propuesta en la sección anterior. Las principales características del prototipo son las siguientes:

- Puede trabajar con documentos XML con diferente DTD y además un fichero físico puede contener uno o más documentos lógicos.
- Realiza búsquedas eficientes por contenido y estructura. El prototipo puede resolver consultas del estilo a ¿qué documentos contienen la palabra t en el título? ó ¿qué documentos tienen el elemento de estructura X?
- El proceso de búsqueda se lleva a cabo sin necesidad de descomprimir los ficheros, lo que permite obtener mejores tiempos de respuesta.
- Soporta las características de las lenguas latinas y el almacenamiento y búsqueda de fechas con independencia del formato en el que estén escritas.

4.1. Compresión de los documentos

Los documentos se comprimen utilizando una adaptación propia del modelo de alfabetos separados [BSTW86] e implementa una codificación Huffman basada en palabras ya que este tipo de codificación es sin pérdida, bien conocido, proporciona buenas tasas de compresión [WMB99] y además permite buscar, comparar y descomprimir palabras directamente sin tener que procesar ningún fragmento previo del texto. La codificación utiliza la *palabra* como el elemento básico del diccionario y la secuencia de bits que se utiliza para codificarla se denomina *código léxico*. Podemos considerar que un fichero de texto consiste en la alternancia de cadenas alfanuméricas y de cadenas de puntuación. Una cadena alfanumérica es cualquier secuencia máxima de caracteres alfanuméricos y una cadena de puntuación es cualquier cadena máxima de caracteres no alfanuméricos.

Se han realizado una serie de modificaciones para adaptar la compresión a las características de las lenguas latinas y demás requisitos del sistema. Cuando se aplican las modificaciones siempre se supondrá que sucederá el caso más probable, si no es así se insertará un código de control que indica la excepción. Se han implementado una serie de modificaciones que permiten representar una misma palabra utilizando el mismo código léxico: por lo general una palabra suele aparecer escrita en minúsculas (`alveolo`) pero ocasionalmente puede aparecer escrita completemente en maýusculas (`ALVEOLO`), con alguna letra en mayúscula (`Alveolo`), con alguna letra acentuada (`alvéolo`) o puede suceder cualquier combinación anterior (`AlvÉOlo`); el prototipo detecta estas situaciones y codifica todas las formas de la palabra utilizando el mismo código léxico y, excepto en el caso más habitual, añadirá los códigos de control correspondientes en cada caso. Asimismo, para permitir la búsqueda de fechas con independencia del formato en el que estén escritas se ha extendido el concepto de palabra de manera que englobe las fechas

4.2. Búsqueda por contenido y estructura

El prototipo permite realizar búsquedas eficientes por contenido y estructura. Hecho que permite eliminar la necesidad de procesar partes previas de los textos para comprobar dónde se encuentra un determinado código léxico. Se ha propuesto una nueva forma de indexación que se describe a continuación y que se denomina *índice invertido con direccionamiento a elementos de estructura.*

Esta variante de indexación se apoya en una estructura de control complementaria: la Tabla de Control de Estructura (TCE). La TCE es una matriz en la que cada fila hace referencia a un único elemento de estructura. Utilizaremos la palabra *registro* para hacer referencia a una fila de la TCE. Cada registro contiene un identificador de fichero físico (FID), un identificador del elemento de estructura (SID), las posiciones de inicio y final (en número de bits) de la estructura referenciada en el fichero comprimido, y por último, una autoreferencia (denominada *documento*) que contiene el número de registro del elemento de estructura de mayor jerarquía que contiene el elemento actual. Todos estos datos son importantes al efectuar búsquedas y recuperaciones.

Es necesario expresar la correspondencia entre los ficheros y los elementos de estructura y sus identificadores numéricos. Por ejemplo, supóngase que se dispone de una colección de documentos formada por los ficheros que se muestran en la figura 2. Si se aplican las equivalencias que se muestran en la figura 3, se obtiene la TCE que se ilustra en la figura 4.

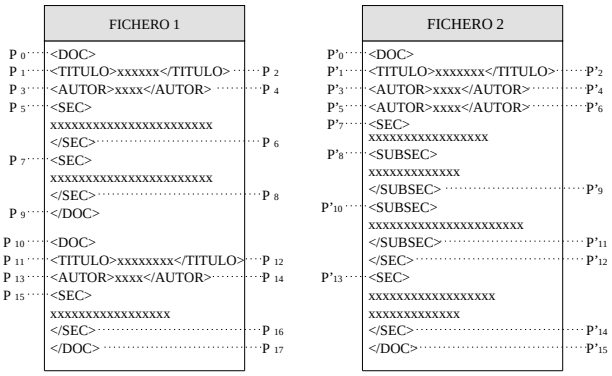

Figura 2. Ficheros de ejemplo

En las listas de ocurrencias del índice se almacena el número de registro de la TCE que se corresponde con el elemento de estructura dónde aparece la palabra, en otras palabras, se almacena la identificación de una determinada estructura en un fichero en concreto. Además, junto al número de registro también se almacena el número de veces que aparece dicha palabra en ese elemento, este dato se utiliza para generar los rankings de la consulta. Si se considera el ejemplo anterior

