Usando técnicas de compresión de textos en bibliotecas digitales*

Eva L. Iglesias[2], Nieves R. Brisaboa[1], José R. Paramá[1], Antonio Fariña[1], Gonzalo Navarro[3], María F. Esteller[1]

[1] Laboratorio de Bases de Datos, Univ. da Coruña, Facultade de Informática, Campus de Elviña s/n, 15071 A Coruña, España.
{brisaboa,fari,parama}@udc.es, mfesteller@yahoo.es
[2] Departamento de Informática, Univ. de Vigo , Escola Superior de Enxeñería Informática, Campus As Lagoas s/n, 32001, Ourense, Espa˜na.
eva@uvigo.es
[3] Dept. of Computer Science, Univ. de Chile, Blanco Encalada 2120, Santiago, Chile.
gnavarro@dcc.uchile.cl

Resumen El almacenamiento de los textos de una biblioteca digital en formato comprimido es una alternativa que se hace cada vez más interesante a medida que las colecciones textuales crecen. Sin embargo, la mayoría de las técnicas de compresión impiden la búsqueda de palabras sobre el texto comprimido sin descomprimirlo de modo que se hace imposible aplicar eficientemente técnicas de *text retrieval.*

Recientemente han aparecido algunas técnicas específicas de compresión de textos que permiten la búsqueda de palabras directamente sobre el texto comprimido sin necesidad de descomprimirlo. En este trabajo se introducen dichas técnicas y se presenta un nuevo método de compresión de textos que denominamos *Código Denso con Post-Etiquetado* que no sólo tiene un menor coste computacional sino que, además, consigue mejores ratios de compresión, conservando plenamente las capacidades de búsqueda sobre el texto comprimido de palabras exactas, frases, búsqueda aproximada, etc. de las técnicas anteriores.

Por otro lado, las técnicas de compresión de textos nacieron para la compresión del inglés y no se adaptan bien a lenguas romances que presentan una distribución de las frecuencias de las palabras bastante diferente. Por ello, en este trabajo presentamos una técnica de preprocesado de los textos en lenguas romances para que la posterior compresión resulte más adaptada a sus características.

Palabras relevantes: Compresión de textos, Lenguas Romances, Segmentación de palabras, Ley de Heaps, Ley de Zipf

* Este trabajo está siendo parcialmente subvencionado por CICYT (TEL99-0335-C04-02 y TIC2002-04413-C04-04)

1. Introducción

En los últimos años estamos asistiendo a un espectacular desarrollo de las tecnologías de la información y las comunicaciones. Esto ha favorecido el crecimiento en número y tamaño de las colecciones de textos en lenguaje natural.

Cuando se manejan grandes volúmenes de información es necesario que los documentos se almacenen y recuperen eficientemente. La cantidad de espacio requerido para almacenar los textos se puede reducir significativamente utilizando *técnicas de compresión*, mientras que la mejora en la eficiencia de la recuperación se logra aplicando técnicas de *text retrieval (Recuperación de Textos)* [1,3] que permiten realizar búsquedas más complejas sobre los documentos.

Las técnicas de compresión clásicas, como los algoritmos de Ziv y Lempel [11,12] o el algoritmo de Huffman [4], no resultan adecuadas para bases de datos textuales. Una importante desventaja de estos métodos es la ineficiencia a la hora de realizar búsquedas directamente sobre el texto comprimido. En los últimos años se han desarrollado nuevas técnicas de compresión especialmente indicadas para su aplicación sobre grandes corpus de textos. Entre ellas cabe destacar las codificaciones *Plain Huffman* y *Tagged Huffman* propuestas por Moura et al. [8]. Estos métodos de compresión obtienen unos ratios de compresión próximos al 30% al mismo tiempo que permiten búsquedas directas sobre el texto comprimido que son hasta 8 veces más rápidas que la búsqueda sobre el texto original.

En este trabajo se presenta un nuevo esquema de codificación, denominado *Codificación Densa con Post-Etiquetado*, que emplea un algoritmo para la generación de códigos más sencillo que los dos métodos antes citados, al mismo tiempo que combina sus mejores propiedades.

