Una técnica de compresión para documentos de texto considerando su estructura *

Joaquín Adiego[1], Pablo de la Fuente[1], and Gonzalo Navarro[2]

[1]Departamento de Informática, Universidad de Valladolid, Valladolid, España.
{jadiego, pfuente}@infor.uva.es
[2]Departamento de Ciencias de la Computación, Universidad de Chile, Santiago, Chile. gnavarro@dcc.uchile.cl

Resumen En este trabajo se describe una nueva aproximación Lempel-Ziv pensada para comprimir documentos estructurados denominada LZCS, que saca partido de la información redundante que aparece en la estructura de los documentos. La idea principal es que pueden existir subárboles repetidos y éstos se pueden sustituir por una referencia a la primera ocurrencia de los mismos. La principal ventaja aportada es que los documentos que genera la transformación LZCS se pueden visualizar, acceder de forma aleatoria y navegar con facilidad. En una segunda etapa, los documentos procesados se pueden comprimir empleando cualquier técnica semiadaptativa, para que siga siendo posible el acceso aleatorio y la navegación por los mismos. LZCS es especialmente eficiente a la hora de comprimir colecciones con documentos muy estructurados, como los formularios XML utilizados en aplicaciones de comercio electrónico y en los documentos intercambiados en los servicios web. La comparación con otros compresores, estándares o basados en la estructura, muestra que LZCS es una elección muy competitiva para este tipo de documentos, mientras que los otros compresores no están pensados para soportar la navegación o el acceso aleatorio sobre los documentos comprimidos.

Palabras clave: Textos semiestructurados, XForms, Compresión de XML, Ziv-Lempel.

1. Introducción

El almacenamiento, intercambio y manipulación de textos semiestructurados como medio de representación de datos estructurados está proliferando en todo tipo de aplicaciones, desde las bases de datos textuales y las bibliotecas digitales hasta los servicios web y el comercio electrónico. El formato XML, se está convirtiendo en un estándar utilizado para codificar información con una estructura simple o compleja por un lado y fija o variable por el otro. Aunque

* Trabajo en parte financiado por el proyecto CYTED VII.19 RIBIDI (todos los autores), el poyecto TIC2003-09268 del MCyT, España (primer y segundo autor) y el proyecto Millennium Nucleus Center for Web Research, Grant P01-029-F, Mideplan, Chile (tercer autor).

hace algún tiempo XML se previó como un mecanismo para describir datos estructurados no ha sido hasta la reciente explosión de la "empresa electrónica" cuando ha mostrado su potencial para describir y representar los diferentes tipos de documentos que almacenan e intercambian las empresas (facturas, albaranes, recibos, etc.).

Aunque la información que gestiona una empresa se puede almacenar en bases de datos relacionales o almacenes de datos, en la actualidad está proliferando el hecho de almacenar copias digitales, en formato XML, de toda la documentación que se ha producido o intercambiado con el paso del tiempo, porque un sistema de recuperación de información estructurada podría proporcionar acceso aleatorio a todos esos documentos estructurados y también se podrían buscar, visualizar y navegar fácilmente. Como valor añadido, sería deseable que este repositorio ocupara el menor espacio posible.

El presente trabajo se presenta una propuesta encaminada a comprimir de forma eficiente textos muy estructurados (que generalmente representan datos estructurados, como los almacenados en una base de datos relacional), como por ejemplo los formularios en los que dentro de cada campo aparece normalmente una cadena de texto pequeña. Las colecciones compuestas por este tipo de documentos/formularios contienen mucha redundancia que no es manipulada correctamente por los métodos de compresión clásicos. Al mismo tiempo, se desea que la colección comprimida pueda ser navegable, visualizable y accesible de forma sencilla. Existen métodos de compresión que tienen en cuenta la estructura pero que no tienen en consideración estas capacidades, por lo que los textos se deben descomprimir antes de acceder a ellos.

Se ha desarrollado un método de compresión inspirado en el esquema de compresión Lempel-Ziv en el que se colapsan subárboles repetidos bajo una referencia. El método, denominado LZCS[1], ha obtenido unas tasas de compresión excelentes que superan a las obtenidas con los métodos clásicos y competitiva con los métodos que consideran la estructura durante el proceso de compresión.

