Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos (estadísticos)
  - Motivación
  - Conceptos básicos
  - DETDC (Dynamic ETDC)
  - DSCDC (Dynamic (s,c)-DC)
  - Resultados Empíricos
- Compresores dinámicos Ligeros
- Compresión variable-to-variable

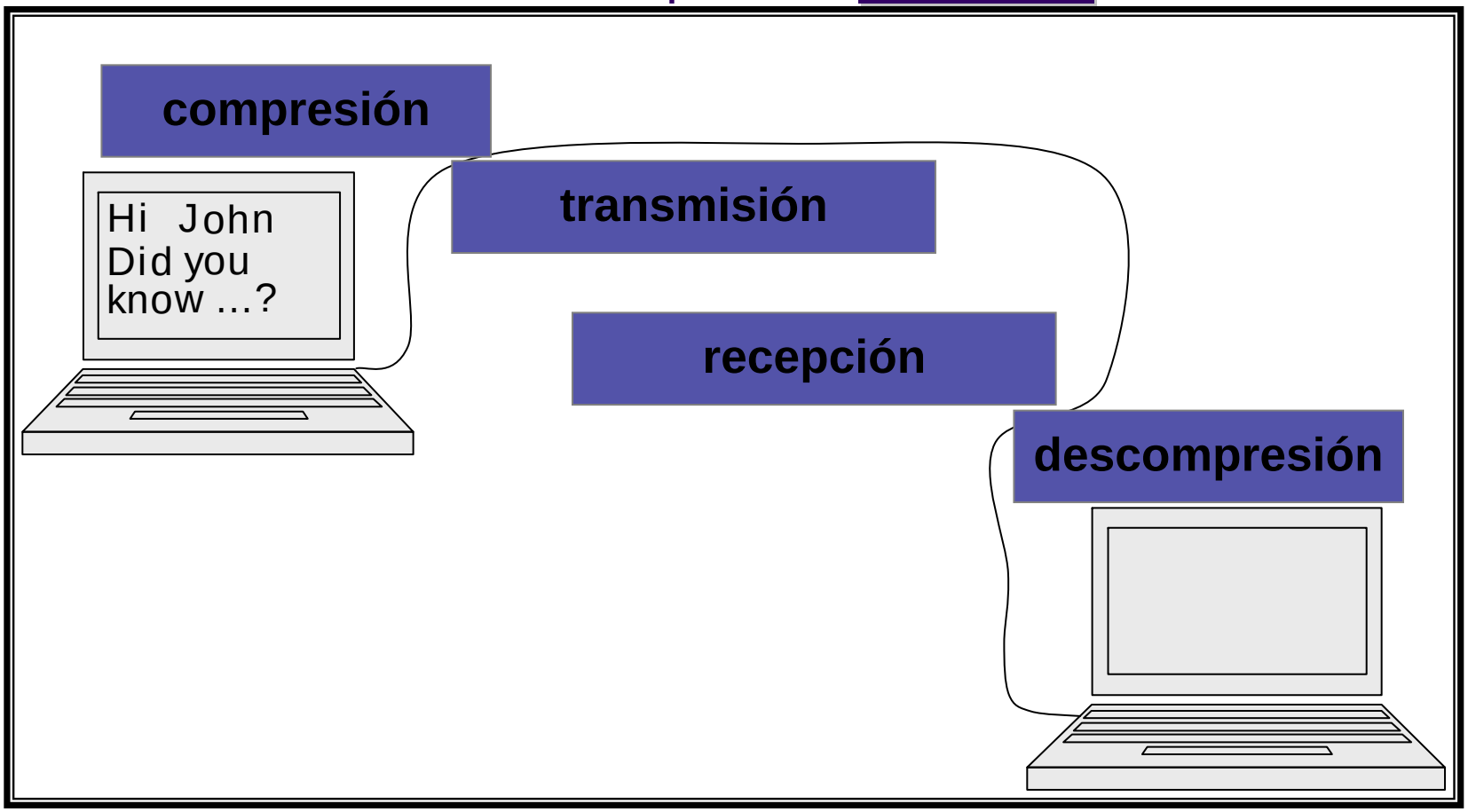
Compresores dinámicos
Motivación General

Tienen diferentes marcos de utilidad:

- **Compresión Semiestática (2 pasadas)**
  - Almacenamiento de Texto
  - Recuperación de Texto
- **Compresión Dinámica (1 pasada)**
  - Transmisión en tiempo real
    - Transmisión de datos y flujos de texto
  - No permite Recuperación de Texto*
    - Asociación variable: palabra ←→ código
  - Paralelismo de:
    - Compresión - transmisión y recepción – descompresión

Compresores dinámicos
Motivación General

- Transmisión usando compresión **dinámica**



ejemplo: servicios mensajería instantánea

Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos
  - Motivación
  - Conceptos básicos
  - DETDC (Dynamic ETDC)
  - DSCDC (Dynamic (s,c)-DC)
  - Resultados Empíricos
- Compresores dinámicos Ligeros
- Compresión variable-to-variable

Compresores dinámicos (estadísticos)

Conceptos básicos

- Las frecuencias son computadas dinámicamente a medida que va apareciendo el texto.

- El emisor y el receptor…
  - Empiezan con un vocabulario vacío.
  - Actualizan su vocabulario durante el proceso.
  - Ambos procesos son simétricos.

Compresores dinámicos (estadísticos)

Estructura básica simétrica

- Estructura general compresión simétrica dinámica

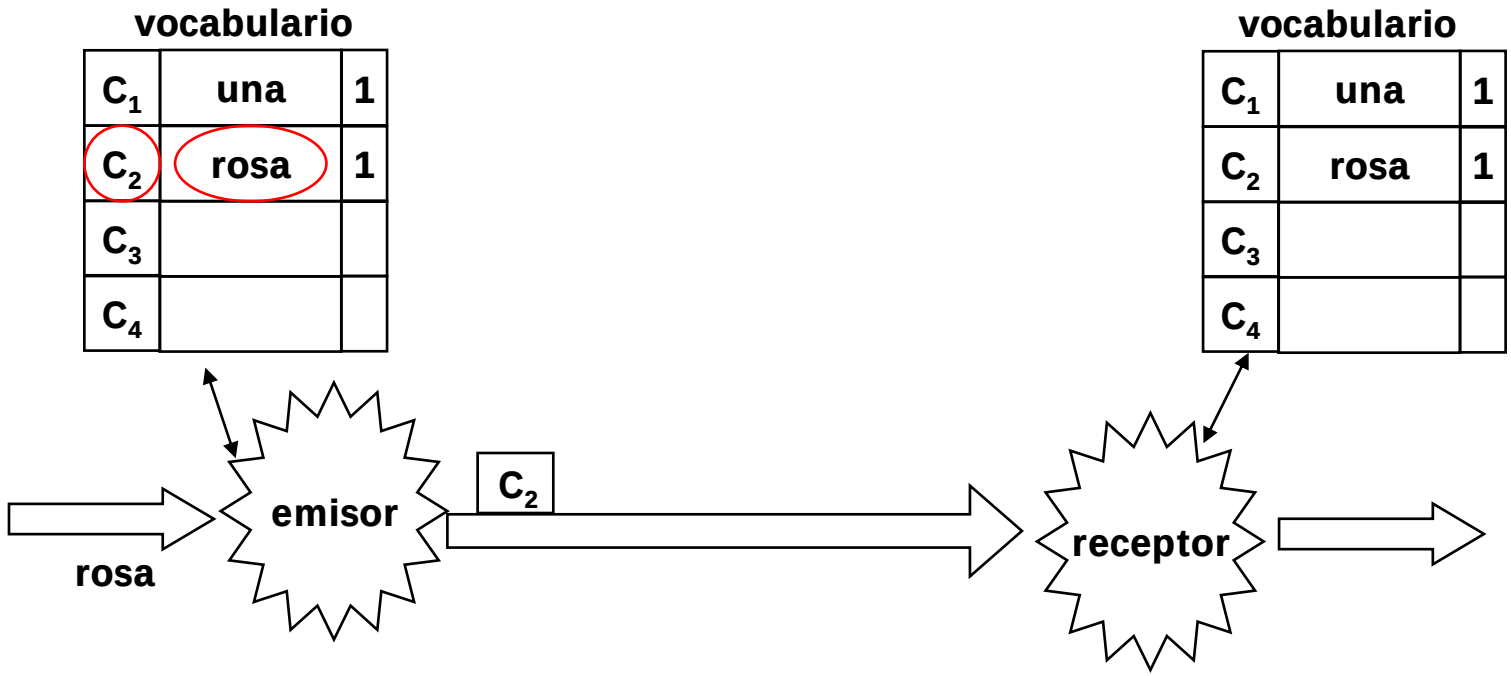
```
Sender ( )
(1)  Vocabulary ← {C_zeroNode};
(2)  Initialize CodeBook;
(3)  while (true) do
(4)       read s from the text;
(5)       if s ∉ Vocabulary then
(6)            send C_zeroNode;
(7)            send s in plain form;
(8)            Vocabulary ← Vocabulary ∪ {s};
(9)            f(s) ← 1;
(10)      else
(11)           send CodeBook(s);
(12)           f(s) ← f(s) + 1;
(13)      Update CodeBook;
```

```
Receiver ( )
(1)  Vocabulary ← {C_zeroNode};
(2)  Initialize CodeBook;
(3)  while (true) do
(4)       receive C;
(5)       if C = C_zeroNode then
(6)            receive s in plain form;
(7)            Vocabulary ← Vocabulary ∪ {s};
(8)            f(s) ← 1;
(9)       else
(10)           s ← CodeBook^-1(C);
(11)           f(s) ← f(s) + 1;
(12)      output s;
(13)      Update CodeBook;
```

Compresores dinámicos (estadísticos)

Conceptos básicos

Ejemplo: una rosa rosa es una rosa

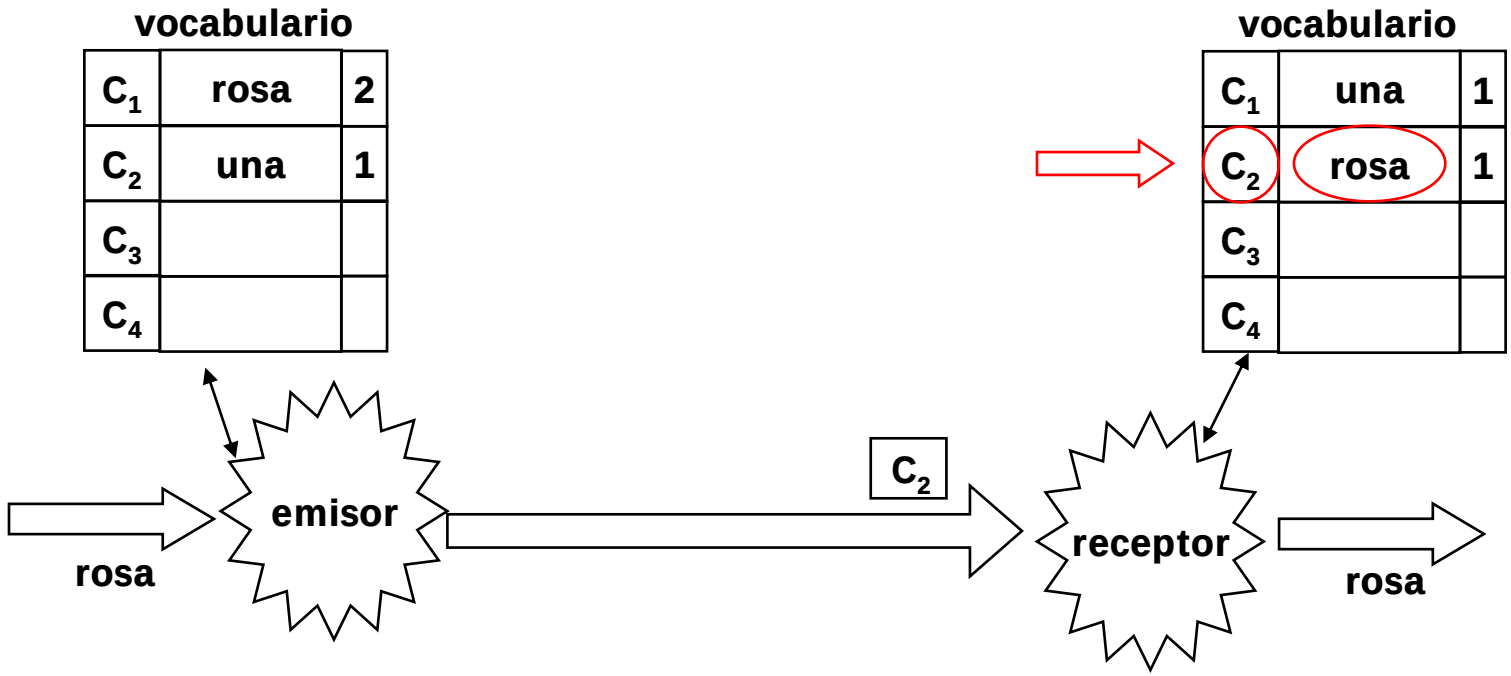


Buscar “rosa” en el vocabulario del emisor y emitir código C2

Compresores dinámicos (estadísticos)

Conceptos básicos

Ejemplo: una rosa rosa es una rosa



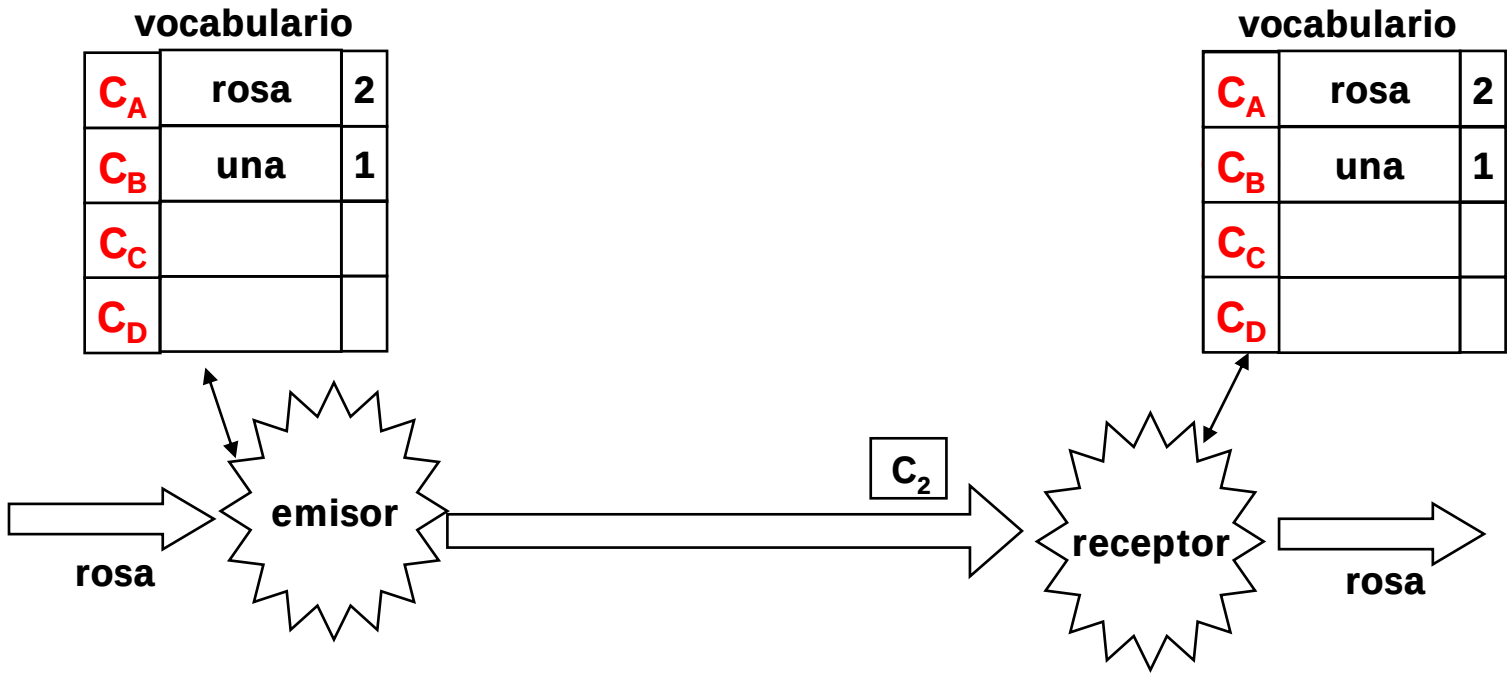
Emisor actualiza vocabulario (intercambio rosa←→una, e incrementa frec. A “rosa”)

Receptor decodifica C2 → es una palabra existente → escribe “rosa”

Compresores dinámicos (estadísticos)

Conceptos básicos

Ejemplo: una rosa rosa es una rosa

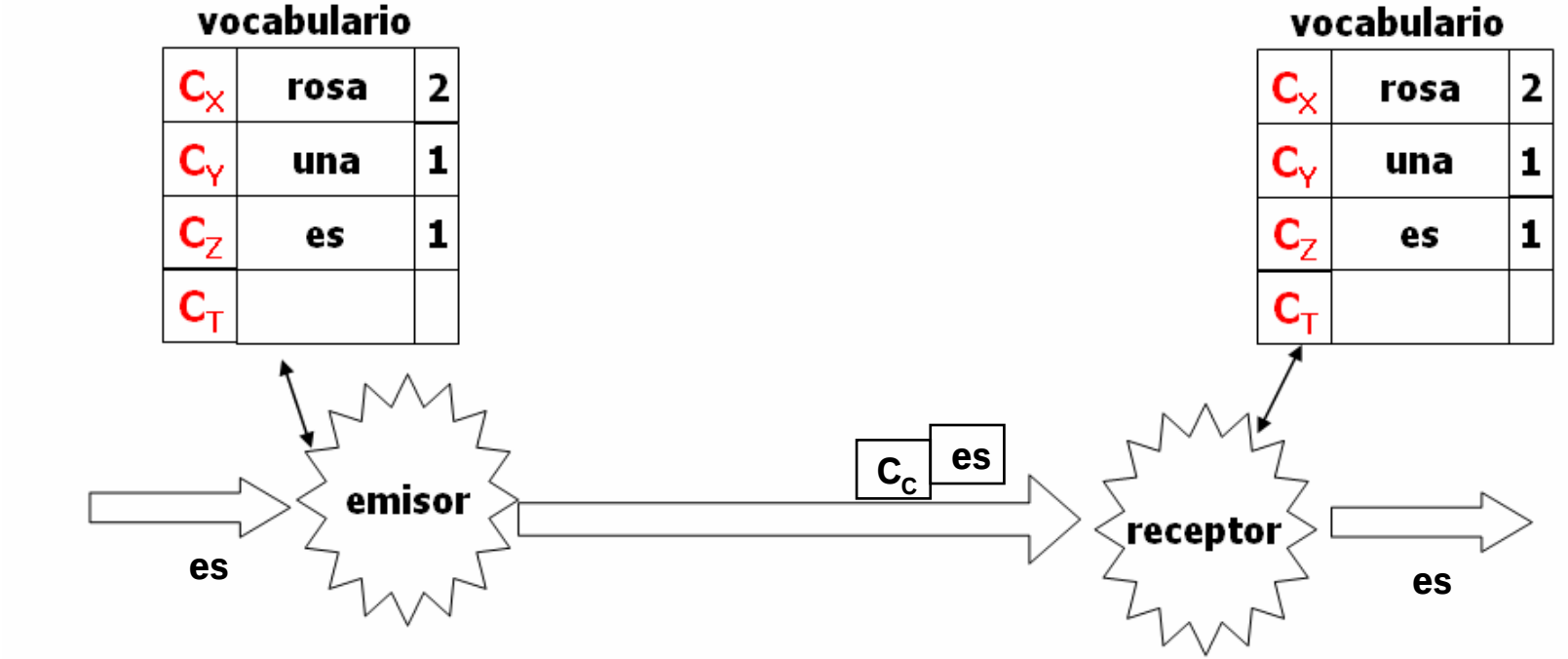


Receptor actualiza vocabulario (intercambio rosa←→ una)

La codificación asociada a cada palabra del vocabulario "puede" haber cambiado (ej. Huffman)

Compresores dinámicos
Conceptos básicos

Ejemplo: una rosa rosa es una rosa



"es" = nueva palabra → añadir al final → enviar código escape (Cc) seguido de "es"
→ Por último la codificación puede haber cambiado de nuevo…

Compresores dinámicos

Conceptos básicos: Huffman (refs)

[Fal73] N Faller. An adaptive system for data compression. In *Record of the 7th Asilomar Conference on Circuits, Systems, and Computers*, pages 593–597, 1973.