Las tres técnicas de compresión antes citadas (Plain Huffman, Tagged Huffman y codificación Densa con Post-Etiquetado) han sido probadas extensivamente sobre textos en inglés. Sin embargo, no se adaptan bien a lenguas romances que poseen un vocabulario de palabras mucho mayor debido a la gran variación morfológica de estas lenguas. Este hecho reduce de forma considerable la eficiencia de las búsquedas. Parece necesario realizar variaciones de las técnicas de compresión mostradas para que se adapten mejor a lenguas que posean una amplia variación gramatical, como es el caso de las lenguas romances.

El resto del artículo se estructura del siguiente modo. En la Sección 2 se introducen las codificaciones Plain y Tagged Huffman para dar paso, en la siguiente sección, a la presentación de nuestro nuevo método de compresión. En la Sección 4 se muestran los ratios de compresión alcanzados al aplicar las técnicas anteriores sobre varios corpus en inglés.

En la segunda parte de este trabajo presentamos una técnica para utilizar nuestro método de compresión sobre lenguas romances de modo que se faciliten las tareas posteriores de recuperación de textos. Finalmente, en la Sección 7 se presentan nuestras conclusiones.

2. Técnicas anteriores de compresión para textos en lenguaje natural

En los últimos años se han desarrollado nuevas técnicas de compresión especialmente indicadas para su aplicación sobre textos en lenguaje natural. Estas técnicas surgen con la premisa de permitir la realización de búsquedas directamente en el texto comprimido, evitando así la necesidad de descomprimir antes de buscar. Todas ellas se basan en la utilización de las palabras como los símbolos a comprimir [5] (en lugar de los caracteres, como sucede en los métodos clásicos). Esta idea encaja perfectamente con los requerimientos de los algoritmos de Recuperación de Textos, dado que las palabras son generalmente los átomos de dichos sistemas.

En [8] Moura et al. presentan dos esquemas de compresión basados en palabras que utilizan Huffman para la codificación. Ambos métodos usan un modelo semi-estático; esto es, en el proceso de codificación se realiza una primera pasada para obtener las frecuencias de las *V* palabras distintas del texto original y, en una segunda pasada, se lleva a cabo la compresión del texto. Cada palabra del texto original se reemplaza por un código que es una secuencia de símbolos de un alfabeto de codificación formado por *d* símbolos.

Los métodos de Moura et al. siguen la idea principal del método Huffman clásico [4]; esto es, pretenden codificar el texto original de forma que se asignen códigos más cortos a aquellos símbolos más frecuentes. Los códigos generados son de *prefijo libre*, lo que significa que ningún código es prefijo de otro. La ventaja de las codificaciones de prefijo libre es que cualquier código puede ser descomprimido sin referenciar a los posteriores debido a que el .n de un código siempre es reconocible.

Para la generación de los códigos Huffman se utiliza un árbol que se construye a partir de las probabilidades de los símbolos (palabras, en este caso) que forman el texto a comprimir. La construcción del árbol Huffman comienza creando un nodo hoja por cada palabra del documento original. A continuación se crea un nodo padre para las $[1 + ((V - d) mod (d - 1))]$ palabras de menor frecuencia de aparición [7] (o para las *d* palabras de menor frecuencia cuando $((V - d) mod (d - 1)) = 0$). Al nuevo nodo padre se le asigna una frecuencia que es igual a la suma de las frecuencias de sus nodos hijo. Esta operación se repite sucesivamente eligiendo los *d* nodos con menor probabilidad (ignorando los nodos que ya son hijos) hasta que quede un único nodo sin procesar, que será el *nodo raíz* del árbol (ver [2,6,9] para más detalle).

Una vez construido el árbol ya se puede calcular el código correspondiente a cada palabra del vocabulario. Para ello, en cada nivel del árbol se etiquetan las ramas con los diferentes símbolos del alfabeto de codificación. El código final de cada palabra estará formado por la secuencia de símbolos asignados a las ramas que unen la raíz con la hoja que representa dicha palabra.