Además el algoritmo LZCS codifica el texto realizando una única pasada sobre el texto por lo que puede emitir el texto comprimido a medida que procesa el mismo sin comprimir. Esto hace que LZCS sea adecuado para su utilización sobre una red de comunicación sin introducir ningún retardo en la transmisión. La salida del algoritmo LZCS sigue siendo un texto plano, lo que facilita su transmisión sobre canales ASCII específicos. Por otro lado, la salida del algoritmo LZCS se puede comprimir en una segunda pasada utilizando cualquier método de codificación, pero es preferible que se utilice un método de codificación que mantenga las propiedades de navegabilidad y acceso aleatorio sobre el texto codificado.

[1] Siglas procedentes de "Lempel-Ziv Compression of Structure", su denominación anglosajona

2. Compresión de texto

2.1. Comprimiendo texto plano

Por lo general, los métodos clásicos empleados en la compresión de texto no consideran la estructura de los documentos que comprimen. A finales de los setenta, Lempel y Ziv diseñaron nuevas tecnologías de compresión de datos basadas en la sustitución de subcadenas de texto por referencias a ocurrencias previamente repetidas. Sus dos algoritmos más famosos se han denominado LZ77[ZL77] y LZ78 [ZL78], así como la última variante LZW [Wel84]. Dependiendo de las variantes de los algoritmos, se pueden referenciar unas cadenas previas u otras. Estas técnicas no tienen en consideración el significado semántico de las secuencias sustituidas. La familia de compresores Lempel-Ziv es la más popular en compresión de texto porque combina unas buenas razones de compresión con la velocidad durante los procesos de compresión y descompresión.

En lo referente a la compresión de textos escritos en lenguaje natural para permitir una recuperación eficiente, las técnicas más exitosas se han basado en modelos en los que se han considerado las palabras (en vez de los caracteres) como los símbolos de la fuente de información [Mof89]. Las palabras reflejan mucho mejor que los caracteres la verdadera entropía del texto [BCW90]. Por ejemplo, al comprimir lenguaje natural, un codificador semiadaptativo de Huffman que considere caracteres como símbolos típicamente obtiene un fichero comprimido cuyo tamaño está en torno al 60 % del tamaño del original. Un codificador de Huffman que considere palabras como símbolos obtiene un 25 % [ZMGBY00]. Otro ejemplo es el algoritmo WLZW que implementa un esquema Ziv-Lempel sobre palabras [BSTW86].

La mayoría de los sistemas de recuperación de información consideran que las palabras son sus principales elementos atómicos; por otro lado, el hecho que coincida el alfabeto y el vocabulario de las colecciones de textos permite una búsqueda eficiente y altamente sofisticada tanto en búsqueda secuencial como en los índices invertidos comprimidos sobre los textos. Por consiguiente, una compresión basada en palabras facilita la integración con un sistemas de recuperación de información. Algunos ejemplos existosos de integración se pueden encontrar en [WMB99,NMN$^+$00,MW01].

2.2. Comprimiendo texto estructurado

SCM [AGF03] es un modelo genérico empleado para comprimir documentos semiestructurados que aprovecha la información del contexto que, habitualmente, se encuentra implícita en la estructura del texto. La idea es utilizar modelos independientes para comprimir el texto que reside en cada tipo de elemento estructural diferente (por ejemplo, cada etiqueta XML diferente) debido a la creencia de que distribución de los textos que se encuentran en un mismo tipo de estructura debe ser similar y diferente a la de otros tipos.

Otro método de compresión que tiene en cuenta la estructura de los documentos es *XMill* [LS00], desarrollado en los laboratorios AT&T. *XMill* es un compresor específico para XML diseñado para intercambiar y almacenar documentos

XML, y su técnica de compresión no está pensada para soportar búsquedas directas o actualización de la base de datos comprimida. *XMill* está basado en la librería *zlib* como motor principal de compresión, que combina una compresión Ziv-Lempel con una variante de Huffman.