[Gal78] R.G Gallager. Variations on a theme by Huffman. *IEEE Trans. on Inf. Theory*, 24(6):668–674, 1978.

[Knu85] Donald E. Knuth. Dynamic Huffman coding. *Journal of Algorithms*, 6(2):163–180, June 1985.

[Vit87] J.S. Vitter. Design and analysis of dynamic Huffman codes. *Journal of the ACM (JACM)*, 34(4):825–845, 1987.

[Vit89] J.S. Vitter. Algorithm 673: dynamic Huffman coding. *ACM Transactions on Mathematical Software (TOMS)*, 15(2):158–167, 1989.

Compresores dinámicos
Conceptos básicos: Huffman

- Huffman dinámico basado en caracteres:
  - [Fal73, Gal78], **[Knu85] algoritmo FGK**, [Vit87]

Definition 8.1 *A binary code tree has the sibling property if each node (except the root) has a sibling and if all nodes can be listed in decreasing weight order, with each node adjacent to its sibling.*

Gallager also proved that a binary prefix code is a Huffman code iff the code tree has the sibling property.

Using the *sibling property*, the main achievement of *FGK* is to ensure that the tree can be updated by doing only a constant amount of work per node in the path from the affected leaf to the tree root. Calling $l(s_i)$ the path length from the leaf of source symbol s_i to the root, and $f(s_i)$ its frequency, the overall cost of the algorithm *FGK* is $\sum f(s_i)l(s_i)$, which is exactly the length of the compressed text measured in number of target symbols.

Compresores dinámicos
Conceptos básicos: Huffman

- Huffman dinámico basado en caracteres:
  - [Fal73, Gal78], **[Knu85] algoritmo FGK**, [Vit87]

- Nuestra implementación: Dynamic Plain Huffman (DPH)
  - Basado en palabras
    - Uso de una tabla *hash* para almacenar las palabras (y encontrarlas eficientemente)
  - Orientado al Byte
    - Peor ratio de compresión pero mejor velocidad de descompresión
    - Árbol 256-ario de Huffman → Más complejo que el orientado a bit
  - Árbol Huffman bien formado durante compresión/decompr.
    - Actualizar el árbol es una tarea compleja O(log h)
  - Ratio de compresión en torno a 30-35 %.

Compresores dinámicos
Conceptos básicos: Huffman

- Nuestra implementación: Dynamic Plain Huffman (DPH)

Definition 9.1 *A 2^b-ary code tree has the sibling property if each node (except the root and the zeroNode and its siblings), has $2^b - 1$ siblings, and if all nodes can be listed in decreasing frequency order, with each node adjacent to its siblings.*

Recall that the *zeroNode* is a special zero-frequency node that is used in dynamic codes to introduce those symbols that have not yet appeared.

Property 9.1 *To achieve a dynamic word-based, byte-oriented Huffman code it is only needed to maintain two conditions:*

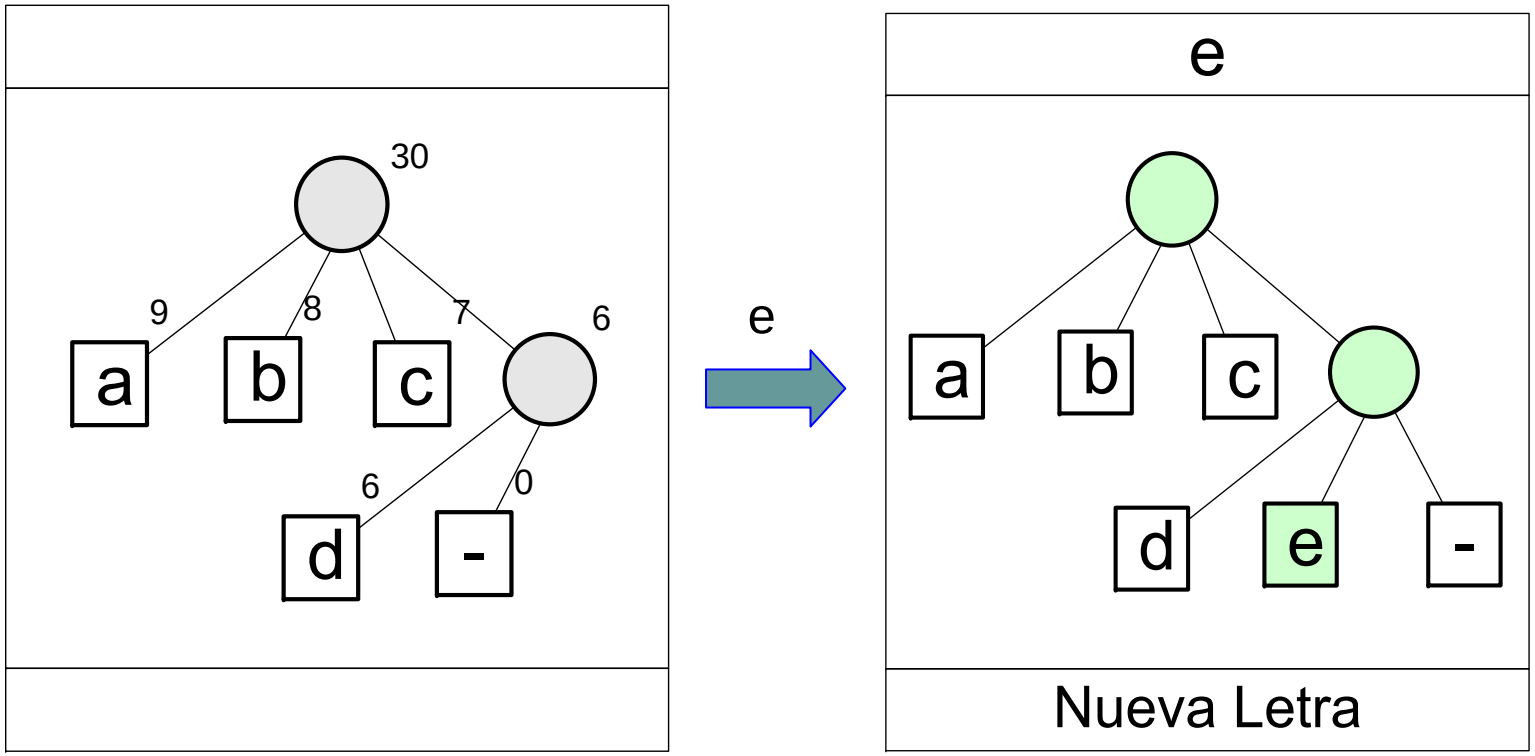
- *All nodes of the Huffman tree (both internal and leaf nodes) remain sorted by frequency. In this ranking the root is the most frequent node, and the zeroNode is the least frequent node.*
- *All the siblings of a node remain adjacent.*

En definitiva

- Nodes in the top levels of the tree precede nodes in the bottom levels.
- For a given level of the tree, nodes are ranked left-to-right. Therefore the left-most node of a level precedes all nodes in that level.

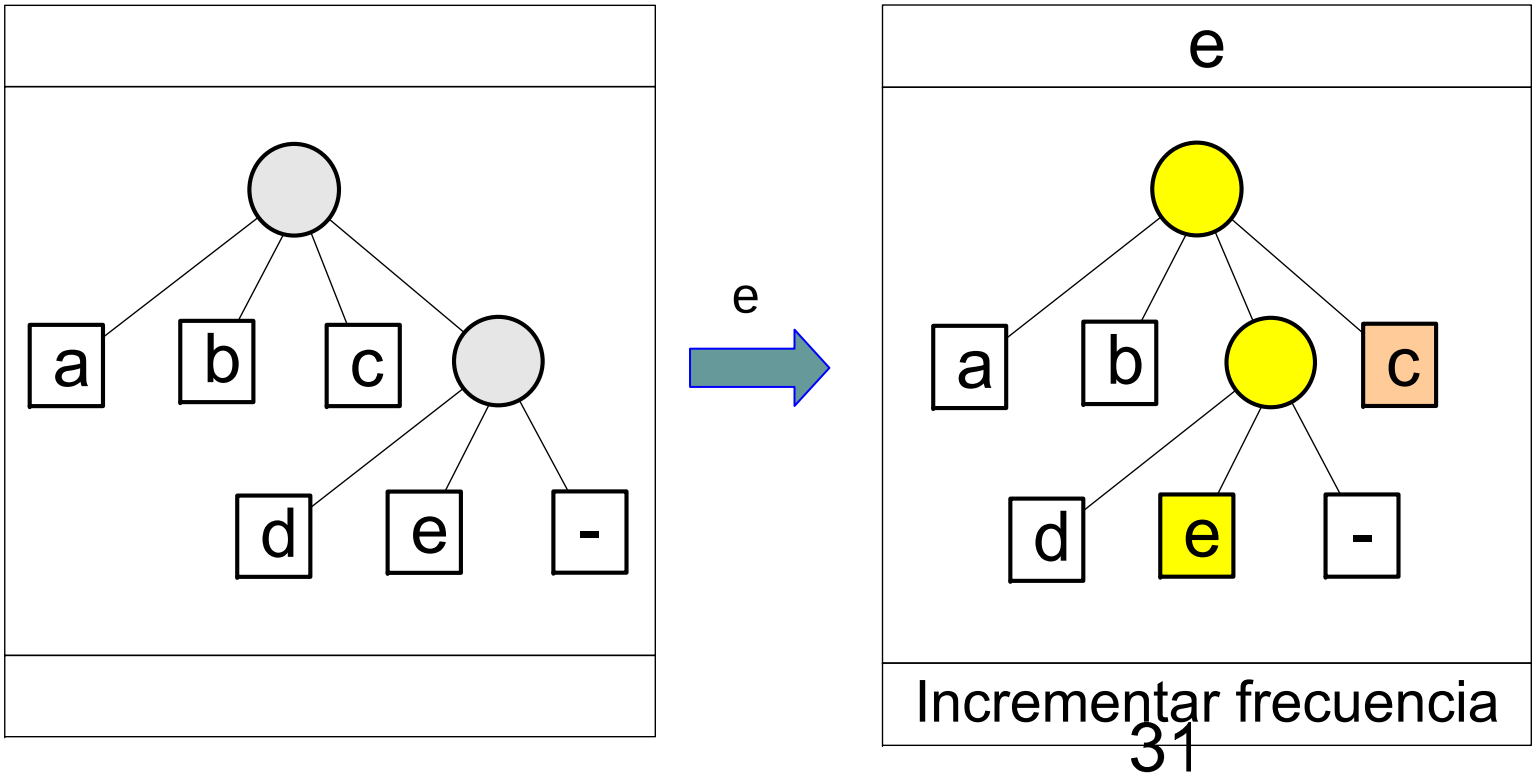
Compresores dinámicos
Conceptos básicos: Huffman

- Ejemplo: b=2 → hasta $2^2 = 4$ hijos por nodo

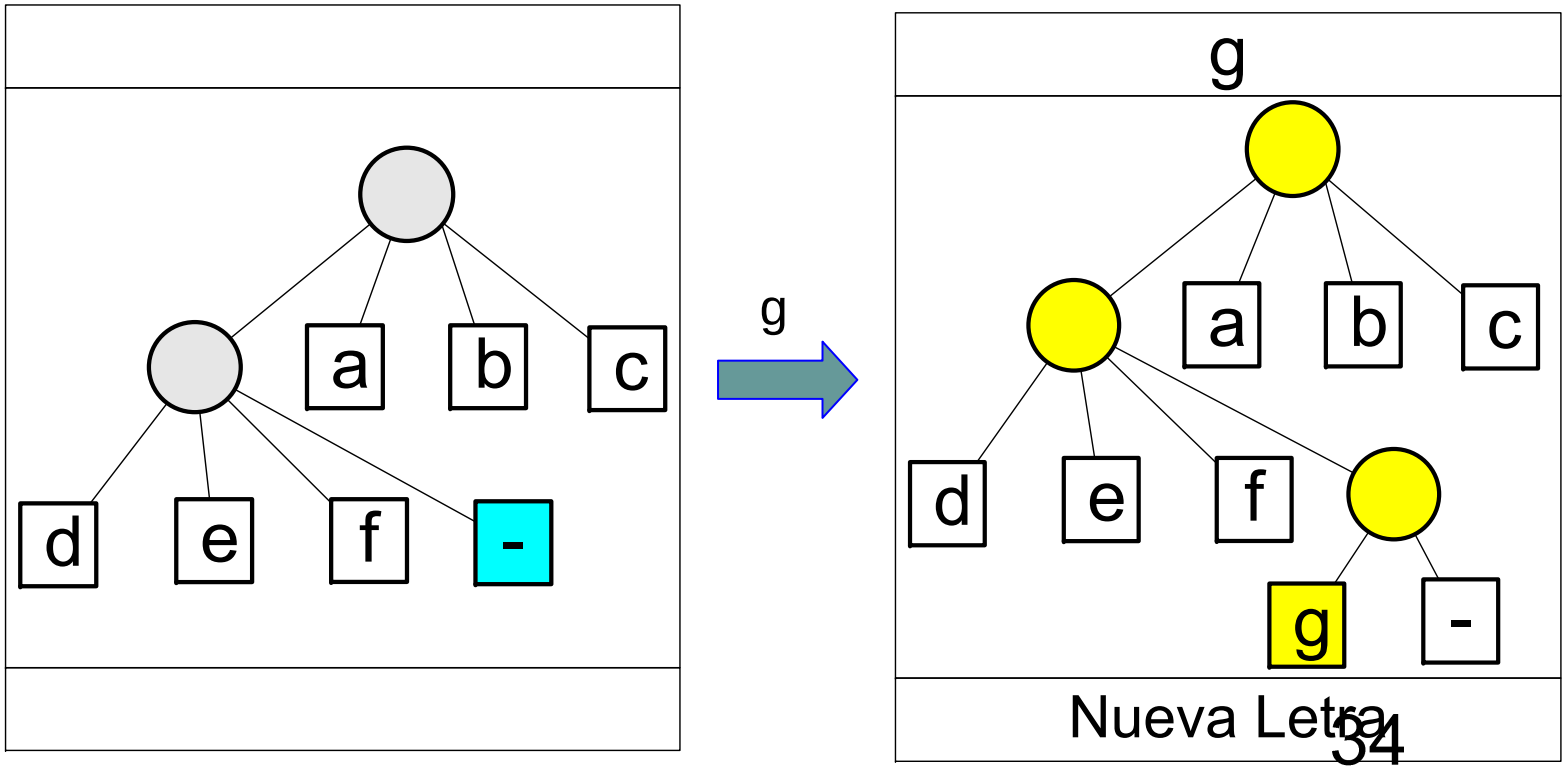


Compresores dinámicos

Conceptos básicos: Huffman

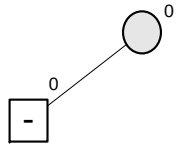
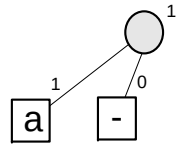
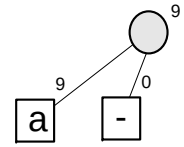
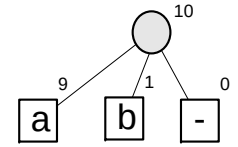


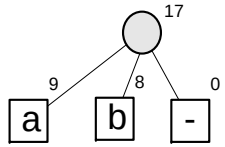
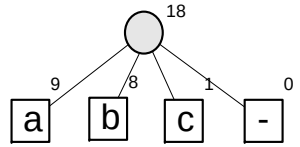
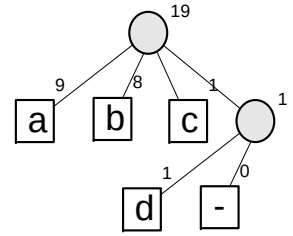
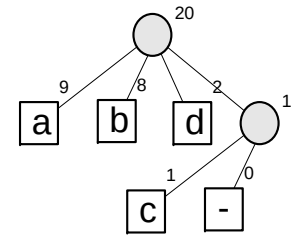
Compresores dinámicos
Conceptos básicos: Huffman