Mientras en el método Huffman clásico el alfabeto de codificación está formado por 2 símbolos, 0 y 1 (es decir, los códigos son una secuencia de bits), los nuevos métodos presentados en [13,8] utilizan bytes en lugar de bits (y, por lo tanto, el alfabeto está formado por los números del 0 al 255). Se denominan por ello, métodos de *Codificación Huffman orientada a byte*

basada en palabras. El primero de los métodos, la codificación *Plain Huffman*, emplea los 8 bits de cada byte en el proceso de codificación. El segundo, denominado *Tagged Huffman*, reserva un bit de cada byte como *marca*, lo que permite distinguir claramente el comienzo de cada código dentro del texto comprimido. En concreto, el primer byte de cada código tiene un bit de marca con valor 1, mientras que los bytes restantes del código tienen su bit de marca a 0. Al usar un bit de marca, quedan sólo disponibles 7 bits de cada byte para realizar la construcción del árbol Huffman. Veamos cómo varían los códigos generados por ambas codificaciones en el siguiente ejemplo:

Ejemplo 1 Consideremos un texto con vocabulario *A, B, C, D, E, F, G, H, I, J* cuyas frecuencias de aparición son 0.2, 0.2, 0.15, 0.15, 0.14, 0.09, 0.04, 0.02, 0.015, 0.015. En la Figura 1 se muestra un posible árbol para la codificación *Plain Huffman* y otro para la *Tagged Huffman*, suponiendo, por simplicidad, que los símbolos del alfabeto de codificación formados por 3 bits (en lugar de 8). Los códigos resultantes figuran en la Tabla 1.

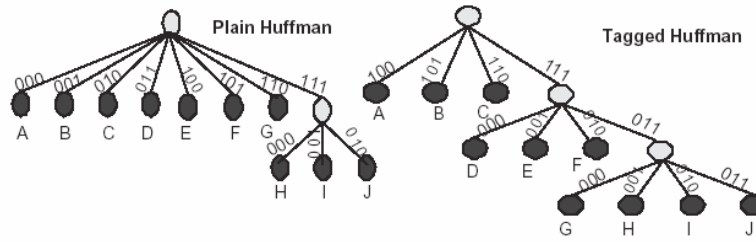

Figura 1. Árboles para Plain Huffman y Tagged Huffman

La codificación Plain Huffman genera códigos más pequeños que Tagged Huffman al utilizar los 8 bits de cada byte en lugar de 7 y, por lo tanto, da lugar a mejores ratios de compresión, pero su eficiencia en la recuperación es peor. La utilización de marcas en la codificación *Tagged Huffman* permite la realización de búsquedas directas sobre el texto comprimido con cualquier algoritmo de emparejamiento de patrones.

3. Codificación Densa con Post-Etiquetado

A continuación se presenta un nuevo esquema de compresión que conserva las ventajas de la codificación Tagged Huffman pero consigue mejores ratios de compresión. La nueva técnica, que denominamos *Codificación Densa con Post-Etiquetado*, y que es nuestra principal aportación en este trabajo, emplea un algoritmo más sencillo para la generación de códigos y no está basado en Huffman. Su sencillez a la hora de la codificación permite agilizar también los procesos de compresión y descompresión del texto. Lo importante es que, a pesar de estas destacables mejoras, se basa igualmente en la utilización de marcas que permiten distinguir los códigos dentro del texto comprimido, por lo que mantiene las mismas posibilidades que la codificación Tagged Huffman

a la hora de realizar búsquedas (exactas o aproximadas) directamente sobre el texto comprimido.

La codificación Densa con Post-Etiquetado es una codificación orientada a byte y basada en palabras donde cada código está formado por una secuencia de bytes. Al igual que sucedía con la codificación Tagged Huffman, el primer bit de cada byte se emplea como bit de marca. La única diferencia estriba en que, en lugar de utilizarlo como marca de comienzo de palabra, ahora el bit indica si el byte actual es el último de un código o no. En concreto, el bit de marca es 1 para el último byte de cada código.