Existen otras aproximaciones para comprimir datos XML basadas en la utilización de codificadores PPM que sacan partido a la estructura. Un ejemplo es *XMLPPM* [Che01] [Che01] el cual es un compresor adaptativo que utiliza diferentes modelos PPM. *XMLPPM* utiliza un parser ESAX, una variante de SAX, para obtener diferentes partes del documento (entendiéndose por "partes" los nombres de las etiquetas, los nombre y valores de los atributos de las etiquetas, el texto, etc.), cada parte se codifica mediante un modelo PPM diferente. *XMLPPM* es un compresor adaptativo y no se pueden realizar búsquedas ni accesos aleatorios sobre los textos comprimidos resultantes.

3. Descripción de LZCS

LZCS es una nueva técnica para comprimir texto estructurado que permite navegar por la estructura comprimida. Por lo tanto, LZCS se puede integrar en un sistema de recuperación con textos estructurados sin perder eficiencia cuando se efectúan las búsquedas o cuando se visualizan los resultados. La idea principal se basa en el concepto introducido por Ziv y Lempel, de manera que los elementos estructurales y los bloques de texto repetidos se reemplazan por una referencia a la primera aparición en el texto procesado. El resultado es un texto estructurado válido con etiquetas especiales adicionales que se utilizan para representar las referencias en el texto. Además, dicho texto estructurado se puede transmitir, manipular o visualizar empleando los mecanismos habituales, o incluso se puede comprimir utilizando algún compresor estándar sin pérdida.

Los documentos se visualizan de forma habitual hasta que aparezca una referencia que, como ya se ha comentado, se encuentra representada mediante una etiqueta especial. Cuando aparece una referencia será preciso introducir la posición actual en una pila (para poder reanudar el proceso una vez que la referencia se haya resuelto) y a continuación se realiza un desplazamiento a la posición indicada por la referencia. La referencia se resolverá (finalizará) cuando se alcance la correspondiente etiqueta que señale el final del elemento estructural si la posición referenciada comienza con una etiqueta que indica el inicio de un elemento estructural. Si, por el contrario, la referencia no comienza con una etiqueta de inicio, se resolverá cuando aparezca una etiqueta de inicio. Cuando termina el texto referenciado se debe sacar de la pila la posición de origen previa y el proceso continua. En algunos casos pueden aparecer otras referencias en el texto referenciado, en este caso se repite el mismo proceso. Se puede emplear un procedimiento similar para buscar o navegar por la estructura arbórea del documento.

Puesto que los documentos que genera LZCS son navegables, una buena idea es comprimirlos empleando un método de compresión semiadaptativo, como por ejemplo Huffman orientado a palabra. Después de esto los documentos no se pueden visualizar como texto convencional (pues es necesario descomprimirlos) pero continúan siendo navegables y accesibles de manera aleatoria. A continuación se describirá formalmente la transformación LZCS.

3.1. Definición formal

Definición 1 (Bloque de texto) *Un bloque de texto será cualquier secuencia maximal de caracteres alfanuméricos consecutivos que no contenga etiquetas que señalen el inicio o fin de un elemento estructural o represente una referencia.*

Definición 2 (Elemento estructural) *Un elemento estructural será cualquier secuencia de caracteres concecutivos que comience con una etiqueta de inicio de elemento estructural y que termine con la etiquete de final del elemento estructural correspondiente.*

Teniendo presente la definición anterior, un elemento estructural puede contener uno o más bloques de texto, uno o más elementos estructurales y/o una o más referencias. Para simplificar, los otros tipos de etiquetas válidas que pueden aparecer en un documento XML (como por ejemplo, etiquetas de comentario, etiquetas autocontenidas y demás) serán consideradas como si se tratasen de texto convencional y sólo las etiquetas que marcan el principio y final de los elementos estructurales se utilizarán para identificar a dichos elementos.

Es habitual representar la estrcutura de los documentos en forma de árbol. Si en este caso se utiliza esa representación, los bloques de texto se representarán mediante subárboles.

Definición 3 (Nodo) *Un nodo será bien un bloque de texto, bien un elemento estructural.*

Una vez definido el concepto de nodo se puede decir que el objetivo principal de LZCS es sustituir subárboles por referencias a subárboles equivalentes que aparecieron con anterioridad.