Compresores dinámicos
Conceptos básicos: Huffman

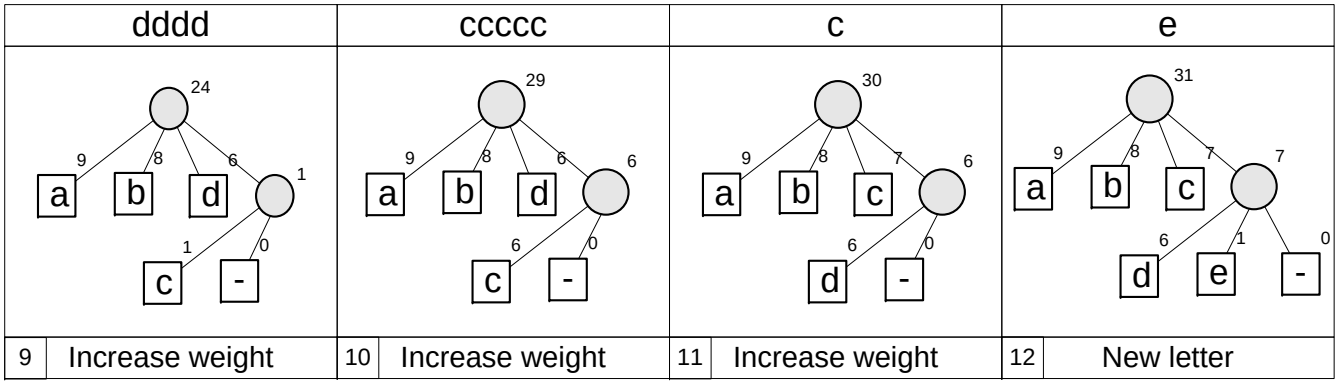
- Ejemplo: Huffman dinámico D-ario:
  texto transmitido: aaaaaaaaabbbbbbbcddddddccccceeefg

-	a	aaaaaaaa	b
			
1 Initial state	2 New letter	3 Increase weight	4 New letter

bbbbbbb	c	d	d
			
5 Increase weight	6 New letter	7 New letter	8 Increase weight

Compresores dinámicos
Conceptos básicos: Huffman

- Ejemplo: Huffman dinámico D-ario:
  texto transmitido: aaaaaaaaabbbbbbbbcddddddccccceeefg



Compresores dinámicos
Conceptos básicos: Huffman

- Coste de compresión O(nº símbolos de salida)
  - Orientado a byte → O(nº bytes de salida)
  - Orientado a bit → O(nº bits de salida)

- La altura del árbol es:
  - **Menor** en la aproximación orientada a byte

Menos propagaciones de cambios hacia arriba en el árbol.

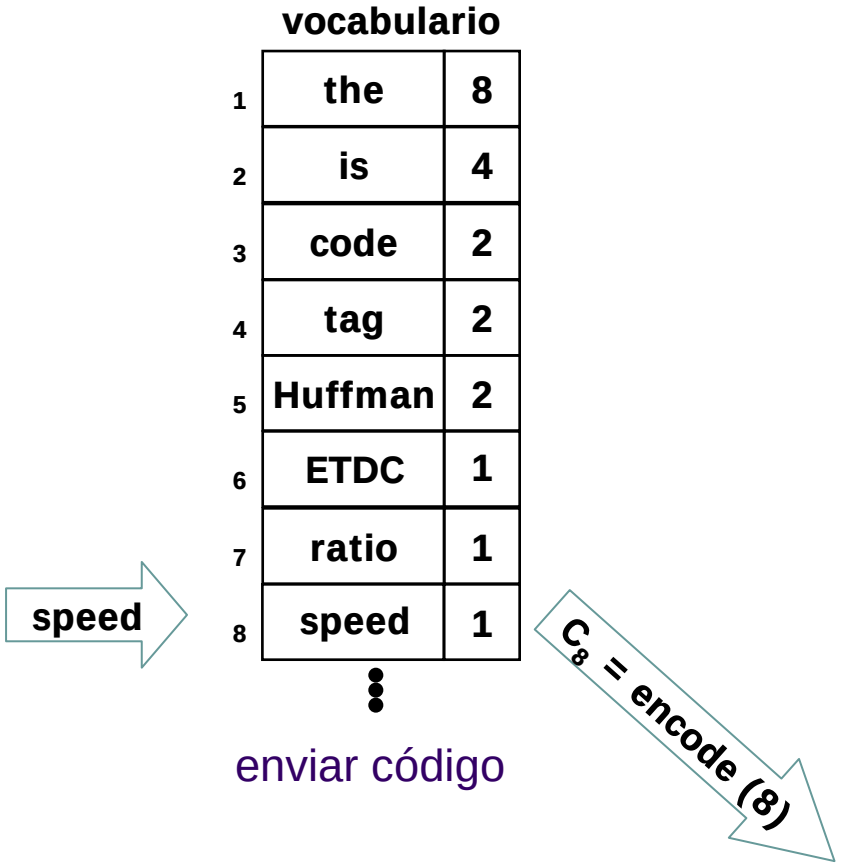
Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos
  - Motivación
  - Conceptos básicos
  - DETDC (Dynamic ETDC)
  - DSCDC (Dynamic (s,c)-DC)
  - Resultados Empíricos
- Compresores dinámicos Ligeros
- Compresión variable-to-variable

Compresores dinámicos densos
Dynamic End-Tagged Dense Code

- Usa los algoritmos de codificación y decodificación directa
  - Requisitos: mantener el vocabulario ordenado por frecuencia

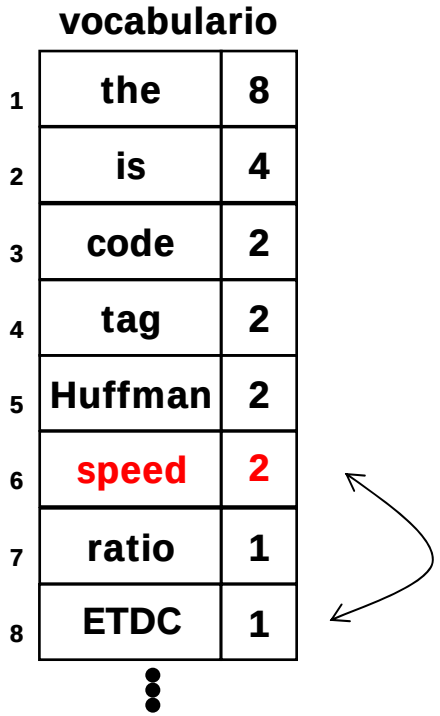
- Ejemplo



Compresores dinámicos densos
Dynamic End-Tagged Dense Code

- Usa los algoritmos de codificación y decodificación directa
  - Requisitos: mantener el vocabulario ordenado por frecuencia

- Ejemplo



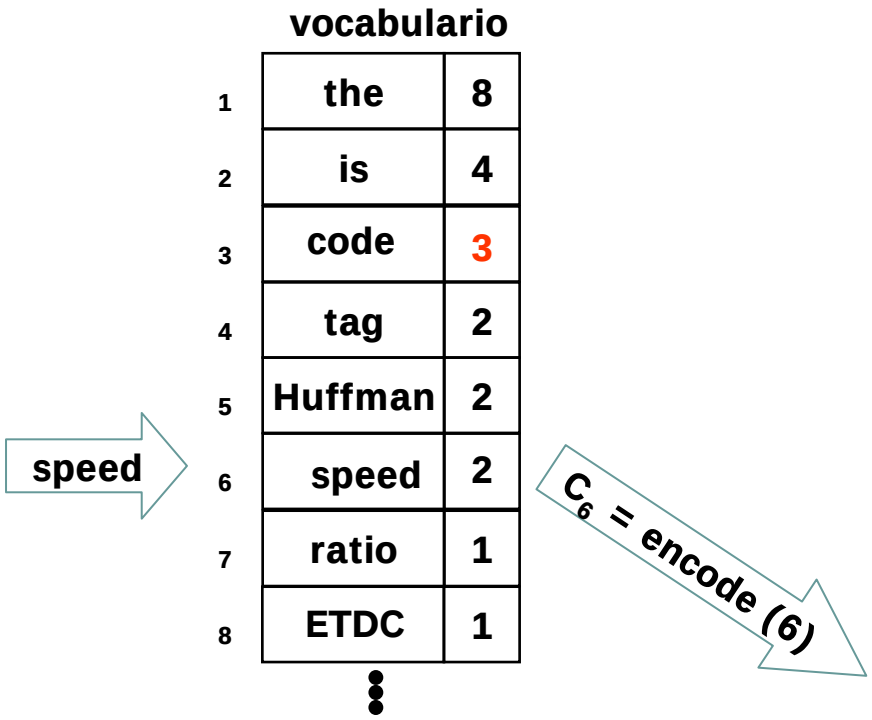
intercambio con top

Incrementar frecuencia

Compresores dinámicos densos
Dynamic End-Tagged Dense Code

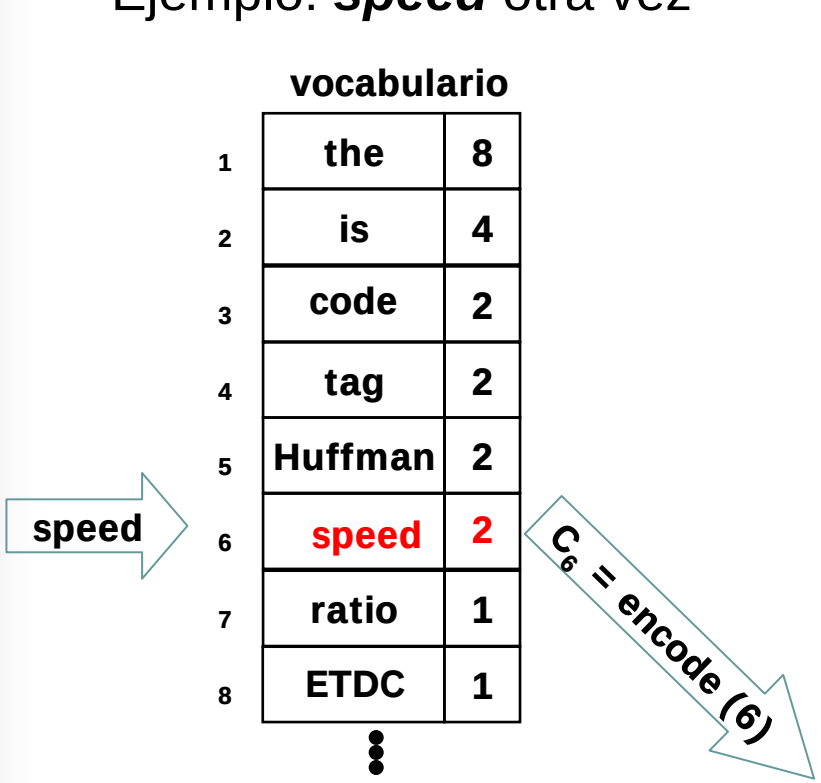
- Usa los algoritmos de codificación y decodificación directa
  - Requisitos: mantener el vocabulario ordenado por frecuencia.

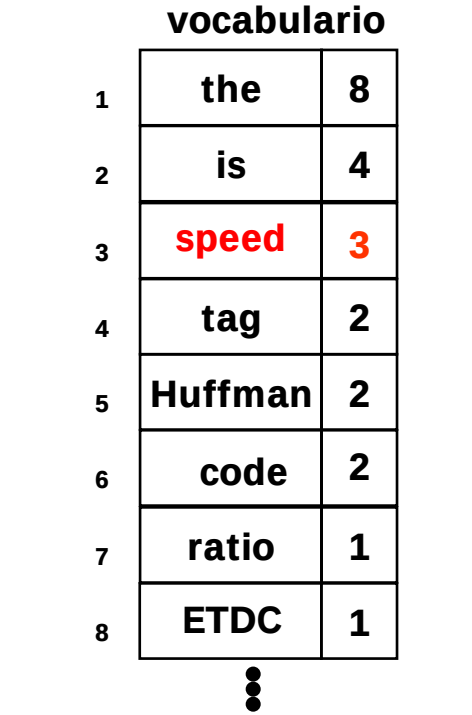
- Ejemplo: “llega ***speed***”



Compresores dinámicos densos
Dynamic End-Tagged Dense Code

- Usa los algoritmos de codificación y decodificación directa
  - Requisitos: mantener el vocabulario ordenado por frecuencia.
- Ejemplo: ***speed*** otra vez

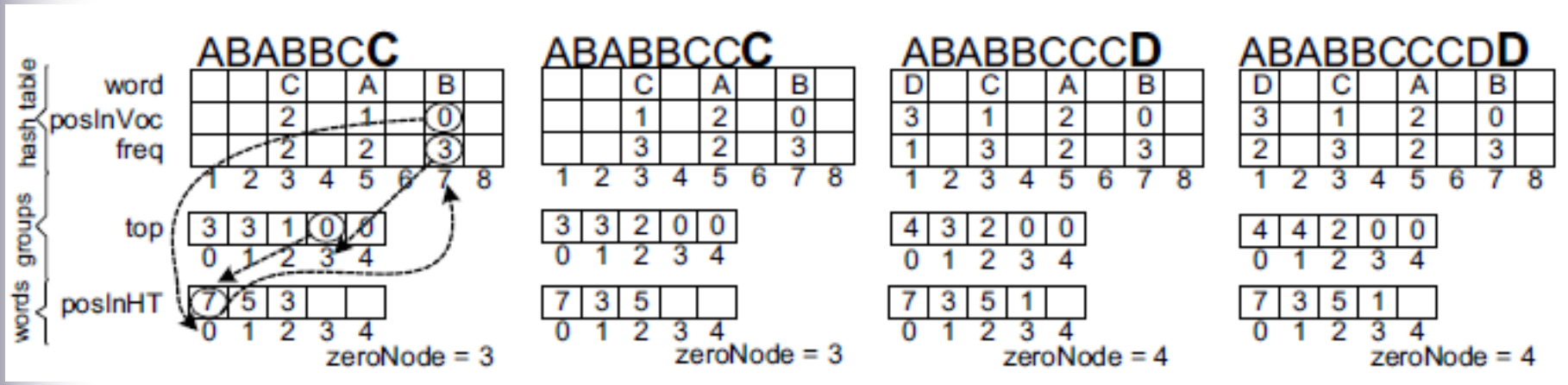




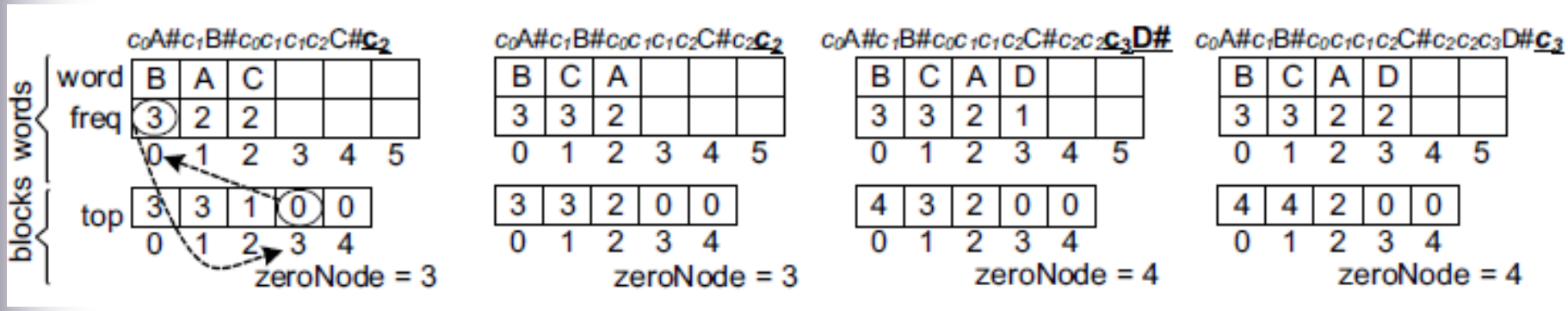
Compresores dinámicos densos

Dynamic End-Tagged Dense Code

Transmission of words C, C, D and D having transmitted ABABBC earlier.



Reception of c2, c2, c3D# and c3 having received c0A#c1B#c0c1c1c2C# previously.