Este cambio tiene sorprendentes consecuencias. El que el bit de marca indique el final de un código es suficiente para asegurar que un código no es nunca *prefijo* de otro, sin importar el contenido de los 7 bits restantes, pues únicamente el bit de marca del último símbolo de cada código vale 1. Por lo tanto, ya no es necesario aplicar codificación Huffman sobre los 7 bits restantes y se pueden utilizar todas las combinaciones de 7 bits para cada uno de los símbolos del código.

Proceso de codificación Una vez eliminada la necesidad de utilizar codificación Huffman, el problema reside en encontrar una asignación de códigos óptima; es decir, una codificación que minimice la longitud de la salida. Se sigue conservando, por tanto, la idea de asignar códigos más cortos a las palabras más frecuentes, como en todas las técnicas de compresión.

El proceso de codificación consta de dos fases: en primer lugar se ordenan las palabras por orden decreciente de frecuencias de aparición; a continuación se realiza una asignación secuencial de códigos a cada palabra utilizando 7 bits y añadiendo el bit de marca a cada uno de los bytes que componen un código. Este bit de marca será 1 en el caso del primer bit del último byte, y 0 en el caso del primer bit de cualquier otro byte del código. De este modo, la primera palabra se codificará como **1**0000000, la 128 como **1**1111111, la palabra 129 como **0**0000000 **1**0000000 y la 130 como **0**0000000 **1**0000001. Tras el código **0**1111111 **1**1111111 se pasa a **0**0000000 **0**0000000 **1**0000000, y así sucesivamente.

Al contrario de lo que sucedía en la codificación Huffman donde la frecuencia de las palabras influía en la longitud de los códigos asignados, esta nueva codificación no depende de dicha frecuencia sino de la posición que una palabra ocupe dentro del vocabulario (estando este ordenado decrecientemente por frecuencias). Así, el código asignado a una palabra que ocupe la posición *i*, con frecuencia 0.15 será el mismo que si esa palabra tuviese frecuencia 0.01.

Los resultados experimentales presentados en la Sección 4 reflejan claramente la importante mejora en las tasas de compresión de la codificación Densa frente a la codificación Tagged Huffman.

En la Tabla 1 se presenta un ejemplo en el que se puede observar la diferencia entre los formatos de los códigos correspondientes a la codificación Plain Huffman y Tagged Huffman frente a la nueva codificación, así como el tamaño del código resultante. En este ejemplo se asume que los símbolos que componen los códigos constan de 3 bits.

Word	Freq	P.H.	Cod Densa	T.H.	Freq*byte P.H.	ETDC	T.H.
A	0,2	000	100	100	0,2	0,2	0,2
B	0,2	001	101	101	0,2	0,2	0,2
C	0,15	010	110	110	0,15	0,15	0,3
D	0,15	011	111	111 000	0,15	0,15	0,3
E	0,14	100	000 100	111 001	0,14	0,28	0,28
F	0,09	101	000 101	111 010	0,09	0,18	0,18
G	0,04	110	000 110	111 011 000	0,04	0,08	0,12
H	0,02	111 000	000 111	111 011 001	0,04	0,04	0,06
I	0,005	111 001	001 100	111 011 010	0,01	0,01	0,015
J	0,005	111 010	001 101	111 011 011	0,01	0,01	0,015
				tamaño medio del código	**1,03**	**1,30**	1,67

Tabla 1. Asignación de códigos y tamaño texto comprimido

4. Estudios experimentales

Se han comprimido varios corpus del trec-2 (AP Newswire 1988 y Zi. Data 1989-1990) y del trec-4 (Congressional Record 1993, Financial Times 1991, 1992, 1993 and 1994) utilizando los métodos: Plain Huffman, Codificación Densa con Post-Etiquetado y Tagged Huffman. Hemos utilizado el *spaceless word model* [7] para crear el vocabulario; esto es, si una palabra es seguida por un espacio, sólo codificamos dicha palabra, en caso contrario se codifican tanto la palabra como el separador.