Definición 4 (Nodos equivalentes) *Sean $\mathcal{N}_1$ y $\mathcal{N}_2$ dos nodos que aparecen en una colección. Diremos que el nodo $\mathcal{N}_1$ es equivalente al nodo $\mathcal{N}_2$ si y sólo si $\mathcal{N}_1$ es textualmente igual a $\mathcal{N}_2$.*

Definición 5 (Transformación LZCS) *La transformación LZCS sustituye cada nodo maximal equivalente a otro, que apareció con anterioridad, por una referencia a la primera ocurrencia del mismo en el texto. El resto de elementos permanece inalterado. Por "maximal" se entiende que el nodo sustituido no es un descendiente de otro que pueda ser reemplazado.*

Una *referencia* se representa en la salida mediante una etiqueta especial, esta etiqueta especial se construye mediante los símbolos `<@` y `>` que se utilizan para marcar respectivamente el pricipio y el final de la etiqueta de referencia. El contenido de esta etiqueta estará formado por dígitos que expresan una cantidad entera positiva, la cual indica la posición absoluta (evidentemente, dicha posición será menor que la posición actual) en la que comienza el elemento referenciado. Por cuestiones de optimización de espacio (y consecuentemente, para mejorar la razón de compresión) se sugiere expresar este número en una base grande, por ejemplo en base 62 utilizando como dígitos los caracteres `0..9`, `A..Z` y `a..z`.

En algún momento puede suceder que un bloque de texto referenciado sea más pequeño que la propia etiqueta de referencia (como por ejemplo cuando el bloque de texto está formado únicamente por el caracter `'\n'`). Bajo estas circunstancias, el hecho de sustituir el bloque por la referencia no es una buena elección pues el resultado de la sustitución provoca un aumento de tamaño, justo lo contrario que se desea obtener. Por lo tanto no se sustituirán los bloque de texto que sean más pequeños que un parámetro l especificado por el usuario. La elección del valor del parámetro l influye sobre el valor obtenido de la razón de compresión, pero no sobre la validez de la transformación.

3.2. Ejemplo

A continuación se ilustrará el funcionamiento de la transformación LZCS mediante un ejemplo. Para simplificar supongamos que se desea comprimir con LZCS una colección formada por tres documentos estructurados. La estructura de los documentos está representada en la figura 1 y está formada por tres elementos estructurales diferentes. Cada tipo de elemento estructural está representado mediante un círculo: el elemento estructural de tipo 1 está representado por un círculo dibujado con una línea continúa, el elemento estructural de tipo 2 dibujado con una línea discontinua y el de tipo 3 dibujado con una línea punteada. Los bloques de textos están representados mediante cuadrados y las letras y números actúan como los identificadores de los nodos.

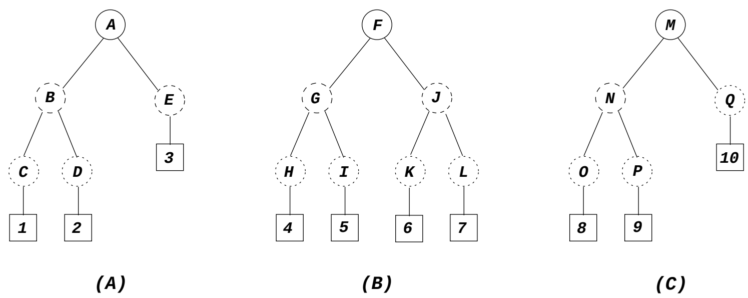


Figura 1. Tres documentos de ejemplo.

Para cubrir todas las posibilidades, supongamos que los bloques de texto identificados en la figura mediante 1, 4, 7 y 9 son equivalentes. También son equivalentes los bloques 3 y 10 por una parte y los bloques 6 y 8 por otra. Con estas equivalencias entre bloques de texto los documentos tienen partes repetidas (o dicho con otras palabras, los documentos tienen subárboles equivalentes). La figura 2 muestra de una manera visual estas correspondencias y la colección transformada mediante LZCS, en la que los triángulos representan a las referencias.