Compresores dinámicos densos

Dynamic End-Tagged Dense Code

Sender main algorithm ( )

```
(1)  Initialize vocabulary structures, zeroNode ← 0;
(2)  for i ← 1 to n − 1 do top[i] ← 0;
(3)  for each new symbol i do
(4)       read s_i from text;
(5)       p ← f_hash(s_i);
(6)       if word[p] = Null (s_i ∉ word) then
(7)            i ← zeroNode;
(8)            send (encode(i));
(9)            send s_i in plain form;
(10)      else
(11)           i ← posInVoc[p];
(12)           send (encode(i));
(13)      update();
```

Sender update ( )

```
(1)  if i =zeroNode then // new word
(2)       word[p] ← s_i;
(3)       freq[p] ← 0;
(4)       posInVoc[p] ← zeroNode ;
(5)       posInHT[zeroNode] ← p;
(6)       zeroNode ← zeroNode + 1;
(7)  f ← freq[p];
(8)  freq[p] ← freq[p] + 1;
(9)  j ← top[f];
(10) h ← posInHT[j];
(11) swap (posInHT[i], posInHT[j]);
(12) posInVoc[p] ← j;
(13) posInVoc[h] ← i;
(14) top[f] ← j + 1;
```

Receiver main algorithm ( )

```
(1)  Initialize vocabulary structures, zeroNode ← 0;
(2)  for i ← 1 to n − 1 do top[i] ← 0;
(3)  for each new codeword C_i do
(4)       i ← decode(C_i);
(5)       if i =zeroNode then
(6)            receive s_i in plain form;
(7)            output s_i;
(8)       else
(9)            output word[i];
(10)      update();
```

Receiver update ( )

```
(1)  if i =zeroNode then // new word
(2)       word[i] ← s_i;
(3)       freq[i] ← 0;
(4)       zeroNode ← zeroNode + 1;
(5)  f ← freq[i];
(6)  freq[i] ← freq[i] + 1;
(7)  j ← top[f];
(8)  swap (freq[i], freq[j]);
(9)  swap (word[i], word[j]);
(10) top[f] ← j + 1;
```

Compresores dinámicos densos
Dynamic End-Tagged Dense Code

- DETDC es mucho más simple que Huffman Dinámico:
  - Mantener las palabras ordenadas Vs mantener un árbol de Huffman.
  - Ejecutar un único intercambio por código

    O(1) Vs O(altura árbol Huffman)
  - Coste de la compresión

    O(nº códigos de salida) Vs O(nº bytes de salida)

Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos
  - Motivación
  - Conceptos básicos
  - DETDC (Dynamic ETDC)
  - DSCDC (Dynamic (s,c)-DC)
  - Resultados Empíricos

- Compresores dinámicos Ligeros
- Compresión variable-to-variable

Compresores dinámicos densos
Dynamic (s,c)-Dense Code

- Generalización del ETDC Dinámico
  - Codificación y decodificación *directa*,
  - Mantener el vocabulario ordenado y,
  - Mantener un valor óptimo de **s**

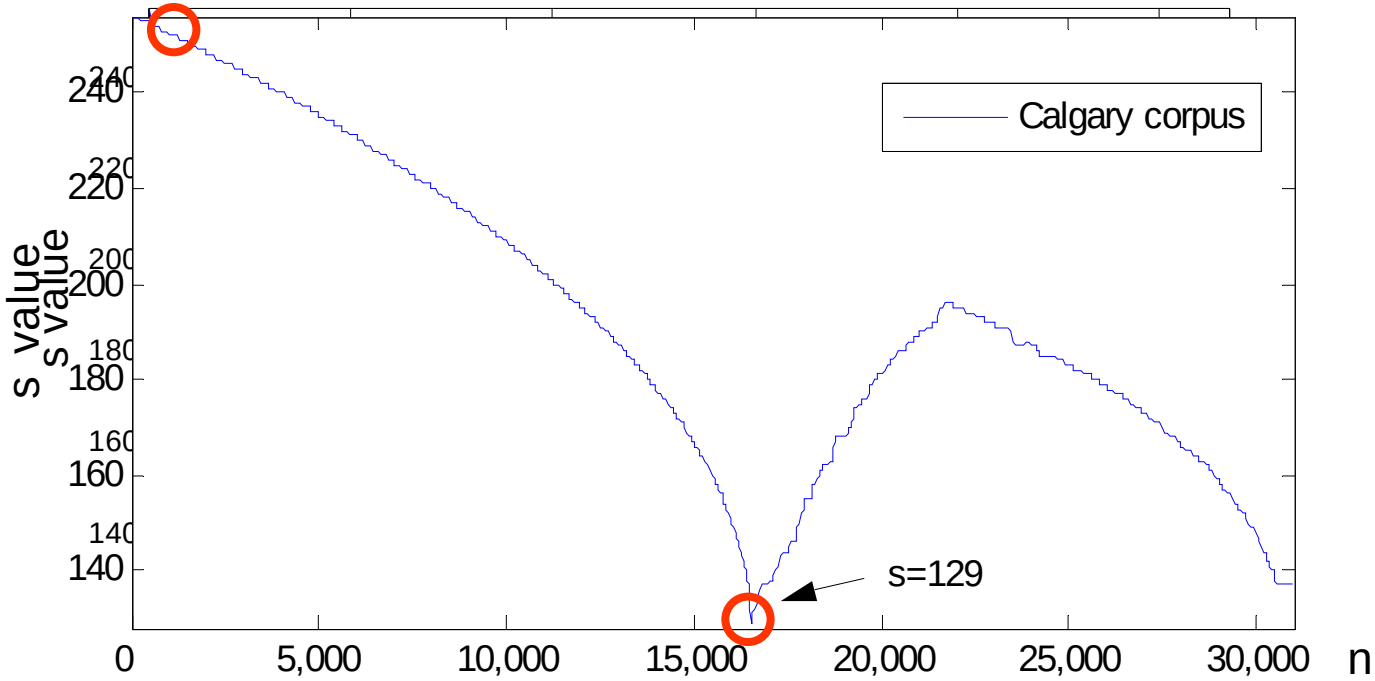
- **Distintas aproximaciones para ajustar valor de s:**
  - Contar bytes
    - Con historia, Sin historia, Umbral

- Ref.:

Brisaboa, N. R.; Fariña, A.; Navarro, G.; Paramá, J. R. "*New Adaptive Compressors for Natural Language Text".* En Software, Practice & Experience(38)-2. pp. 1429-1450, 2008.

Compresores dinámicos densos
Dynamic (s,c)-Dense Code

- Evolución del valor de s



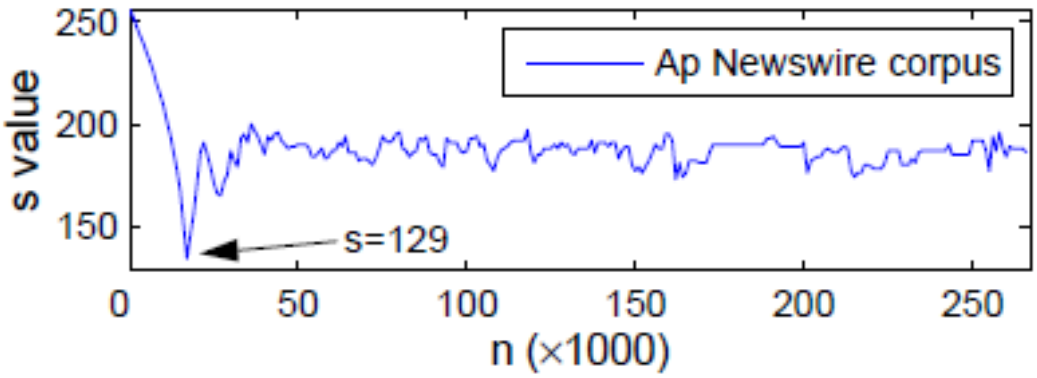
- s = 256 mientras n < 256 → con código de **un byte**
- s = 255 mientras n < 510 → usando códigos de **dos bytes**
- …
- s cae hasta s=129 (para n = 16,512) → códigos de **tres bytes** son usados (s = 129 maximiza s + sc = # 1- or 2-byte códigos)

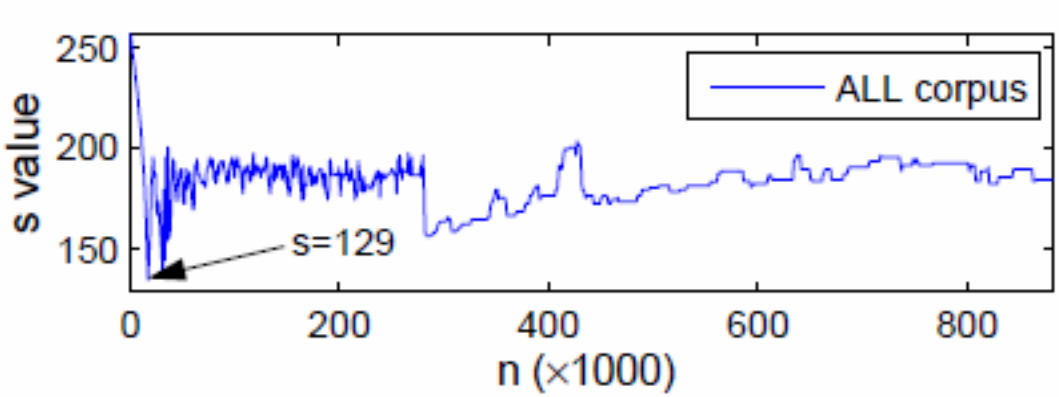
…

- Entonces … el valor s crece para mejorar la compresión en palabras con alta frecuencia

Compresores dinámicos densos
Dynamic (s,c)-Dense Code

- Evolución Otros casos reales.





Compresores dinámicos densos
Dynamic (s,c)-Dense Code

- Después de procesar una palabra …
  - s puede incrementarse en 1: s← s +1
  - s puede decrementarse en 1: s← s -1

- Usamos 3 variables ***prev***, ***curr***, y ***next*** para acumular el tamaño del texto comprimido para esos 3 valores de ***s***.

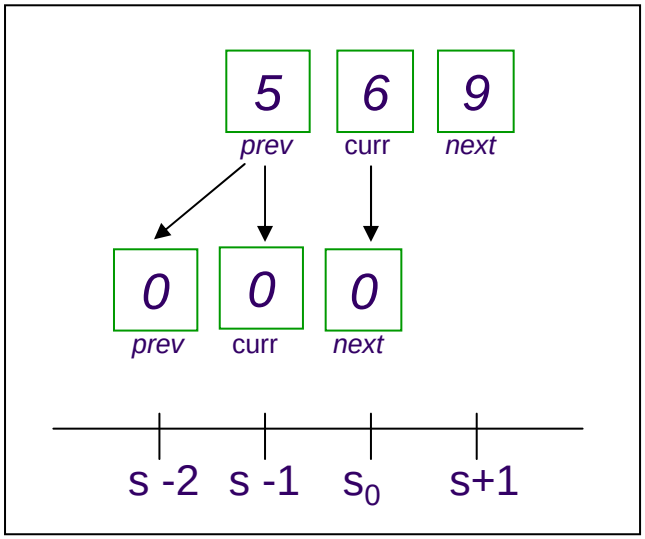
prev	Tamaño texto Compr. sí #stoppers = s-1
Curr	Tamaño texto Compr. sí #stoppers = s_0
next	Tamaño texto Compr. sí #stoppers = s+1

- If *prev* < *curr* then s_0← s-1
- If *next* < *curr* then s_0← s+1

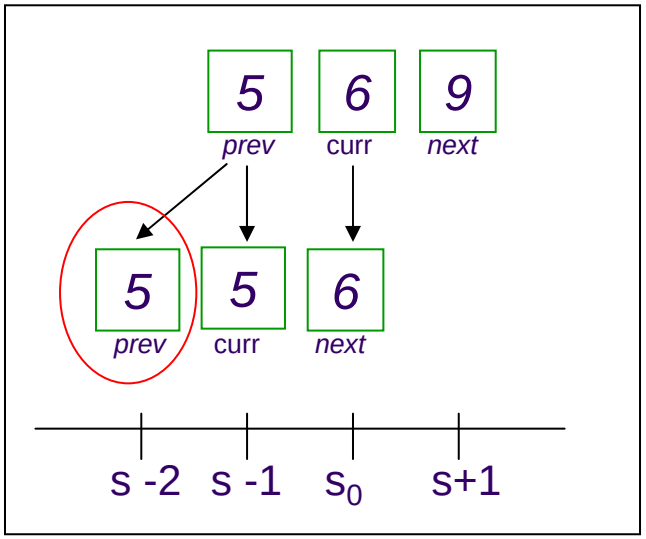
Compresores dinámicos densos
Dynamic (s,c)-Dense Code

- Supongamos $prev < s_0$ y por tanto s← s-1

Pero…. *¿como* reinicializar *prev?*



Descarta Historial
Inicialización a 0



Guarda Historial (next ←s_0+1)
Inicialización a S_0

- ¿Uso de un umbral? → menos cambios → pérdida de compresión

Compresores dinámicos densos

Dynamic (s,c)-Dense Code

CheckAndUpdateS Algorithm (s, c, i)

```
(1)  prev ← countBytes (s − 1, i);
(2)  s0 ← countBytes (s, i);
(3)  if prev < s0 then
(4)       s ← s − 1; //s is decreased
(5)       c ← c + 1;
(6)       next ← s0;
(7)       s0 ← prev;
(8)  else
(9)       next ← countBytes (s + 1, i);
(10)      if next < s0 then
(11)           s ← s + 1; //s is increased
(12)           c ← c − 1;
(13)           prev ← s0;
(14)           s0 ← next;
```

countBytes Algorithm (s_i, i_{pos})

```
(1)  k ← 1;
(2)  last ← s_i;
(3)  pow ← s_i;
(4)  c_i ← 256 − s_i;
(5)  while last ≤ i_pos do
(6)       pow ← pow × c_i;
(7)       last ← last + pow;
(8)       k ← k + 1;
```

Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos
  - Motivación
  - Conceptos básicos
  - DETDC (Dynamic ETDC)
  - DSCDC (Dynamic (s,c)-DC)
  - Resultados Empíricos

- Compresores dinámicos Ligeros
- Compresión variable-to-variable

Compresores dinámicos densos

Resultados Empíricos

- Técnicas comparadas:
  - Versiones Semiestáticas y Dinámicas de:
    - Plain Huffman
    - (s,c)-Dense Code
    - End-Tagged Dense Code
  - Compresor aritmético basado en palabras y orientado a bit
  - Gzip
  - Bzip2

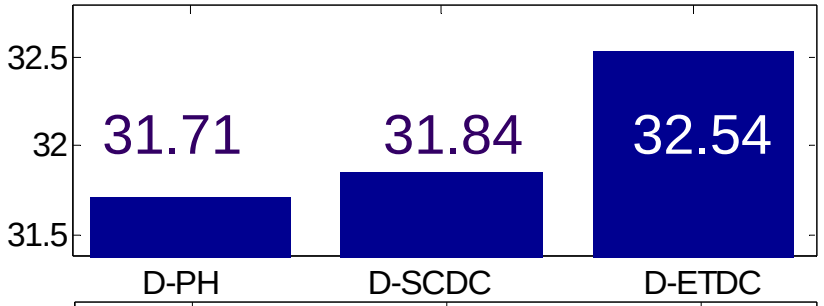
- Comparaciones en:
  - Ratio de compresión
  - Velocidad de compresión
  - Velocidad de descompresión

Compresores dinámicos densos

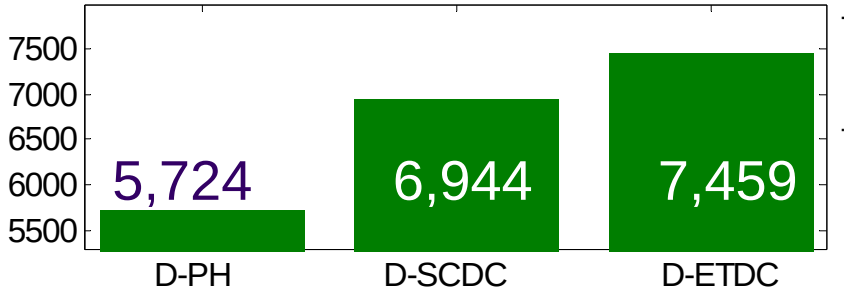
Resultados Empíricos: dinámicos

FT_ALL

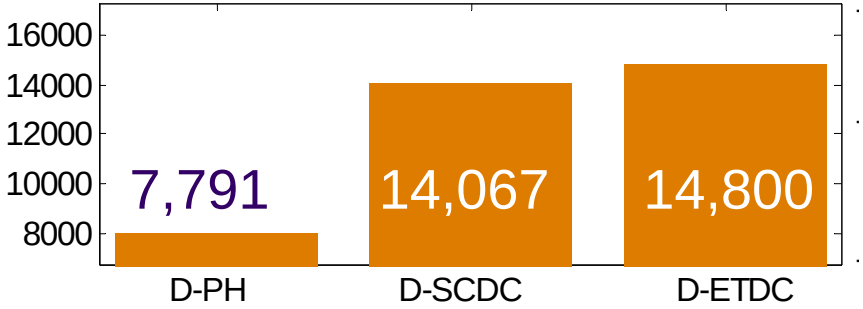
Ratio de compresión (%)



Veloc. compresión (Kbytes/sg)



Veloc. Descompresión (Kbytes/sg)



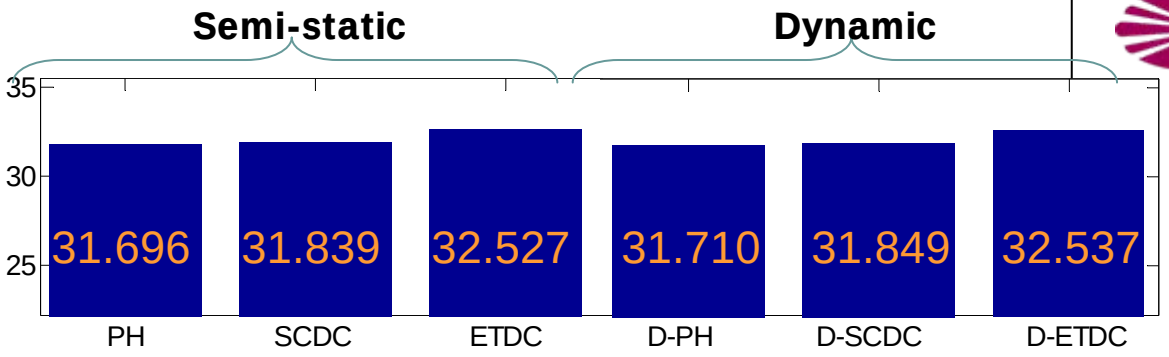
Huffman <u>mejor compresión,</u> pero <u>más lento</u> que DETDC

DSCDC <u>5 % más lento</u> que DETDC, pero con <u>mejor compresión</u>

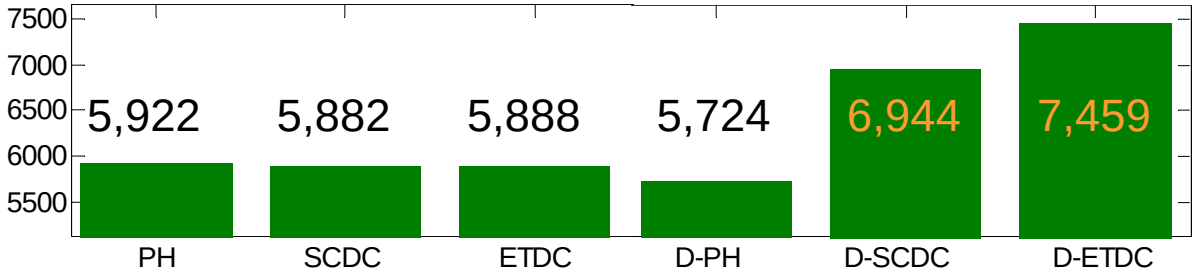
Compresores dinámicos densos
Resultados Empíricos: dinámicos