Fichero	FID
File 1	1
File 2	2

Estructura	SID
DOC	1
TITULO	2
AUTOR	3
SEC	4
SUBSEC	5

Figura 3. Correspondencias

	FID	SID	Inicio	Final	Doc
Reg.1	1	1	P_0	P_9	1
Reg.2	1	2	P_1	P_2	1
Reg.3	1	3	P_3	P_4	1
Reg.4	1	4	P_5	P_6	1
Reg.5	1	4	P_7	P_8	1
Reg.6	1	1	P_{10}	P_{17}	6
Reg.7	1	2	P_{11}	P_{12}	6
Reg.8	1	3	P_{13}	P_{14}	6
Reg.9	1	4	P_{15}	P_{16}	6
Reg.10	2	1	P'_0	P'_{15}	10
Reg.11	2	2	P'_1	P'_2	10
Reg.12	2	3	P'_3	P'_4	10
Reg.13	2	3	P'_5	P'_6	10
Reg.14	2	4	P'_7	P'_{12}	10
Reg.15	2	5	P'_8	P'_9	10
Reg.16	2	5	P'_{10}	P'_{11}	10
Reg.17	2	4	P'_{13}	P'_{14}	10

Figura 4. TCE

y suponiendo que un término ω cualquiera aparece una vez en el título del primer documento del primer fichero, tres veces en la segunda sección de ese mismo documento y cuatro veces en la primera subsección del primer tema del documento que se encuentra almacenado en el segundo fichero, la lista de ocurrencias para ese término se puede expresar como $\omega \rightarrow$ `[ (2,1), (5,3), (15,4) ]`

Cuando se desea buscar una palabra en un elemento de estructura, se recorre secuencialmente la lista de ocurrencias. La TCE también mantiene información implícita sobre la jerarquía de la estructura. Se puede saber qué elementos contienen a otros comparando las posiciones de inicio y final de cada registro. Esto permite resolver consultas del tipo ¿qué documentos tienen el elemento de estructura X?.

5. Evaluación del prototipo

Se han realizado varias pruebas para analizar y evaluar el rendimiento del prototipo. Las pruebas se han realizado en un Pentium 4 a 1.4 Ghz con 128 Mb de memoria RAM y bajo un sistema operativo Linux Red Hat 7.2. Para efectuar los experimentos se han seleccionado diferentes tamaños de colecciones de noticias de la agencia EFE, las características de las mismas se pueden observar en la Tabla 1.

El tiempo necesario para indexar y comprimir las colecciones varía entre los 8.92 segundos para la colección de 1 Mbyte a los 78 minutos para la colección de 400 Mbytes. Como era de suponer, al incrementar el tamaño de la colección se incrementa el tiempo de procesamiento, en parte debido a la relación directa que existe con el número de palabras del vocabulario. Se ha observado que existe una relación casi lineal entre ambos factores ya que al doblar el tamaño de la colección, el tiempo de proceso aproximadamente se dobla.

Las razones de compresión se muestran en la Tabla 1. Dichas razones disminuyen (mejora la compresión) a medida que crece el tamaño de la colección, esto se debe a la relación que existe entre el número de palabras del vocabulario y

Texto		Vocabulario		Vocabulario/Texto		Compresión	
Tamaño	#Palabras	Tamaño	#Palabras	Tamaño	#Palabras	Tamaño	Razón
1067929	228514	146487	16523	13.716 %	7.230 %	495519	46.400 %
5846138	1244741	375680	41629	6.426 %	3.344 %	2323485	39.744 %
10029607	2134130	502864	55553	5.013 %	2.603 %	3830442	38.191 %
19867634	4225847	717565	79021	3.611 %	1.869 %	7353757	37.014 %
39099633	8317080	1026944	113510	2.626 %	1.364 %	14244228	36.431 %
58920064	12531978	1292018	143208	2.192 %	1.142 %	21295209	36.143 %
105264105	22403512	1793323	199975	1.703 %	0.892 %	37754812	35.867 %
209942125	44676521	2642743	295434	1.258 %	0.661 %	74630233	35.548 %
419994481	89691667	3923453	435384	0.934 %	0.485 %	148768608	35.422 %