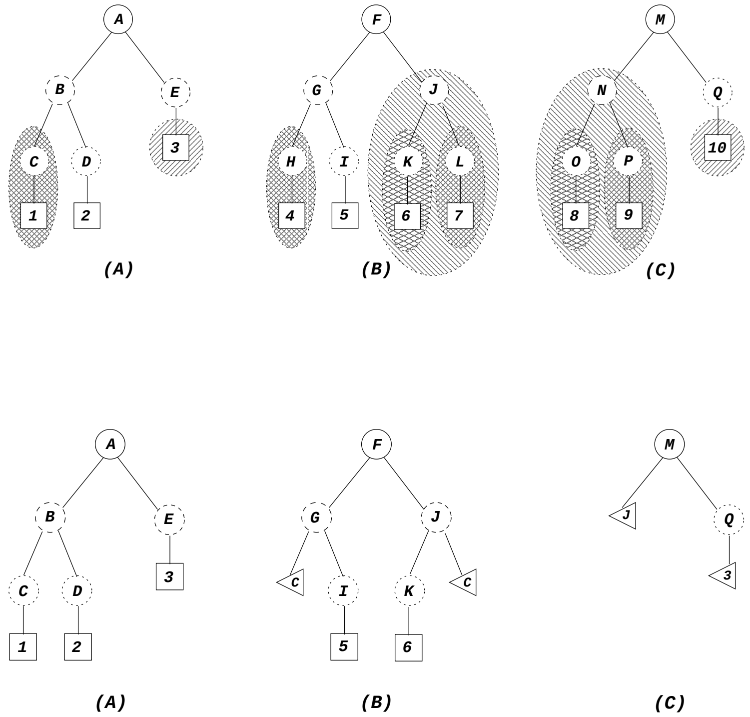


Figura 2. Subárboles equivalentes en los documentos y los tres documentos de ejemplo después de aplicar la transformación LZCS. Las referencias están representadas mediante triángulos.

4. Evaluación de la respuesta

El modelo LZCS se ha probado utilizando diferentes colecciones de XForms, las cuales se corresponden con documentos reales utilizados en la pequeña y mediana empresa chilena. XForms[2], un dialecto de XML, es una recomendación candidata de W3C con el objetivo de especificar formularios web, que separa de una manera muy precisa el contenido semántico de los aspectos de presentación.

[2] `http://www.w3.org/Markup/Forms`

En concreto, la utilización de XForms está siendo habitual en la representación e intercambio de información y transacciones entre empresas.

Por cuestiones de privacidad no se han podido utilizar bases de datos de XForms actuales, pero podemos simularlas de una manera muy precisa. Se han obtenido cinco tipos diferentes de formularios (por ejemplo facturas) que tienen varios campos y cada uno tiene un vocabulario controlado (por ejemplo nombres de piezas o fechas) al que se ha tenido acceso. Por lo tanto, se ha generado el contenido de las bases de datos de forma aleatoria, eligiendo el contenido de cada campo de su vocabulario controlado correspondiente. Aunque las bases de datos obtenidas de esta manera son válidas para experimentar con ellas, es necesario destacar que los datos reales posiblemente tengan más redundancia que los datos obtenidos de forma aleatoria y, por lo tanto, nos encontramos ante una "situación pesimista". A continuación se realiza una breve descripción de los cinco tipos de formularios que se han utilizado.

- XForm de tipo 1: Centralización de remuneraciones. Representa la contabilización de las remuneraciones mensuales, por cantidades totales y con desglose de su composición. Es un documento muy usado.
- XForm de tipo 2: Factura de venta. Es un documento legal chileno.
- XForm de tipo 3: Factura de compra. Es un documento legal chileno, similar al anterior.
- XForm de tipo 4: Orden de trabajo. Es el documento que usan las empresas de proyectos de instalaciones de calefacción para registrar el detalle contable de un trabajo contratado.
- XForm de tipo 5: Cubicación de trabajo. Es el documento que usan en las empresas que construyen letreros y señales bajo demanda para determinar las partes y el costo de un trabajo a desarrollar. Las empresas constructoras usan un documento similar.

Para realizar los experimentos se han utilizado las colecciones de XForms de tipo 4 y 5 por separado y de forma indivisible debido a su pequeño tamaño; por otro lado, se han seleccionado diferentes tamaños de colecciones de XForms de tipo 1, 2 y 3.

En todos los experimentos, se ha probado LZCS variando el valor del parámetro l (ver sección 3.1) para comprobar la incidencia del valor de l, que indica cuál será el tamaño mínimo de los bloques de texto para que éstos sean candidatos a ser sustituidos, sobre la razón de compresión.