FT_ALL

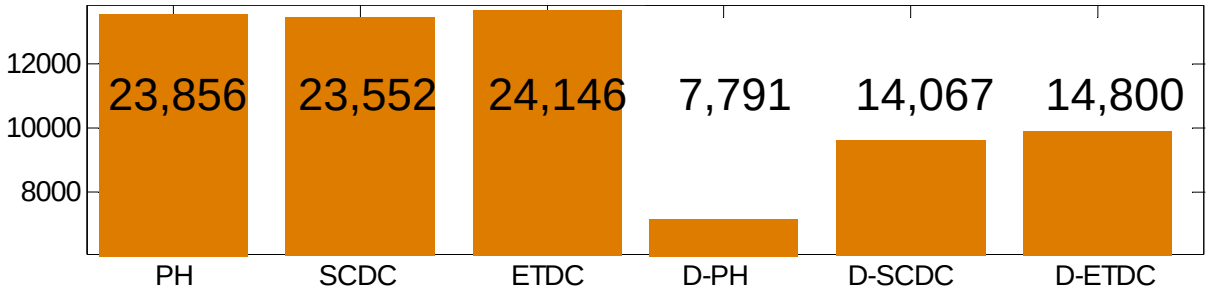
Ratio de compresión (%)



Veloc. compresión (Kbytes/sec)



Veloc descompresión (Kbytes/sec)



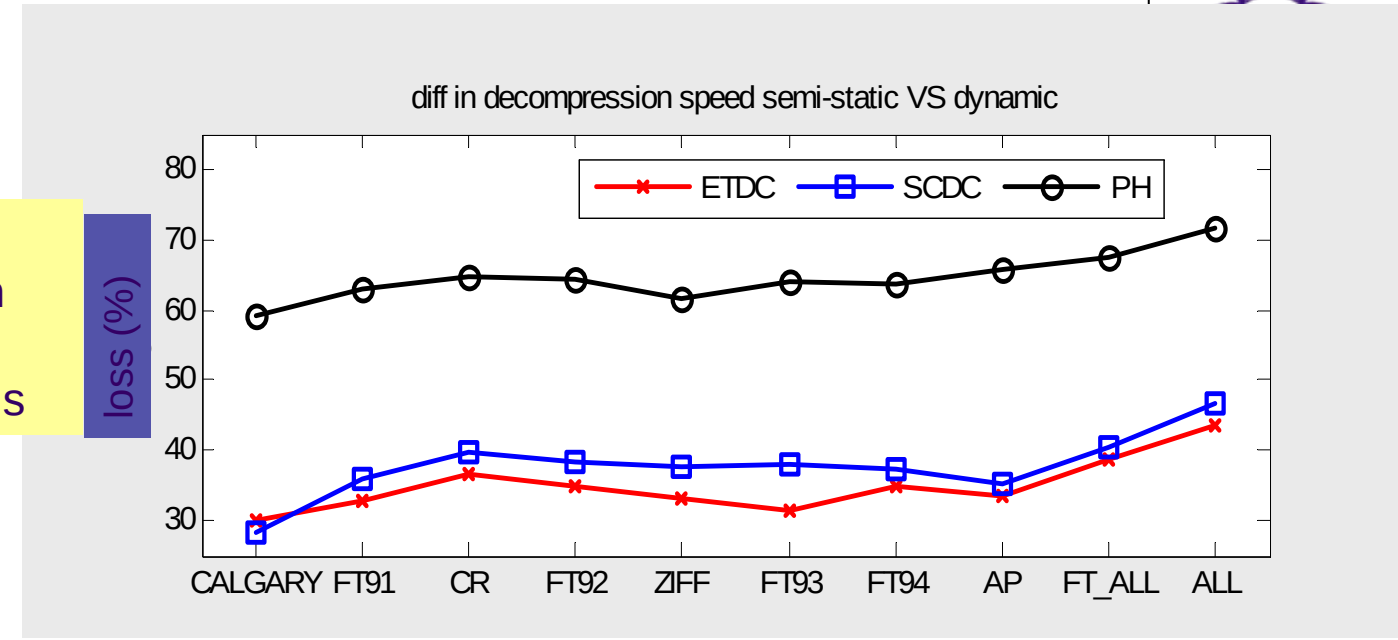
Ratio compresión: **dinamismo empeora imperceptiblemente el ratio**

Veloc compresión: **1-pass más rápidos (excepto DPH en textos largos)**

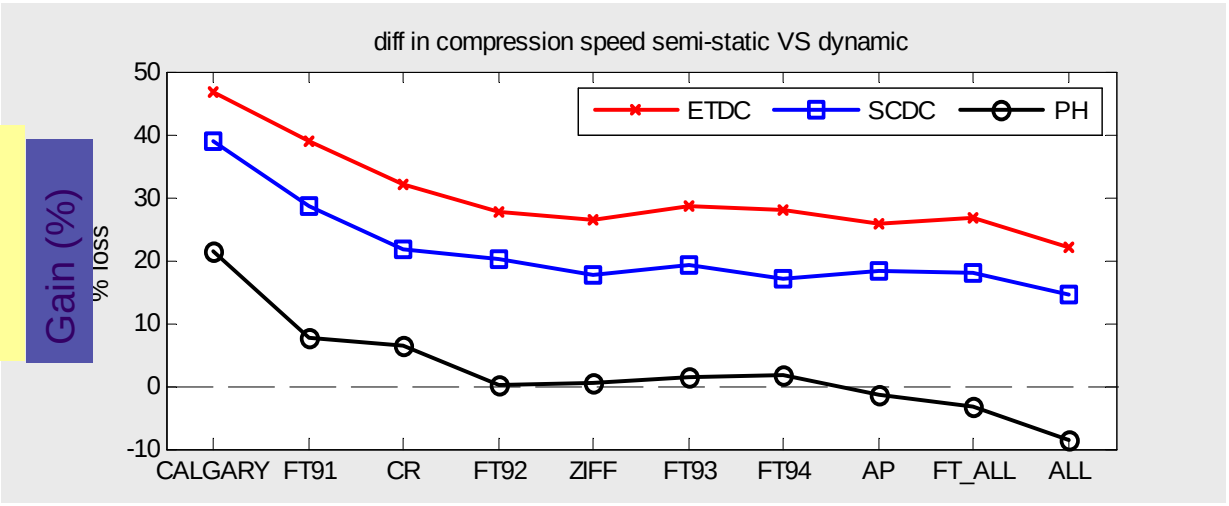
Veloc descompresión: 2-pass mucho más rápidos (no hay *updates*)

Compresores dinámicos densos
Resultados Empíricos: dinámicos

Loss in **decompression speed** of the dynamic methods

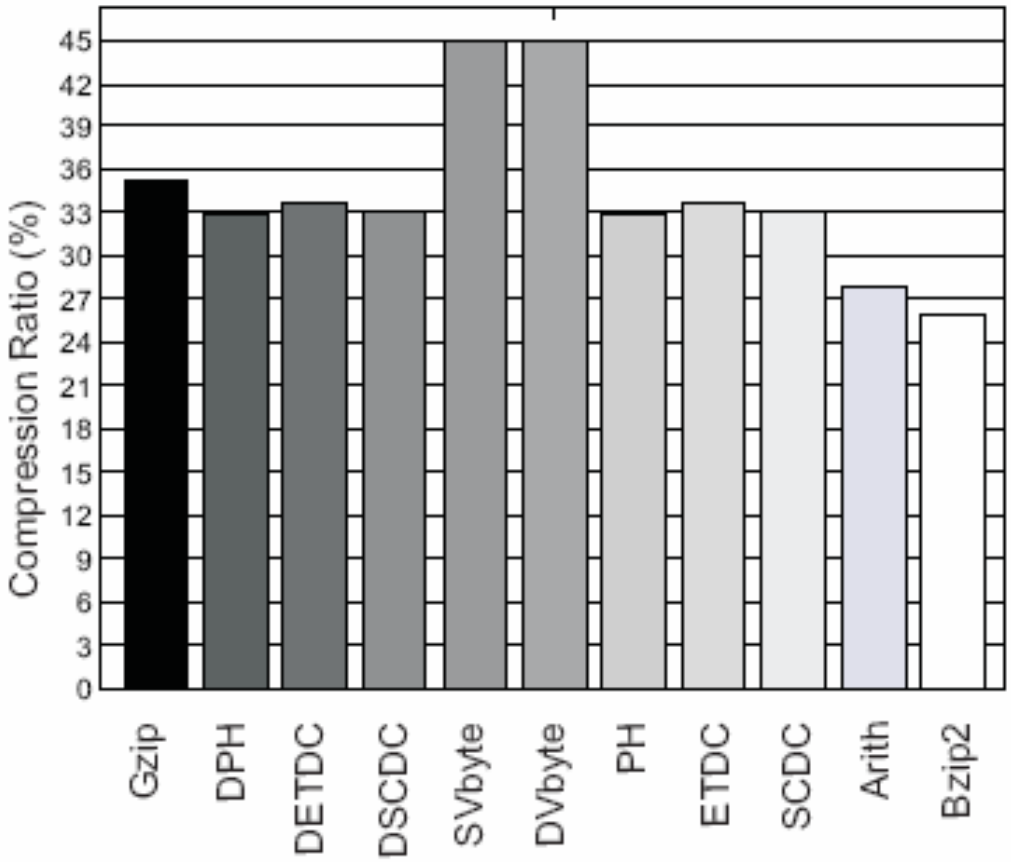


Gain in **compression speed** of the dynamic methods



Compresores dinámicos densos

Resultados Empíricos: Ratio

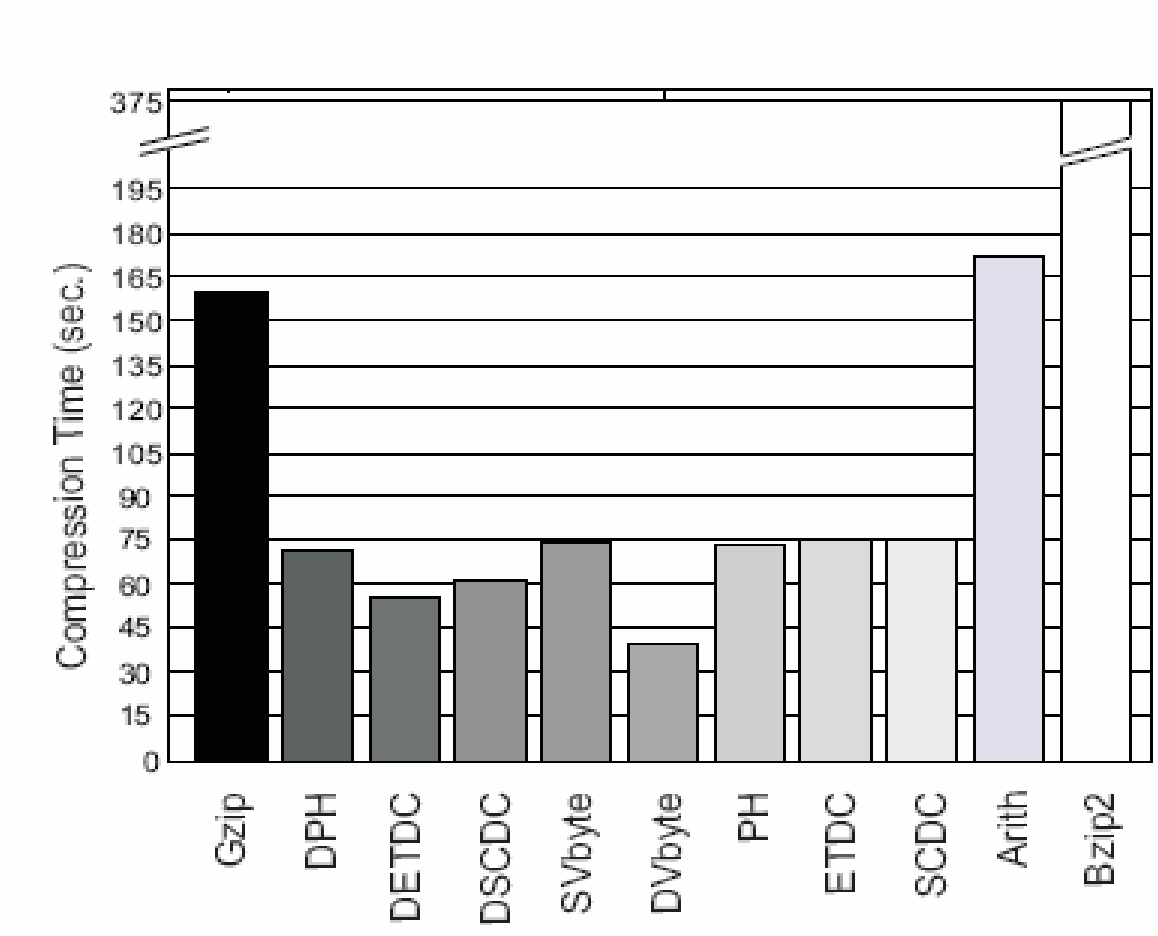


Corpus	Size KB	Gzip	DPH	DETDC	DSCDC	DVbyte	PH	ETDC	SCDC	SVbyte	Arith	Bzip2
ALL	1,055,391	35.09	32.85	33.66	33.03	45.00	32.83	33.66	33.02	45.00	27.98	25.98

Ratio de compresión: ***bzip2*** **el mejor.** **Cód. Densos mejor que** ***gzip***

Compresores dinámicos densos

Resultados Empíricos: Tiempo compresión

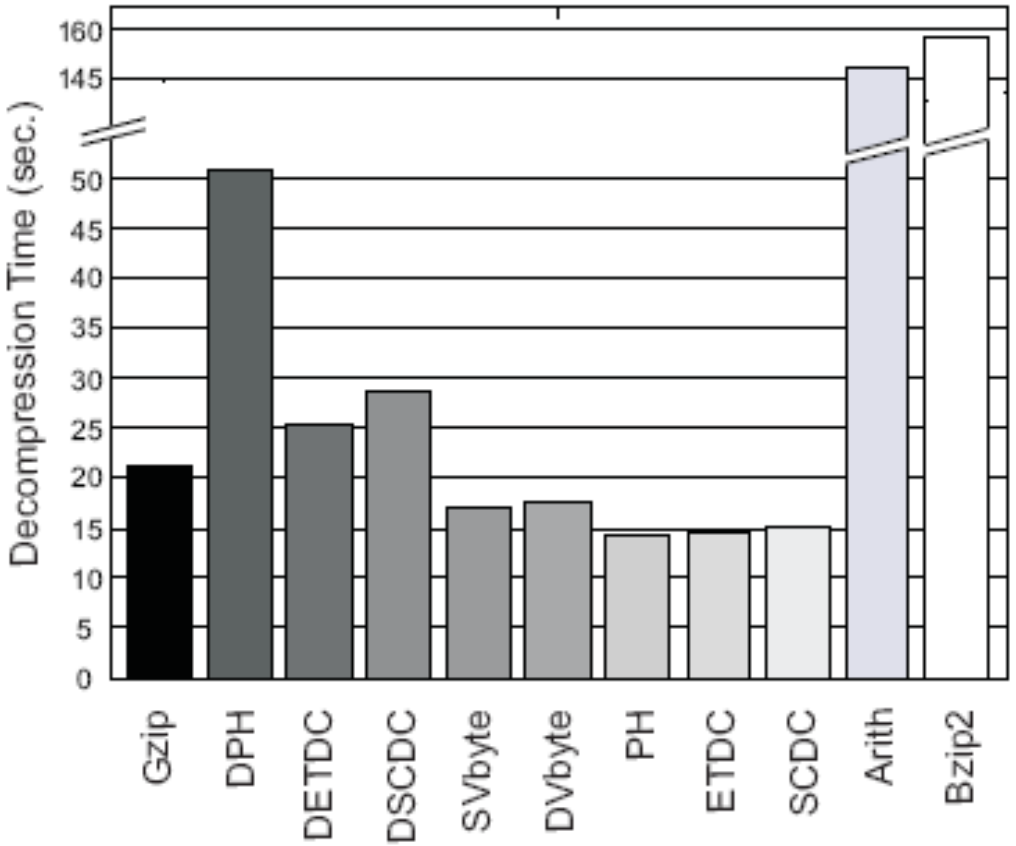


Corpus	Gzip	DPH	DETDC	DSCDC	DVbyte	PH	ETDC	SCDC	SVbyte	Arith	Bzip2
ALL	160.01	71.54	55.31	61.35	39.55	72.77	75.58	75.20	73.78	171.56	375.55

Velocidad compresión: **Compresores densos son más rápidos (excp Dvbyte)**

Compresores dinámicos densos

Resultados Empíricos: Tiempo descompresión



Corpus	Gzip	DPH	DETDC	DSCDC	DVbyte	PH	ETDC	SCDC	SVbyte	Arith	Bzip2
ALL	21.05	50.82	25.27	28.62	17.45	14.26	14.56	15.08	16.96	147.15	156.18

Velocidad descompresión: *gzip* es el más rápido (excep semistáticos).