Corpus	Tamaño original	Palabras voc	Plain H.	Cod. Densa	Tagged H.
AP Newswire 1988	25,099,4525	241,315	31.18 %	32.00 %	34.57 %
Ziff Data 1989-1990	185,417,980	221,443	31.71 %	32.60 %	35.13 %
Congress Record	51,085,545	114,174	27.28 %	28.11 %	30.29 %
Financial Times 1991	14,749,355	75,597	30.19 %	31.06 %	33.44 %
Financial Times 1992	175,449,248	284,904	30.49 %	31.31 %	33.88 %
Financial Times 1993	197,586,334	291,322	30.60 %	31.48 %	34.10 %
Financial Times 1994	203,783,923	295,023	30.57 %	31.46 %	34.08 %

Tabla 2. Comparativa del ratio de compresión

La Tabla 2 muestra el ratio de compresión obtenido por los tres métodos de compresión anteriormente mencionados. En la segunda columna se indica el tamaño original del corpus procesado, y las siguientes columnas muestran el número de palabras del vocabulario y el ratio de compresión de cada método. Como puede apreciarse, Plain Huffman es el método que obtiene mejores ratios de compresión, mientras que la codificación Densa mejora a Tagged Huffman hasta en 2.5 puntos.

5. Problemática de las lenguas romances

La mayoría de los trabajos existentes en el campo de compresión de textos se han realizado sobre textos en inglés. Faltan, por tanto, datos experimentales sobre el comportamiento de cualquiera de las técnicas de compresión comentadas cuando se aplican sobre lenguas romances.

La principal característica diferenciadora entre un corpus en inglés y otro en una lengua romance es la variación morfológica existente en este último. Esto provoca que una misma palabra en inglés se pueda traducir generalmente por una familia de palabras de raíz común con variación en la desinencia. Por

ejemplo, las tres formas típicas de los verbos en inglés corresponden a más de sesenta formas en lenguas romances. Los adjetivos tienen cuatro posibles terminaciones (corresponden a variaciones de género y número) y así sucesivamente. Esto conlleva que nos encontremos una distribución de frecuencias notablemente diferente según el corpus tratado pertenezca a lenguas romances o al inglés. Por tanto, se observa que el vocabulario es mayor en lenguas romances y que posee una distribución de frecuencias más homogénea; es decir, menos sesgada.

Para capturar la variabilidad de los diferentes corpus se utilizan dos leyes, la ley de Heaps y la ley de Zipf.

5.1. Ley de Heaps

Se trata de una ley empírica que relaciona el tamaño del vocabulario (número de palabras distintas) y el número total de palabras en el texto. La Ley de Heaps indica que un texto de $O(n)$ palabras tiene un vocabulario de tamaño $v = O(n^{ß})$ para $0 < ß < 1$, donde $ß$ depende del texto y suele tomar valores entre 0.4 y 0.6.

La Figura 2 (izquierda) ilustra los diferentes tamaños de vocabulario que se obtienen con textos de distintos tamaños tomando los valores de $ß = 0.4$, 0.5 y 0.6. Se puede observar que para un tamaño del corpus dado, el tamaño del vocabulario aumenta a medida que lo hace el parámetro $ß$.

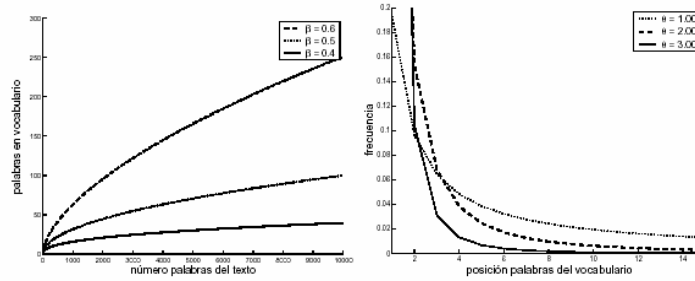

Figura 2. Leyes de Heaps y Zipf

5.2. Ley de Zipf

La ley de Zipf [10] establece que si ordenamos las v palabras del vocabulario de un texto en orden decreciente de frecuencia, la probabilidad de la palabra más frecuente es $i^{?}$ veces la de la i-ésima palabra. Esto significa que la probabilidad de la i-ésima palabra es $p_i = 1/(i^{?}A)$, donde $A = \sum_{j=1}^{V} \frac{1}{j^{q}}$ se utiliza para normalizar las probabilidades.