La figura 3 muestra la evolución de las razones de compresión cuando se utilizan diferentes valores de l para las colecciones de XForms de tipo 3. Con los otros tipos de formularios se han obtenido resultados similares. La razón de compresión se define como el tamaño del texto comprimido dividido entre el tamaño del texto sin comprimir. En este caso todavía no se ha aplicado ningún tipo de compresión después de realizar la transformación LZCS.

Como se puede comprobar, la peor compresión se ha obtenido (en todos los casos) cuando $l = 0$, es decir, cuando se han realizado todas las sustituciones posibles. La compresión cuando $l = \infty$ ha obtenido unos resultados intermedios,

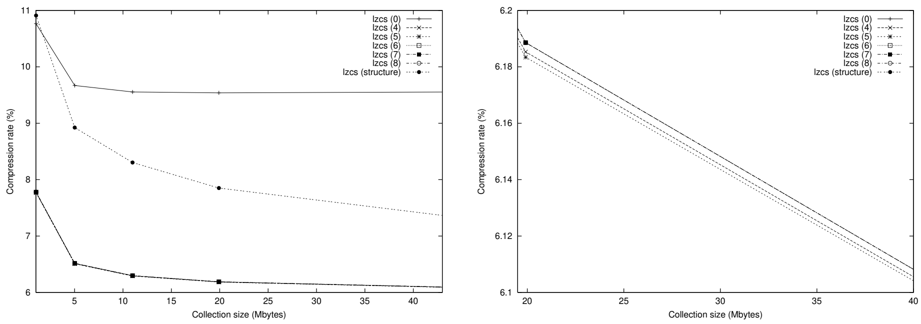


Figura 3. Razones de compresión para valores diferentes de l, para las colecciones XForms de tipo 3. La representación de la derecha es una ampliación de la representación de la izquierda.

obteniendo para las colecciones más grandes una reducción en el tamaño del texto en torno al 28$l = \infty$ es aún mucho peor que en los casos con valores de l intermedios. Cuando se tienen diferentes valores intermedios de l se obtiene una compresión similar con variaciones muy pequeñas. En estos casos, la compresión mejora en una 18 % respecto a la obtenida cuando $l = \infty$ y en un 42 % respecto a la obtenida cuando$l = 0$ para las colecciones de mayor tamaño.

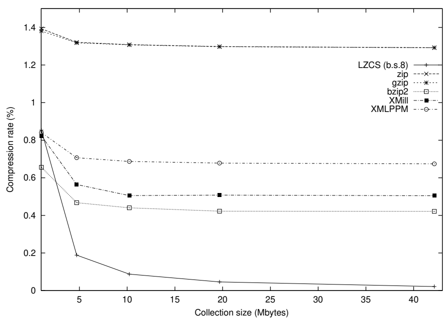
A continuación se comparará LZCS con respecto al codificador de Huffman basado en palabras [Mof89]. La figura 4 muestra las mejores razones de compresión obtenidas por cada método y tipo de documento. La columna "LZCS" muestra la compresión obtenida cuando sólo se aplica la transformación LZCS y la columna "LZCS + WH" muestra la compresión obtenida una vez que se ha realizado una codificación de de Huffman basada en palabras sobre el texto transformado mediante LZCS.

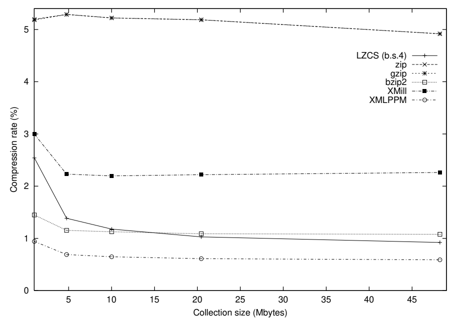
Collection / Method	Word Huffman	LZCS (first stage)	LZCS (complete)
XForms 1	9.6935 %	0.037432 %	0.021522 %
XForms 2	12.646 %	4.3111 %	0.92209 %
XForms 3	11.550 %	6.0872 %	1.3294 %
XForms 4	13.994 %	4.8861 %	0.89281 %
XForms 5	12.441 %	3.6245 %	0.83933 %

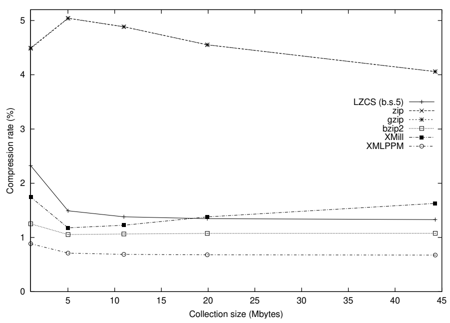
Figura 4. Mejores razones de compresión obtenidas para cada método y tipo de documento.