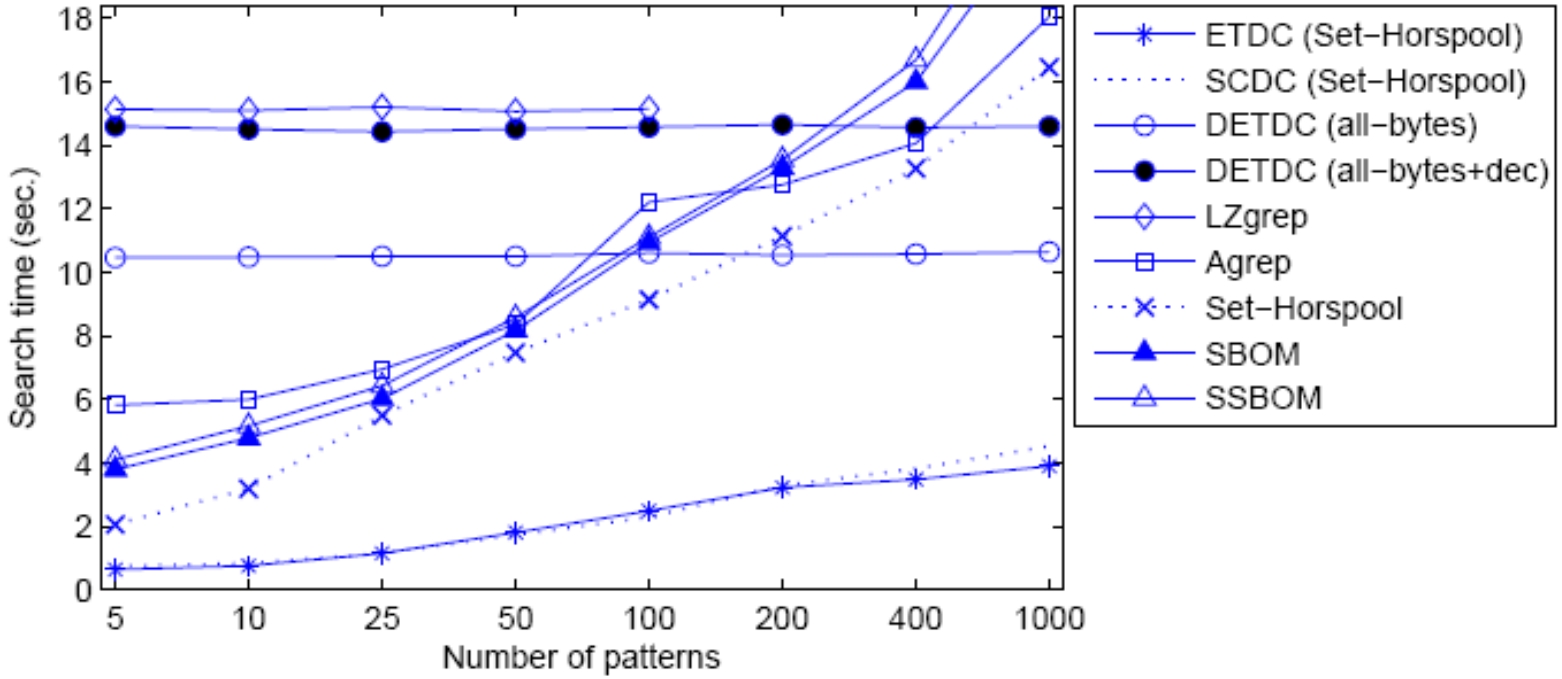
DETDC y DSCDC los siguientes

Compresores dinámicos densos

Resultados Empíricos: Tiempo búsqueda

Es posible buscar ?

Sí → simular descompresión… pero sin necesidad de escribir códigos !



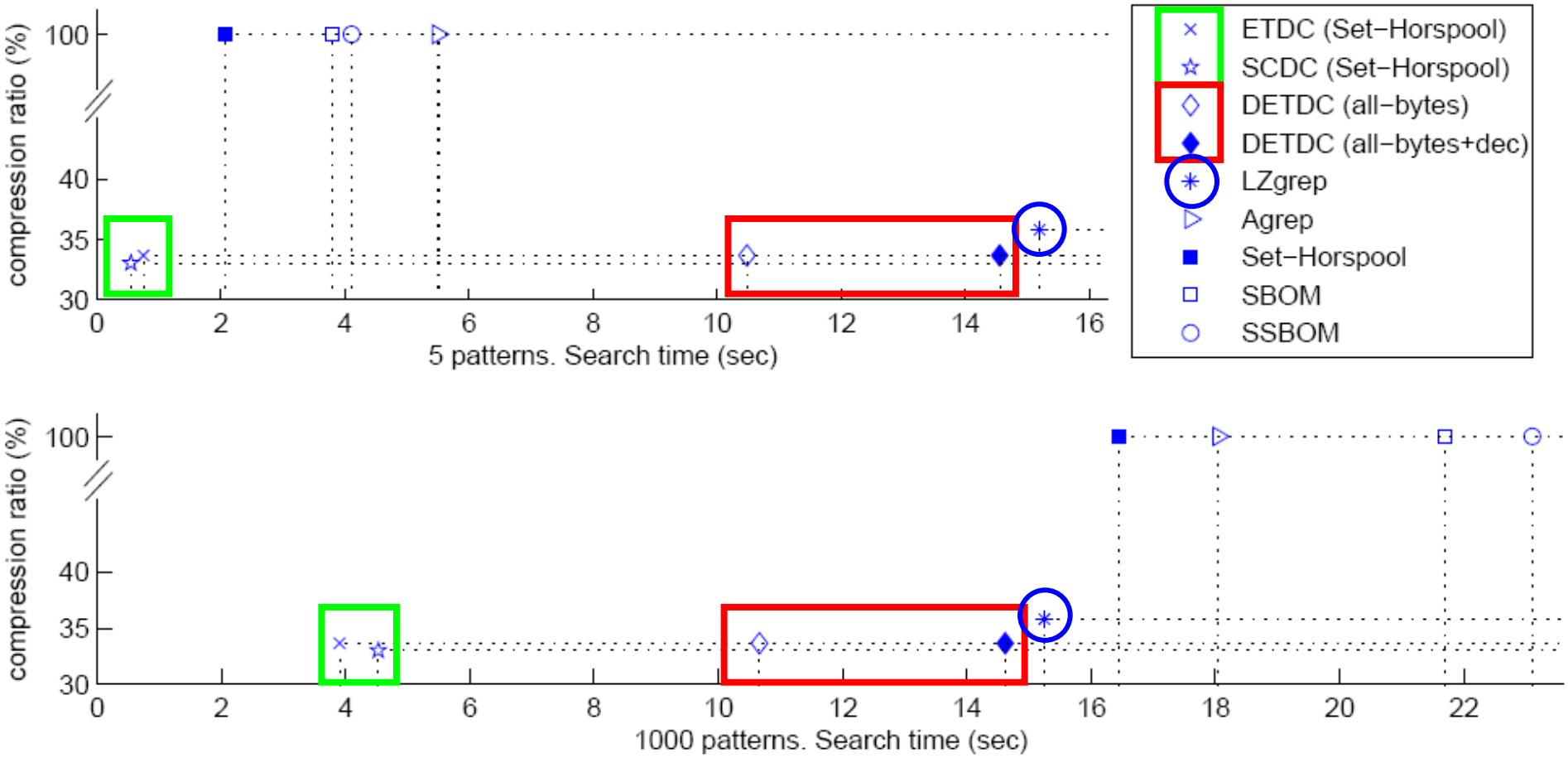
(patrones con 5 letras)

- Técnicas semistáticas son prácticamente “imbatibles”

Compresores dinámicos densos

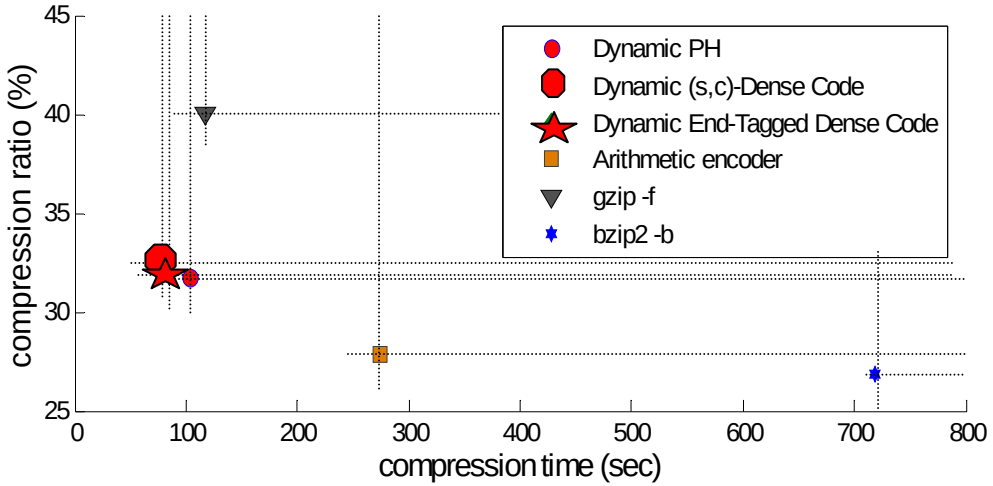
Resultados Empíricos: compresión Vs tiempo

- Ratio compresión Vs tiempo de búsqueda multipatrón.

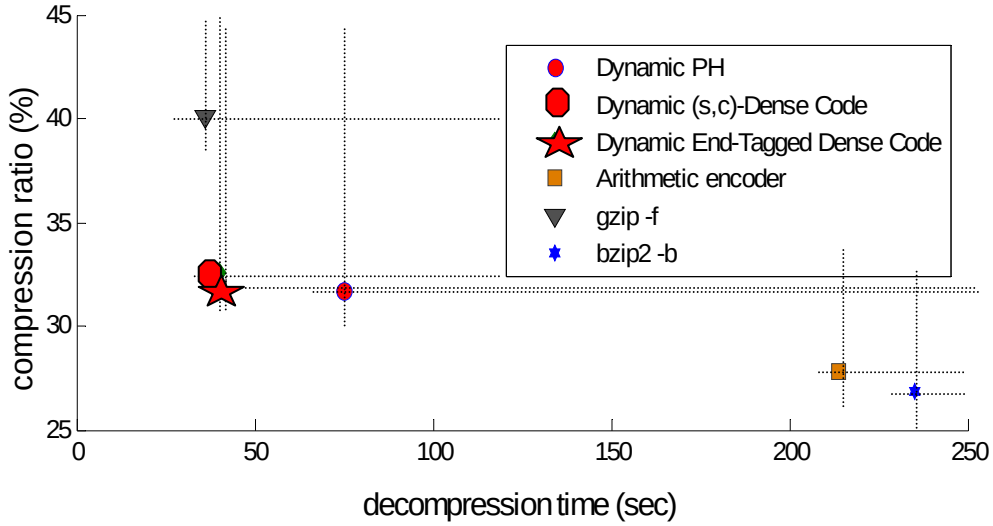


Compresores dinámicos densos

Resultados Empíricos: Resumen



Buen equilibrio entre Ratio Compresión y Tiempo Compresión



Buen equilibrio entre Ratio Compresión y Tiempo Descompresión

Compresores dinámicos densos

Sumario dinámicos

- 3 técnicas dinámicas estadísticas orientadas a palabras
  - Dyn-Huffman comprime más que los otros
  - DETDC es el más rápido (DSCDC está cercano a DETDC)
- Dinamismo:
  - Poco impacto en el ratio de compresión
  - Semiestáticos son más lentos que dinámicos al comprimir
  - Semiestáticos mucho más rápidos al descomprimir

+ Su interés

  - Menos compresión que bzip2 y aritm. pero mucho más rápido
  - Mejor ratio de compresión y velocidad que gzip. Más lento en descompresión
  - Búsquedas más rápidas que lzgrep (uncompress+search).
    - Simular descompresión: asociación “palabra” ←→ “código” varía

Buen ratio de compresión,

Muy **rápido** en **compresión** y

Buena **velocidad** en **descompresión**

→ **Búsquedas** simulando descompresión

Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos
- Compresores dinámicos Ligeros
  - Motivación
  - DLETDC (Dynamic Lightweight ETDC)
  - Búsquedas en texto comprimido con DLETDC
  - DLSCDC (Dynamic Lightweight (s,c)-DC)
  - Resultados Empíricos
- Compresión variable-to-variable

Compresores dinámicos ***ligeros***
Motivación

- El objetivo: Nuevas técnicas dinámicas, pero…
  - ***Descompresión*** más rápida ?
  - ***Permitir búsquedas*** eficientes Texto comprimido
  - Mantener ratio de compresión
  - Mantener velocidad de compresión
  - Permitiendo transmisión *en tiempo real*

Compresores dinámicos *ligeros*

Motivación

- Especialmente si …

  Tenemos problemas que impiden el uso de cualquier compresor dinámico:

  - Una PDA tiene que poca potencia de cálculo y memoria.
  - Necesitamos filtrar la información que llega a través de un canal (p.ej noticias)
    - Clasificación documentos según *keywords*

Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos
- Compresores dinámicos Ligeros
  - Motivación
  - DLETDC (Dynamic Lightweight ETDC)
  - Búsquedas en texto comprimido con DLETDC
  - DLSCDC (Dynamic Lightweight (s,c)-DC)
  - Resultados Empíricos
- Compresión variable-to-variable

- Brisaboa, N. R., Fariña, A., Navarro, G., Paramá, J. R. Dynamic Lightweight Text Compression. ACM Transactions on Information Systems (ACM-TOIS). Por aparecer, 3(1). New York, NY, USA, 2010

- Brisaboa, N. R., Fariña, A., Navarro, G., Paramá, J. R. Efficiently decodable and searchable natural language adaptive compression. Proc. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05), pp. 234-241. Bahia (Brazil), 2005

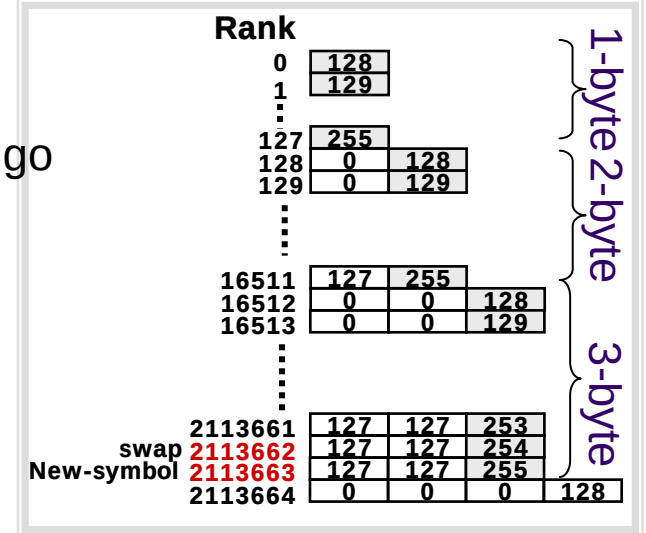
Compresores dinámicos ***ligeros***
DLETDC – Ideas básicas

No mantienen correspondencia entre posición y código

- Códigos mantenidos explícitamente por el emisor
- El receptor ignora rangos y frecuencias

- **EMISOR:** guarda el vocabulario ordenado por frecuencia
  - Intercambia palabras $\mathbf{s_i \leftrightarrow top(f_i)}$ para mantener el vocabulario ordenado
  - Mantiene los códigos originales a no ser que sus longitudes varíen
  - De lo contrario: **intercambio** y notificación al receptor

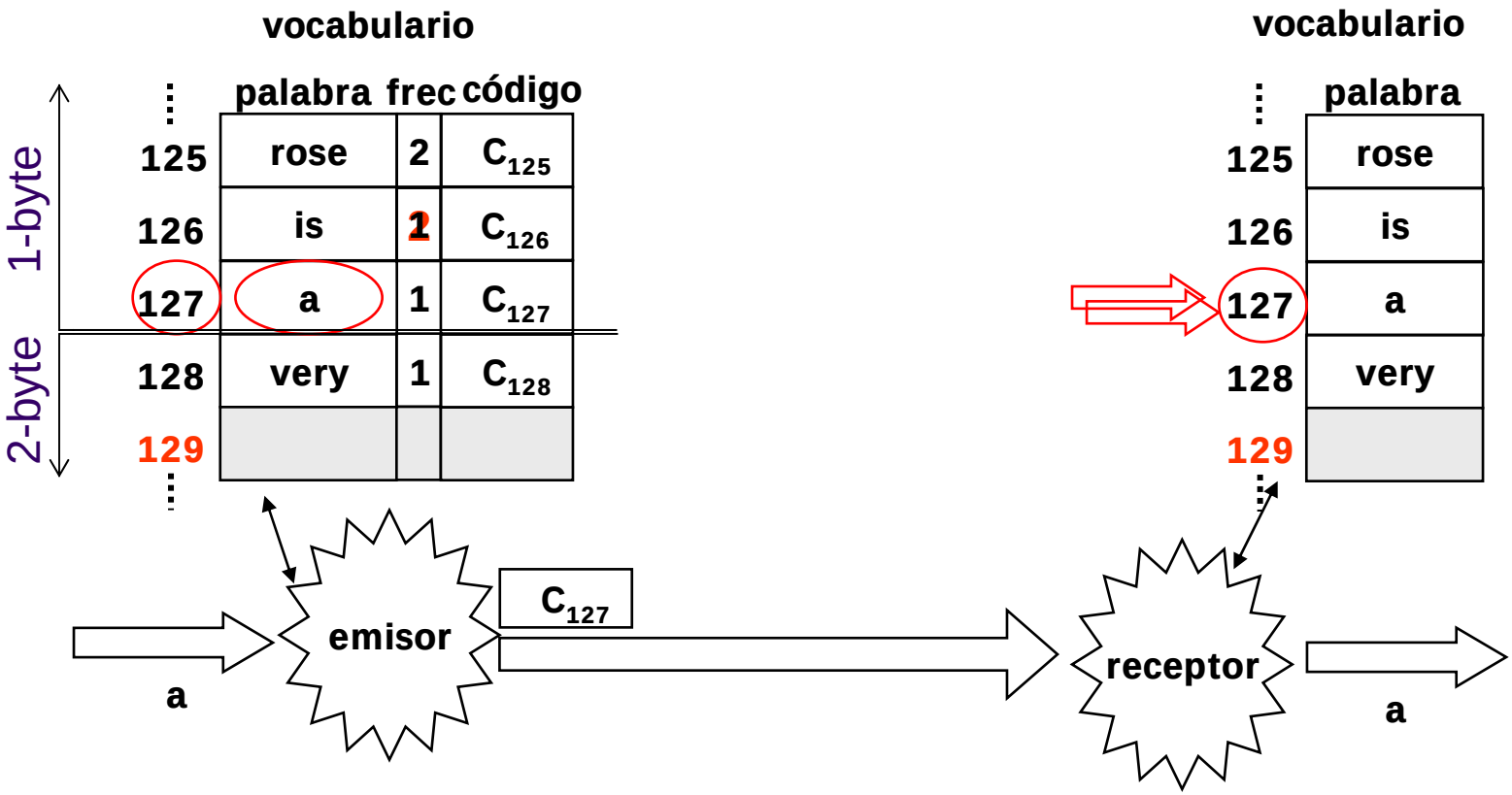
- **RECEPTOR:** mantiene palabras ordenadas por código
  - Decodifica los códigos
  - Añade nuevos símbolos sobre C_{new}
  - Sólo intercambia cuando decodifica a C_{swap}