Tabla 1. Características de las colecciones

Colección	Índice	TCE	Directorio
1067929	900808	76776	19
5846138	4148920	416136	95
10029607	6857536	701736	190
19867634	13070184	1386168	330
39099633	24962592	2710344	590
58920064	37110936	4070640	850
105264105	65280056	7223664	1450
209942125	128276904	14266056	2872
419994481	255023472	28625688	5938

Tabla 2. Tamaños de ficheros

Bytes	Tiempos (cent.)	
Colección	Buscar	Mostrar
1067929	0.034	2.25
5846138	0.410	2.38
10029607	0.638	2.35
19867634	1.310	2.46
39099633	2.853	2.44
58920064	4.206	2.52
105264105	7.286	2.61
209942125	18.796	2.75
419994481	46.898	2.93

Tabla 3. Tiempos de recuperación

el número total de palabras en la colección disminuye a medida que aumenta el tamaño de la colección.

Los tamaños de los ficheros utilizados en el proceso de recuperación se muestran en la Tabla 2. El tamaño de la TCE está en relación con la complejidad de la estructura de los documentos. Los documentos de noticias de la agencia EFE tienen una jerarquía muy simple y hay un gran número de documentos de tamaño pequeño (uno por noticia), esto implica un número elevado de etiquetas estructurales y, en consecuencia, un tamaño grande de la TCE. El fichero de directorio almacena las correspondencias entre los nombres de ficheros y sus identificadores correspondientes y el fichero de estructura —que tiene un tamaño constante de 121 bytes— almacena las correspondencias entre los nombres de las etiquetas y sus identificadores correspondientes.

Las pruebas de recuperación se han llevado a cabo empleando una lista de parada vacía y como consecuencia todos las palabras de la colección se han indexado. Los tiempos de respuesta son los tiempos medios de buscar 500 palabras extraídas aleatoriamente del vocabulario de la colección más pequeña, de esta forma se buscarán siempre las mismas palabras en todas las colecciones. La primera columna de la tabla 3. La primera columna de la tabla muestra el *tiempo medio* necesario para buscar un término y visulizar la lista de resultados. La segunda columna muestra el *tiempo medio* necesario para descomprimir y mostrar un elemento de estructura extraído al azar de la lista de resultados, pudiendo ser el documento completo o una parte de él (por ejemplo el título).

6. Conclusiones y trabajo futuro

Considerando el prototipo y los resultados experimentales, podemos afirmar que se han alcanzado las metas definidas en la introducción de este trabajo. Se ha propuesto una arquitectura que permite implementar de forma eficiente bibliotecas digitales con compresión trabajando con documentos semiestructurados (XML, SGML). En la actualidad se está trabajando con una colección de documentos jurídicos y los resultados obtenidos hasta ahora son gratificantes.

En un futuro próximo, el objetivo es plantear nuevos modelos de compresión considerando nuevos modelos de compresión, cuyo coste y prestaciones se pretende evaluar.

Referencias

[BSTW86] J. Bentley, D. Sleator, R. Tarjan, and V. Wei. A locally adaptive data compression scheme. *Communications of the ACM*, 29:320–330, 1986.

[BVNF98] Ricardo Baeza-Yates, Jesús Vegas, Gonzalo Navarro, and Pablo de la Fuente. A model of visual query language for structured text. In *Proceedings of SPIRE'98*, pages 7–13. IEEE Computer Society, Sep 1998.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley-Longman, may 1999.

[HMQS96] Georg Haider, Cecilia Magnusson Sjöberg, Gerlad Quirchmayr, and Verena Sebald. The comparative part of the corpus legis project - using SGML for intelligent information retrieval of legal documents. In *EXPERSYS-96, Artificial Intelligence Applications*, pages 181–186, 1996.

[LS00] Hartmut Liefke and Dan Suciu. XMill: an efficient compressor for XML data. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153–164, 2000.

[MT97] Alistair Moffat and Andrew Turpin. On the implementation of minimum-redundancy prefix codes. *IEEE Transactions on Communications*, 45(10):1200–1207, 1997.

[NBY97] Gonzalo Navarro and Ricardo A. Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *Information Systems*, 15(4):400–435, 1997.

[TH02] Pankaj Tolani and Jayant R. Haritsa. XGRIND: A query-friendly XML compressor. In *ICDE*, 2002.

[VdC02] Jesús Vegas, Pablo de la Fuente, and Fabio Crestani. A graphical user interface for structured document retrieval. In *Advances in Information Retrieval, 24th BCS-IRSG European Colloquium on IR Research Proceedings*, volume 2291 of *Lecture Notes in Computer Science*. Springer, March 2002.

[Wel84] Terry A. Welch. A technique for high-performance data compression. *IEEE Computer*, 17(6):8–19, 1984.

[WMB99] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes*. Morgan Kaufmann Publishers, Inc., second edition, 1999.