En cualquier caso la compresión que ha obtenido la transformación LZCS de por sí es sorprendentemente buena, sobre todo si se tiene en cuenta que la salida de la transformación sigue siendo un documento de texto. Cuando se aplica una codificación de Huffman basada en palabras sobre el texto transformado se obtiene una compresión todavía mejor llegando a reducir el texto transformado por LZCS de un 20 % a un 60 % de su tamaño.

Por último, se comparará LZCS contra otros sistemas de compresión que ni permiten navegar ni acceder de forma aleatoria sobre los textos comprimidos. Se han elegido dos tipos de sistemas de compresión para comparar: unos consideran la estructura a la hora de comprimir (como *X*Mill y *XMLPPM* descritos brevemente en la sección 2) y los otros son estándar. La mayoría de los sistemas estándar están basados en los esquemas LZ clásicos. Los sistemas estándar que se han elegido para realizar la comparación contra LZCS son aquellos que generalmente se encuentran disponibles que cualquier distribución de Linux, son los siguientes: (1) compress de UNIX, que implementa el algoritmo LZW; (2) *z*ip y (3) *g*zip, que utilizan el algoritmo LZ77 junto a una variante del algoritmo de Huffman; (4) *b*zip2, que aplica una codificación de Huffman a un bloque de texto previamente transformado mediante la transformación de Burrows-Wheeler. En general, la compresión obtenida por *b*zip2 es considerablemente mejor que la que obtienen los compresores más convecionales basados en LZ77/LZ78 y se aproxima a las obtenidas por la famila de compresores estadísticos PPM.







Method / Size	T.4 (7.19 Mb)	T.5 (5.74 Mb)
zip	2.1056 %	4.4351 %
gzip	2.1040 %	4.4331 %
compress	10.300 %	10.396 %
bzip2	0.95276 %	0.84302 %
XMill	0.94260 %	0.92420 %
XMLPPM	0.71240 %	0.55300 %
Word Huffman	13.994 %	12.441 %
LZCS $l = 0$	0.89281 %	0.93944 %
LZCS $l = 4$	0.89373 %	0.84746 %
LZCS $l = 5$	0.89672 %	0.84630 %
LZCS $l = 6$	0.90421 %	0.83933 %
LZCS $l = 7$	0.89704 %	0.83935 %
LZCS $l = 8$	0.89309 %	0.84139 %
LZCS $l = \infty$	1.9535 %	1.2566 %

Figura 5. Comparación entre LZCS y otros sistemas de compresión para las colecciones de XForms types 1 (arriba a la izquierda), de tipo 2 (arriba a la derecha), 3 (abajo a la izquierda), 4 y 5 (abajo a la derecha).

Se han comprimido todas las colecciones con los sistemas descritos con anterioridad y las razones de compresión obtenidas se muestran en la figura 5. Tanto en las tablas como en las gráficas, LZCS utiliza el valor de l que obtiene la mejor compresión en cada caso.

Entre los compresores estándar, *compress* y Huffman orientado a palabra obtienen las peores razones de compresión, las cuales no son competitivas en este experimento. A éstos les siguen *zip* y *g*zip, ambos con niveles de compresión muy similares. El mejor compresor dentro de la categoria de los compresores estándar es, sin lugar a dudas, *bzip2* que, en general, sigue siendo inferior a LZCS aunque en algunos casos por un pequeño margen. El motivo de estos resultados (relativamente malos) de los compresores estándar se debe a que estas técnicas no tienen en consideración la estructura de los documentos en el proceso de compresión, hecho que la LZCS obtiene una ventaja significativa. Además, es necesario recordar que LZCS permite navegar y acceder de manera aleatoria sobre el texto comprido, lo que no es nada sencillo con los compresores estándar.