Compresores dinámicos *ligeros*

DLETDC – Transmisión

Ejemplo: … a rose rose is a very nice one

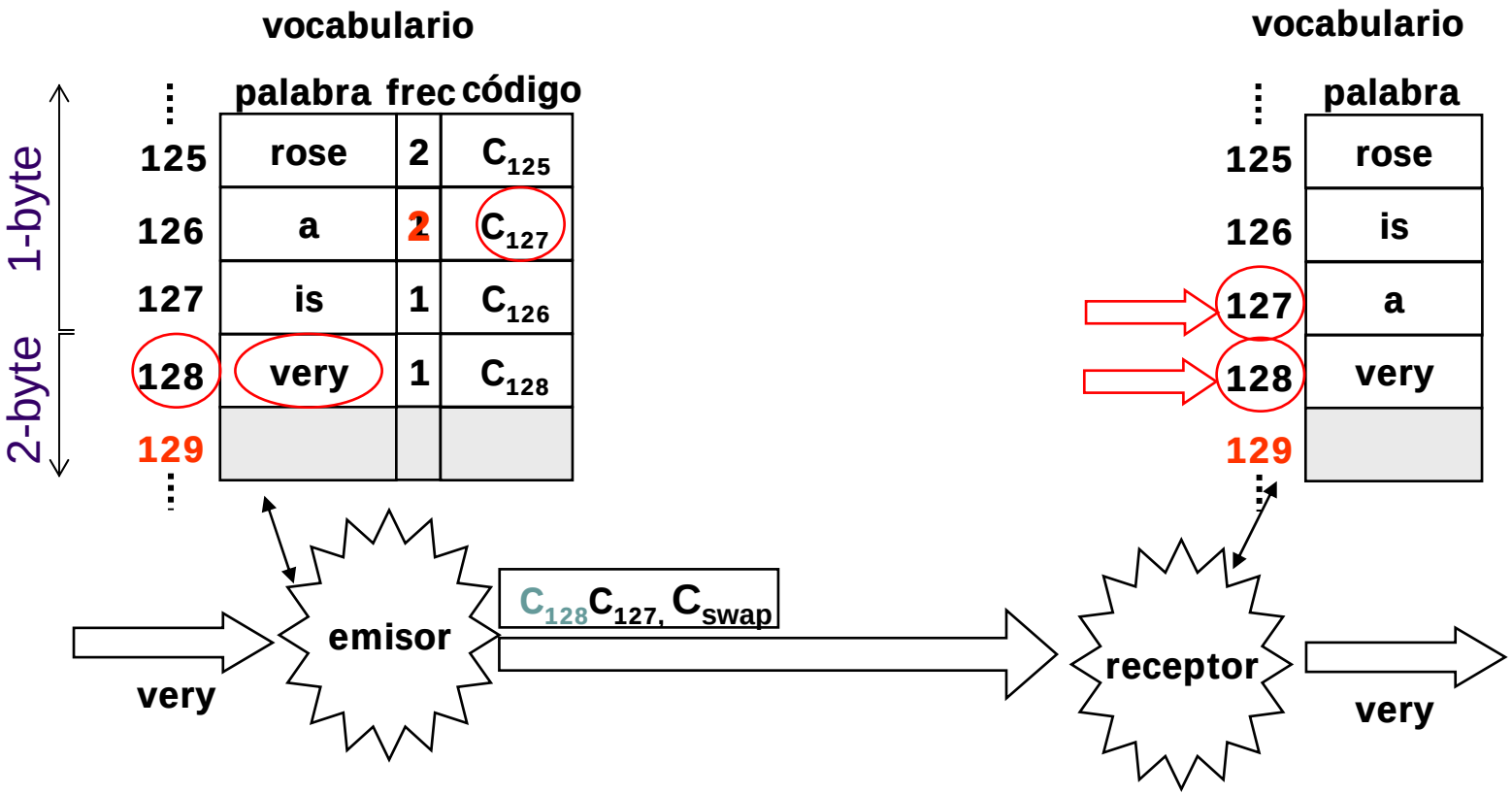


No se necesita intercambio de código puesto que: "a" ↔ C_{127} y "is" ↔ C_{126}

Compresores dinámicos *ligeros*

DLETDC – Transmisión

Ejemplo: … a rose rose is a very nice one

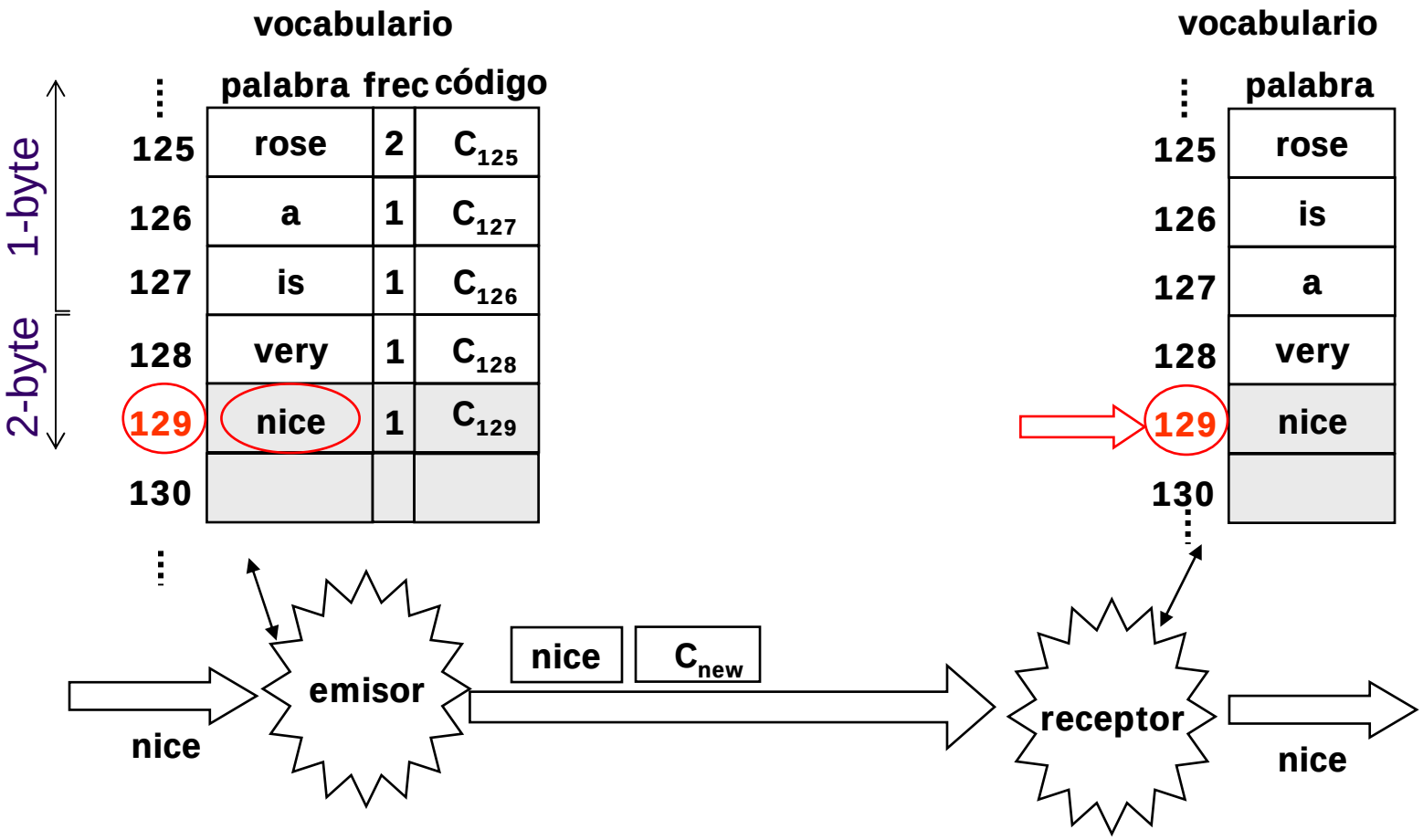


Se necesitar realiza un intercambio de códigos (1 y 2 bytes)

Compresores dinámicos *ligeros*

DLETDC – Transmisión

Ejemplo: … a rose rose is a very nice one

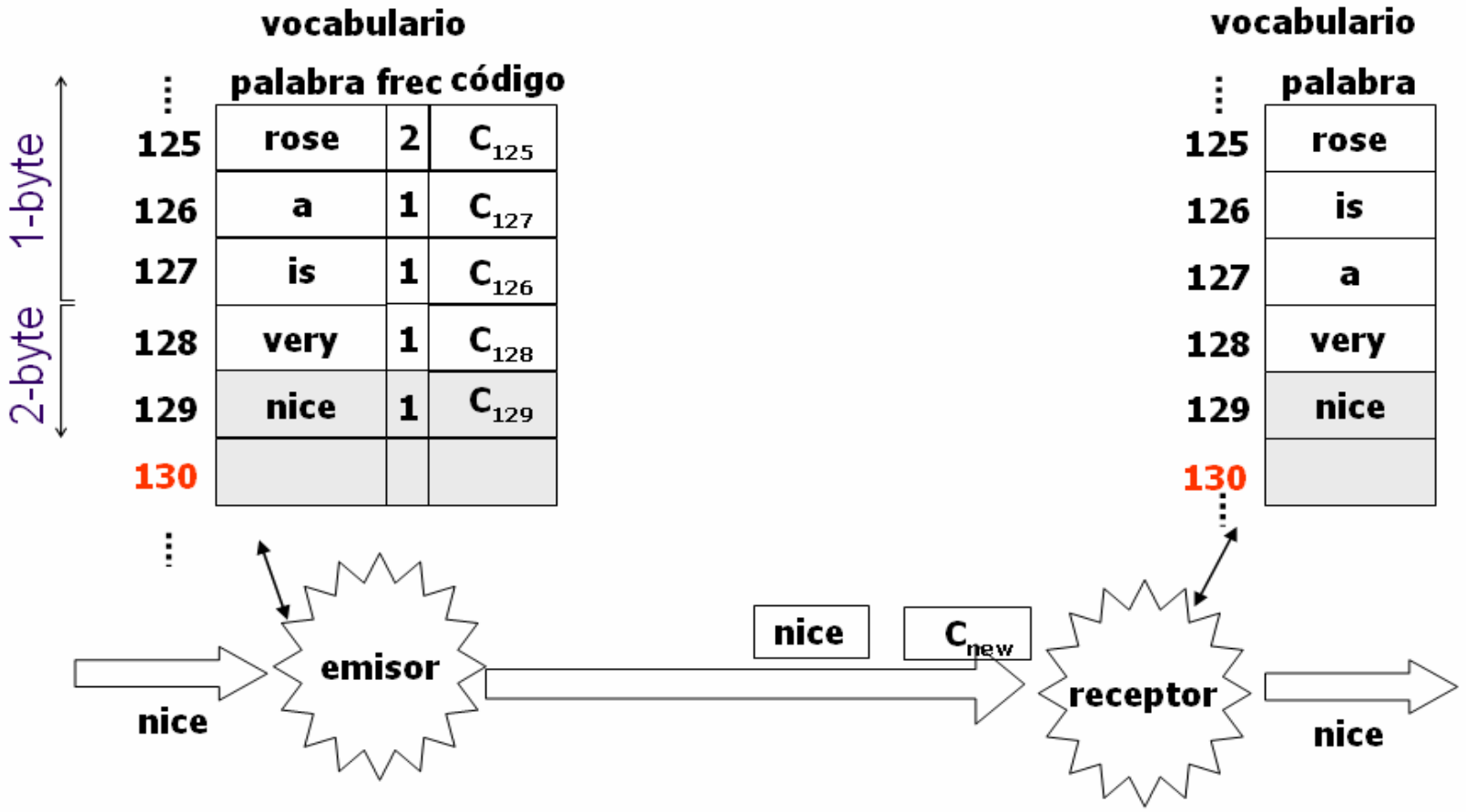


Caso en que se procesa una palabra nueva

Compresores dinámicos *ligeros*

DLETDC – Transmisión

Ejemplo: … a rose rose is a very nice one

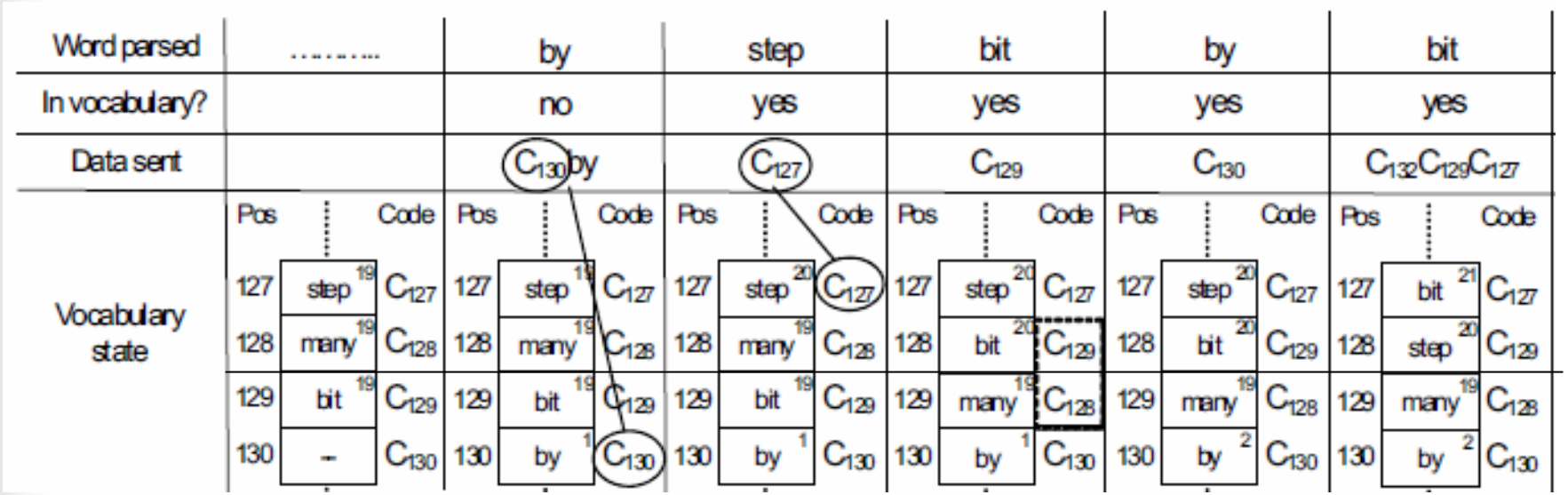


Caso en que se procesa una palabra nueva

Compresores dinámicos ***ligeros***

DLETDC – Ejemplo 2

- Ejemplo: Transmisión de "… by step bit by bit".



Compresores dinámicos ***ligeros***

DLETDC – pseudocódigo

Sender algorithm ( )

(1) Initialize *vocabulary structures*, $last \leftarrow 0$;
(2) **for** $i \leftarrow 1$ **to** $n-1$ **do** $top[i] \leftarrow 0$;
(3) **for** *each new symbol* i **do**
(4) **read** s_i *from the text*;
(5) $p \leftarrow hash(s_i)$;
(6) **if** $word(p) = Null$ (word $\notin$ vocabulary) **then**
(7) **send** $\{C_{zeroNode}$, s_i in plain form$\}$;
(8) $word[p] \leftarrow s_i$; $posInVoc[p] \leftarrow last$;
(9) $freq[p] \leftarrow 1$; $posInHT[last] \leftarrow p$;
(10) $codeword[p] \leftarrow encode(last)$;
(11) $last \leftarrow last + 1$;
(12) **else**
(13) $i \leftarrow posInVoc[p]$; $C_i \leftarrow codeword[p]$;
(14) $f \leftarrow freq[p]$; $j \leftarrow top[f]$;
(15) $h \leftarrow posInHT[j]$; $C_j \leftarrow codeword[h]$;
(16) **if** $size(C_j) = size(C_i)$ **then send** C_i;
(17) **else send** $\{C_{swap}, C_i, C_j\}$;
(18) **swap** $(codeword[p], codeword[h])$;
(19) **swap** $(posInHT[i], posInHT[j])$;
(20) **swap** $(posInVoc[p], posInVoc[h])$;
(21) $top[f] \leftarrow top[f] + 1$;
(22) $freq[p] \leftarrow f + 1$;

Receiver algorithm ( )

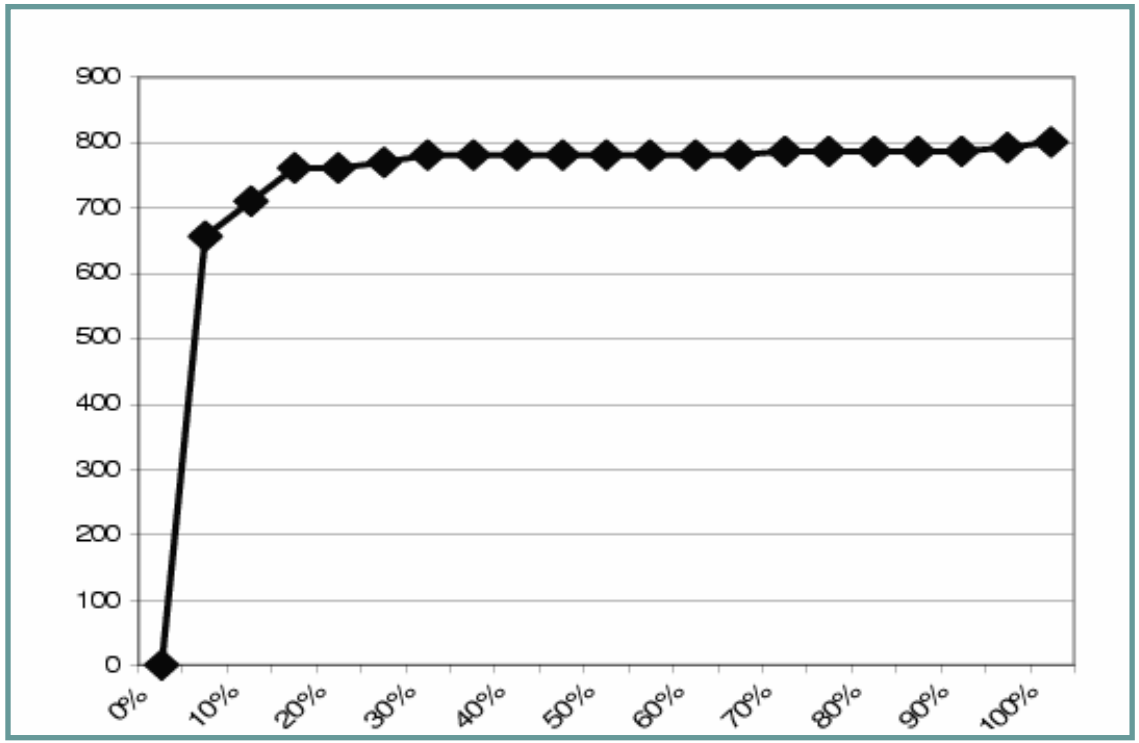
(1) Initialize *word*; $last \leftarrow 0$;
(2) **for** *each new codeword* p **do**
(3) **receive** C_p;
(4) **if** $C_p = C_{zeroNode}$ **then**
(5) **receive** s_p in plain form;
(6) $word[last] \leftarrow s_p$;
(7) $last \leftarrow last + 1$;
(8) **output** s_p;
(9) **else if** $C_p = C_{swap}$ **then**
(10) **receive** C_i,C_j;
(11) **swap** $(word[decode(C_i)], word[decode(C_j)])$;
(12) **output** $word[decode(C_j)]$;
(13) **else output** $word[decode(C_p)]$;

Compresores dinámicos *ligeros*

DLETDC – Evolución de intercambios

- Los intercambios empeoran el ratio de compresión y el tiempo de procesado.
- ¿Cuántos intercambios ocurren en la práctica?