El parámetro $?$ aporta información respecto a la variabilidad del texto. Así, cuando las frecuencias de las palabras son más similares entre sí $?$ es más pequeño, mientras que si la distribución de frecuencias es más sesgada (es decir, hay palabras con frecuencias muy altas y palabras con frecuencias muy bajas) el valor de $?$ es más alto.

La Figura 2 (derecha) ilustra las probabilidades de las palabras de tres textos con distintos valores de $?$. Se puede observar que cuando más grande es el

valor de *?*, más sesgada es la distribución de frecuencias del corpus (mejor compresión), mientras que valores menores de *?* se asocian a textos con distribuciones de frecuencia más uniformes.

5.3. Influencia de la variación morfológica en la compresión de documentos

Se ha demostrado [8] que, para textos en inglés, el valor de *ß* es lo suficientemente bajo (entre 0.4 y 0.6) para asegurar la eficiencia en la compresión utilizando las técnicas *Plain Huffman* y *Tagged Huffman*, cuando el texto original tiene un tamaño superior a 1 MB. Sin embargo, para textos escritos en lenguas romances -con más variación léxica-, el valor de *ß* es mayor y la compresión obtenida peor debido a que el vocabulario que debe almacenarse es más grande. Debido a ello, el tamaño mínimo del texto para obtener una compresión aceptable parece que, a priori, debe ser mayor al de un texto en inglés.

Por otro lado, la compresión alcanzada con el método de Huffman es mejor cuando la frecuencia de las palabras no es uniforme. Realmente, el mejor ratio de compresión se alcanza cuando la distribución de frecuencias es sesgada; esto es, cuando existen palabras con muchas ocurrencias y el resto del vocabulario tiene pocas ocurrencias, pues de lo contrario habría un gran número de palabras con frecuencia media a las que se les asignan códigos grandes. Es decir, los códigos cortos de las palabras más frecuentes no compensarán los códigos largos de los menos frecuentes si la diferencia entre las palabras más frecuentes y las menos frecuentes no es lo bastante grande.

Como se ha explicado anteriormente, la variación morfológica de las lenguas romances provoca que, dados dos corpus del mismo tamaño, siendo uno de ellos perteneciente a inglés y otro a lenguas romances, el tamaño del vocabulario sea mayor en este último y, consecuentemente, la longitud de los códigos será también mayor, debido a la necesidad de usar más bytes para codificar todo el vocabulario. Esto nos induce a pensar que el ratio de compresión será peor en corpus de lenguas romances. Sin embargo, cabría preguntarse si la distribución de frecuencias más adecuada de los corpus en inglés puede llegar a verse contrarrestada por el hecho de que la longitud media de las palabras es mayor en corpus de lenguas romances. En este caso, la diferencia entre el número de caracteres de la palabra a comprimir y el número de bytes del código que se le asigna es mayor en las palabras de mayor frecuencia.

Es evidente que para contestar estas cuestiones es preciso realizar gran cantidad de experimentación con corpus reales para llegar a tener resultados empíricos concluyentes.

6. Alternativa para la compresión de textos en lenguas romances

Vista la problemática de los métodos de compresión sobre corpus de lenguas romances, y dado que la mayor parte de las búsquedas sobre estos corpus están orientadas principalmente a la raíz de la palabra y no a su desinencia, en

esta sección se propone una nueva técnica para preprocesar el texto antes de comprimirlo: utilizar un *algoritmo de segmentación*.

La *segmentación* es una técnica basada en la división de las palabras en dos partes: la raíz y el sufijo. De este modo, palabras como *andaban* y *cantaban* se sustituyen, en el texto original, por *and -aban* y *cant -aban*, respectivamente. El texto *cantaban andaban jugaban cantaría andaría jugaría cantaré andaré jugaré*, después de la segmentación queda de la forma *cant-aban and-aban jug-aban cant-aría and-aría jug-aría cant-aré and-aré jug-aré*. Claramente se puede observar que antes de la segmentación el vocabulario estaría formado por nueve palabras, mientras que tras la segmentación únicamente hay tres raíces (*cant, and, jug*) y tres sufijos (*aban, aría, aré*).