En el caso de los métodos que consideran la estructura de los documentos a la hora de comprimir se puede comentar que LZCS es significativamente mejor que *XMill* en todas la colecciones, obteniendo textos comprimidos un 5pequeños en el peor caso y veinticinco veces más pequeños en el mejor. Por otro lado, *XMLPPM* obtiene la mejor compresión en la mayoría de los casos, con la notable excepción de las colecciones de XForms de tipo 1 en las que LZCS está a mucha distancia de ser superado. El principal problema que tiene *XMLPPM* es que es adaptativo y, por consiguiente, no es adecuado para navegar o acceder de manera aleatoria sobre los documentos comprimidos.

5. Conclusiones y trabajo futuro

Se ha presentado un método denominado LZCS, un esquema de compresión inspirado en el esquema Lempel-Ziv, pensado para comprimir documentos estructurados. La idea principal de LZCS es sustituir subestructuras completas por una referencia a una ocurrencia previa de la misma, de manera que se capture la redundancia que introduce la estructura en una colección de documentos. En general, la transformación LZCS tiene las siguientes ventajas: (1) Muy buenas razones de compresión, que mejoran las obtenidas por los métodos clásicos y la mayoría de las obtenidas por los métodos de compresión que consideran la estructura. (2) Facilidad en la navegación, visualización y acceso aleatorio sobre las colecciones comprimidas. (3) Compresión y descompresión en una única etapa y rapidez del proceso.

En la experimentación realizada, sólo *XMLPPM* comprime mejor que LZCS pero, como ya se ha comentado con anterioridad, con *XMLPPM* es imposible realizar un acceso aleatorio a un documento en concreto puesto que es adaptativo y necesita descomprimir todos los documentos que preceden al deseado. Con todo lo dicho, se puede descartar el uso de *XMLPPM* en el escenario de las bases de datos textuales comprimidas.

En muchos escenarios se pueden añadir documentos a la colección, pero en ningún caso se eliminarán o modificarán. LZCS se puede adaptar con facilidad para permitir la inserción de nuevos documentos, pero se necesita más trabajo para permitir la modificación o el borrado de documentos. Asimismo, puede ser interesante diseñar esquemas de indexación que permitan realizar buscar con rapidez documentos que contengan algunos términos o subestructuras dadas, teniendo presente que la colección está comprimida.

Agradecimientos. Agradecemos a Pablo Palma, de Hypernet Ltd. (Chile), por proporcionarnos las muestras de datos casi reales que se han utilizado en los experimentos.

Referencias

[AGF03] J. Adiego, G. Navarro, and P. Fuente. SCM: Structural contexts model for improving compression in semistructured text databases. In *Proc. 10th Intl. Symp. on String Processing and Information Retrieval (SPIRE'03)*, LNCS 2857, pages 153–167. Springer, 2003.

[BCW90] T. Bell, J. Cleary, and I. Witten. *Text Compression.* Prentice Hall, Englewood Cliffs, N.J., 1990.

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

[Che01] J. Cheney. Compressing XML with multiplexed hierarchical PPM models. In *Proc. Data Compression Conference (DCC 2001)*, pages 163–, 2001.

[LS00] H. Liefke and D. Suciu. XMill: an efficient compressor for XML data. In *Proc. ACM SIGMOD 2000*, pages 153–164, 2000.

[Mof89] A. Moffat. Word-based text compression. *Software - Practice and Experience*, 19(2):185–198, 1989.

[MW01] A. Moffat and R. Wan. RE-store: A system for compressing, browsing and searching large documents. In *Proc. 8th Intl. Symp. on String Processing and Information Retrieval (SPIRE 2001)*, pages 162–174, 2001.

[NMN+00] G. Navarro, E. Silva de Moura, M. Neubert, N. Ziviani, and R. Baeza-Yates. Adding compression to block addressing inverted indexes. *Information Retrieval*, 3(1):49–77, 2000.

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

[WMB99] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes.* Morgan Kaufmann Publishers, second edition, 1999.

[ZL77] J. Ziv and A. Lempel. An universal algorithm for sequential data compression. *IEEE Trans. on Information Theory*, 23(3):337–343, 1977.

[ZL78] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, IT-24(5):530–536, 1978.

[ZMGBY00] N. Ziviani, E. Moura, G. Navarro, and R. Baeza-Yates. Compression: A key for next-generation text retrieval systems. *IEEE Computer*, 33(11):37–44, November 2000.