Longitudes entre 1 y 2



ZIFF corpus

$4{,}6 \times 10^7$ words

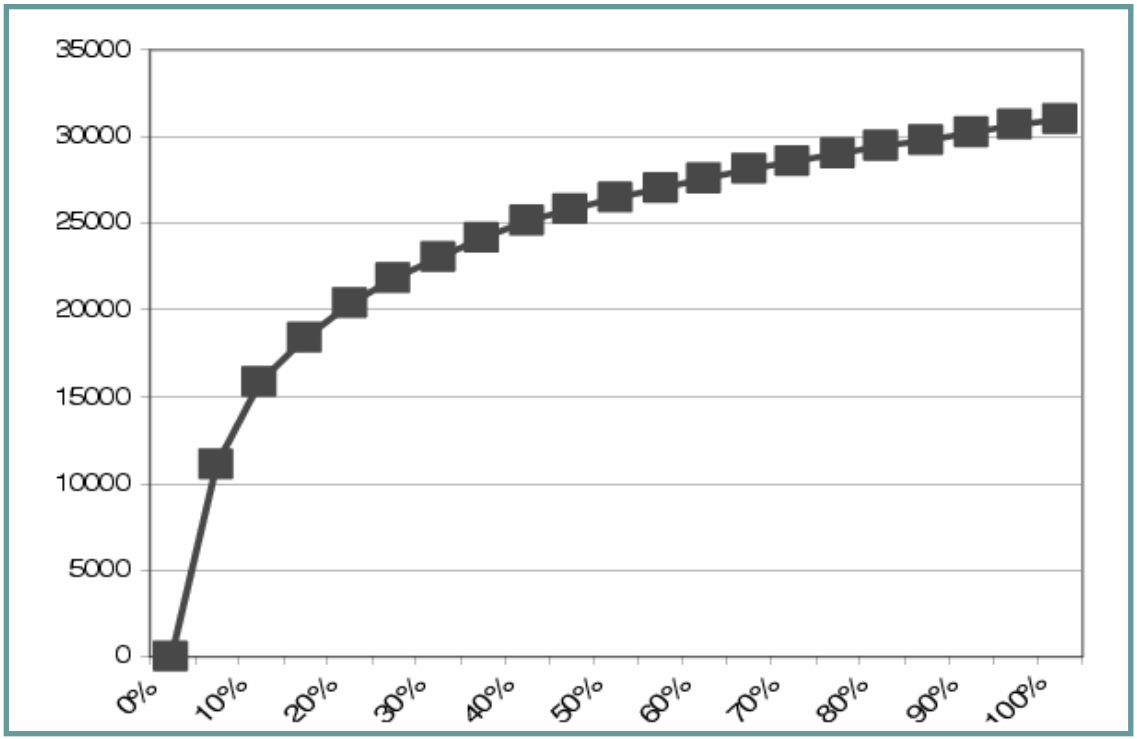
237,622 diff words

Compresores dinámicos *ligeros*

DLETDC – Evolución de intercambios

- Los intercambios empeoran el ratio de compresión y el tiempo de procesado.
- ¿Cuántos intercambios ocurren en la práctica?

Longitudes entre 2 y 3



ZIFF corpus

$4{,}6 \times 10^7$ words

237,622 diff words

Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos
- Compresores dinámicos Ligeros

  • Motivación
  • DLETDC (Dynamic Lightweight ETDC)
  • Búsquedas en texto comprimido con DLETDC
  • DLSCDC (Dynamic Lightweight (s,c)-DC)
  • Resultados Empíricos

- Compresión variable-to-variable

Compresores dinámicos ***ligeros***

DLETDC – Búsquedas – Filtrado de términos

- Horspool multipatrón
  - Patrones de búsqueda se colocan en un "*trie*"
- Proceso de búsqueda:
  - **Fase inicial**
    - Búsqueda para el patrón ASCII: C_new | (patrón)
  - **Cuando encuentra**
    - Lo reemplaza en el *trie* por su código: C_pat
  - **Observar los intercambios**
    - Buscar C_swap C_pat * y C_swap * C_pat

Guión de la exposición

- Compresores semiestáticos
- Compresores dinámicos
- Compresores dinámicos Ligeros
  - Motivación
  - DLETDC (Dynamic Lightweight ETDC)
  - Búsquedas en texto comprimido con DLETDC
  - DLSCDC (Dynamic Lightweight (s,c)-DC)
  - Resultados Empíricos

- Compresión variable-to-variable

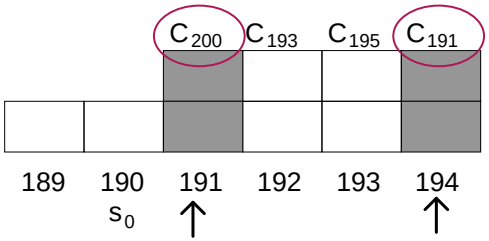
Compresores dinámicos *ligeros*

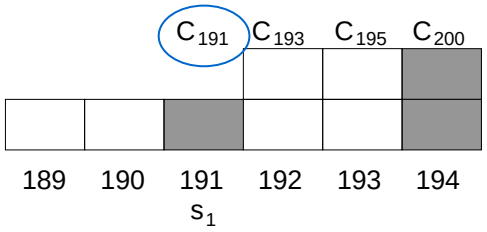
Dynamic Lightweight (s,c)-Dense Code

- Obtención de un DSCDC ligero.
  - Parámetros (s,c) variables → rangos variables
  - Alternativas:
    - Fijar (s,c) y aplicar algoritmo de DLETDC
    - Cada cambio de (s,c) → recolocación y notificación al receptor

- Solución adoptada:
  - Almacenar posición **x** en lugar de código $\mathbf{C_x}$| $\mathbf{C_x}$← *encode*(x,s)
  - Al cambiar s:
    - Recolocar y notificar sólo cambios en rango de 1 y 2 bytes

- Ejemplo: s cambia de **190** a **191**

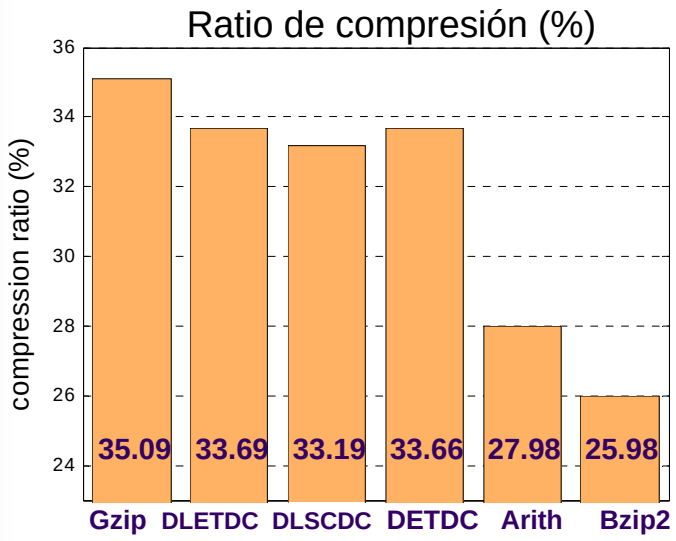


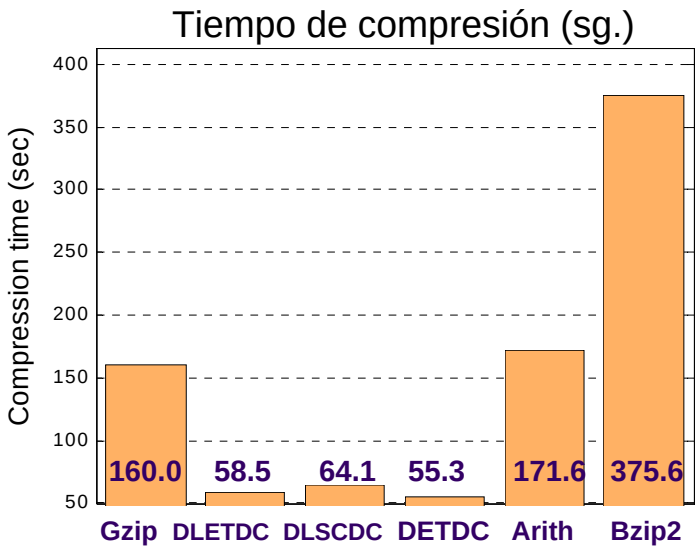


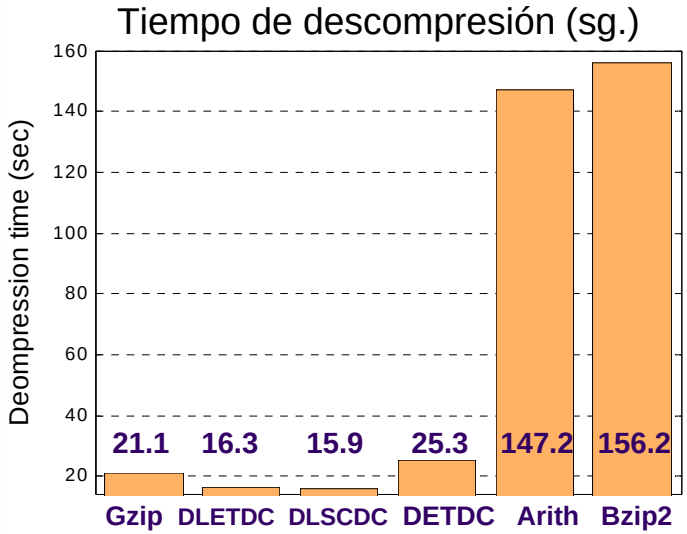
C_{191} código de 1 byte

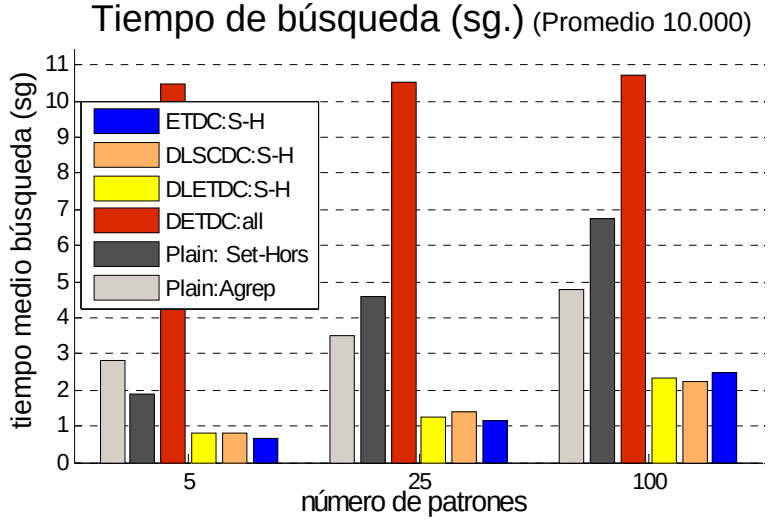
Compresores dinámicos *ligeros*

Resultados Empíricos



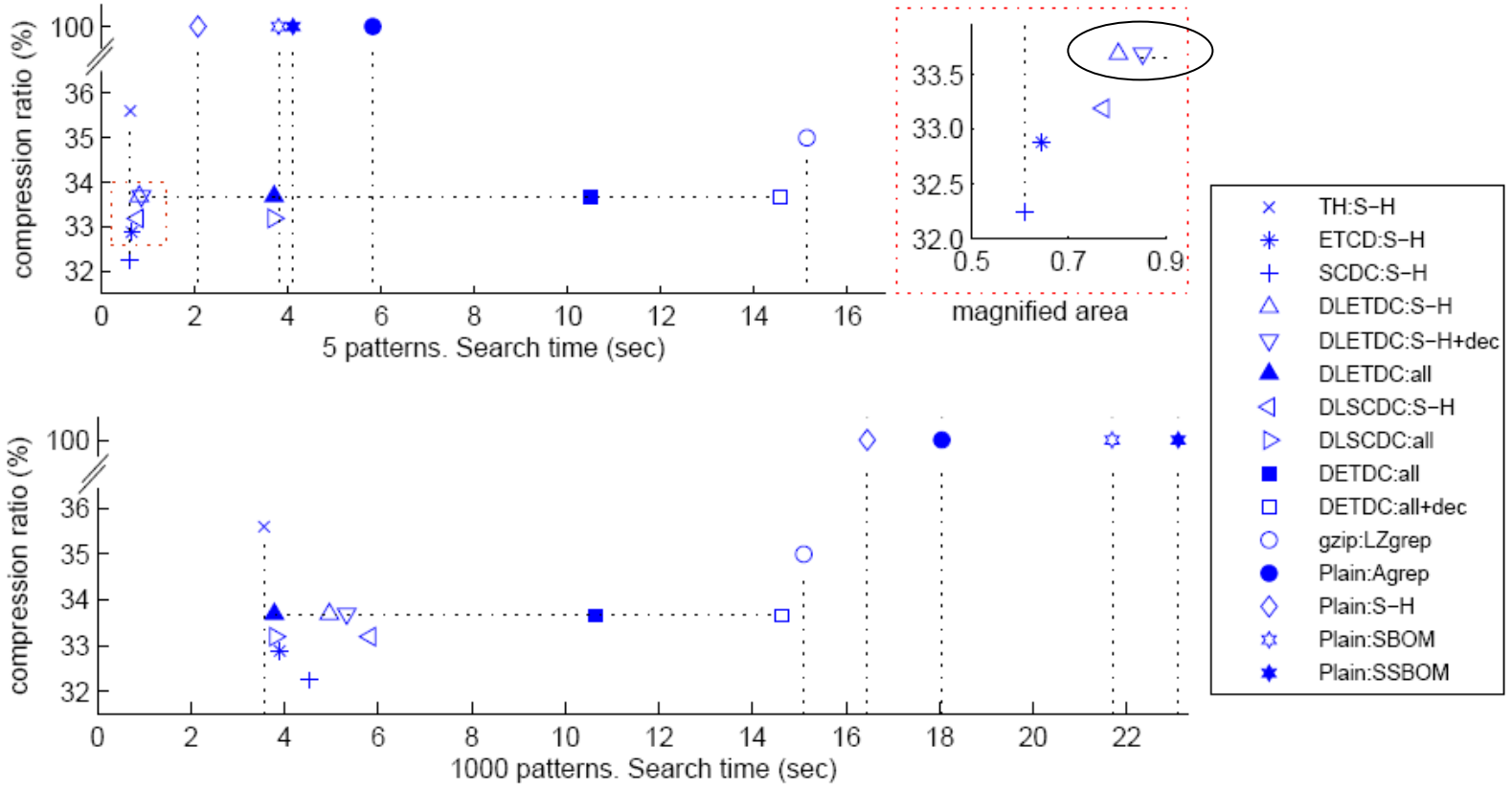






Compresores dinámicos *ligeros*

Resultados Empíricos



(a) Pattern length = 5.

Compresores dinámicos ***ligeros***

Sumario

- DLETDC y DLSCDC rompen la simetría común entre **emisor** y **receptor** en técnicas adaptativas
  - Diseñados para: bajo ancho de banda y receptores poco potentes
  - Búsquedas posibles sobre texto comprimido dinámicamente
- Compresores dinámicos ligeros:
  - Ratio de compresión competitivo (alrededor del 32-34% < gzip)
  - Rápido al comprimir (más rápido otros estadísticos dinámicos)
  - Muy rápido al descomprimir (incluso más rápido que gunzip!)
  - Muy rápido en las búsquedas (3-4 veces más rápido que búsqueda en texto plano)

Dense codes
sumario

- Familia "***Densa***" de compresores
  - Basados en palabras y orientada a bytes.
  - Ratio de compresión cercano al óptimo Plain Huffman
  - Muy fáciles de programar
- Compresores semi-estáticos ***ETDC y SCDC***
  - Mismas capacidades de búsqueda que Tagged Huffman.
  - Superan a Tagged Huffman en todos los aspectos.
- Compresores dinámicos ***DETDC y DSCDC***
  - Muy rápido al comprimir.
  - Buena velocidad de descompresión.
- Compresores dinámicos ligeros ***DLETDC y DLSCDC***
  - Permiten receptores con poca potencia PDA's
  - Rapidísima descompresión, superan a gzip en todos los aspectos.
  - Permiten búsquedas.