Una vez realizada la segmentación, las raíces y los sufijos se almacenan en ficheros separados. Posteriormente se procede a comprimir cada uno de los ficheros por separado. Las raíces tendrán una frecuencia de aparición más alta que las palabras completas debido a que a la misma raíz le corresponden muchas palabras y, por lo tanto, darán lugar a mejores ratios de compresión al codificarse separadamente de los sufijos.

Esta separación a la hora de comprimir permite agilizar las búsquedas debido a que el tamaño del fichero de raíces es menor y facilita enormemente la búsqueda de una familia de palabras. Como contrapartida está una mayor complejidad para buscar palabras exactas (raíz y desinencia) y un peor ratio de compresión debido a que cada palabra es sustituida por dos códigos (uno para la raíz y otro para la desinencia).

7. Conclusiones

En este trabajo se presenta una nueva técnica llamada codificación Densa con Post-Etiquetado, que constituye nuestra principal aportación en este trabajo, y que posee sobre las anteriores importantes ventajas:

- El algoritmo de codificación es muy sencillo. Simplemente consiste en una asignación secuencial de códigos. No es necesario construir un árbol de Huffman para garantizar códigos libres de prefijo, por lo que puede utilizar todas las combinaciones posibles de 7 bits en cada byte de un código.
- No es necesario incluir los códigos en el vocabulario del documento comprimido puesto que como se asignan secuencialmente es suficiente guardar el vocabulario ordenado por frecuencia.
- La construcción del código correspondiente a cada palabra se puede realizar rápidamente ya que únicamente se necesita conocer su posición en el vocabulario para calcular el código que le corresponde.

A diferencia de las codificaciones Huffman, la codificación Densa no hace un uso inteligente de las distribuciones de frecuencia de las palabras, sólo tiene en cuenta el orden en base a éstas. Sin embargo, pese a ello consigue mayores ratios de compresión que la codificación Tagged Huffman (que también usa bit de marca). Esto se debe a que al utilizar todas las combinaciones posibles de 7 bits en cada uno de los bytes que componen un código, se reduce el

tamaño medio de los códigos y, por tanto, la longitud media del texto comprimido.

Por último, se presenta una alternativa para la compresión de textos en lenguas romances. Esta técnica se basa en aplicar un algoritmo de *segmentación* que separa las raíces de los sufijos de las palabras, obteniendo así dos vocabularios en el proceso de compresión. Así se consigue una reducción del tamaño del fichero de raíces, lo que permite realizar más eficientemente búsquedas sobre los textos comprimidos.

Referencias

1. Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman, May 1999.
2. T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression.* Prentice Hall, 1990.
3. W. B. Frakes and Ricardo A. Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Englewood Cli.s,NJ 07632,USA, 1992.
4. D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proc. Inst. Radio Eng.*, pages 1098–1101, September 1952. Published as Proc. Inst. Radio Eng., volume 40, number 9.
5. Alistair Mo.at. Word-based text compression. *Software - Practice and Experience*, 19(2):185–198, 1989.
6. David Salomon. *Data Compression: The Complete Reference.* Springer-Verlag, Berlin, Germany /Heidelberg, Germany /London, UK /etc., 2ª edition, 2000.
7. Edleno Silva de Moura, G. Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Fast searching on compressed text allowing errors. In W. Bruce Croft, Alistair Moffat, C. J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proc. of the 21st Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–306, New York City, August 24–28 1998. ACM Press.
8. Edleno Silva de Moura, G. Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems*, 18(2):113–139, April 2000.
9. Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, USA, 1999.
10. George K. Zipf. *Human Behavior and the Principle of Least E.ort*. Addison-Wesley (Reading MA), 1949.
11. Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.
12. Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.
13. Nivio Ziviani, Edleno Silva de Moura, G. Navarro, and Ricardo Baeza-Yates. Compression: A key for next-generation text retrieval systems. *IEEE Computer*, 33(11):37–44, 2000.