ΜΥΕ003: Ανάκτηση Πληροφορίας

Διδάσκουσα: Ευαγγελία Πιτουρά

Κεφάλαιο 5: Στατιστικά Συλλογής. Συμπίεση.

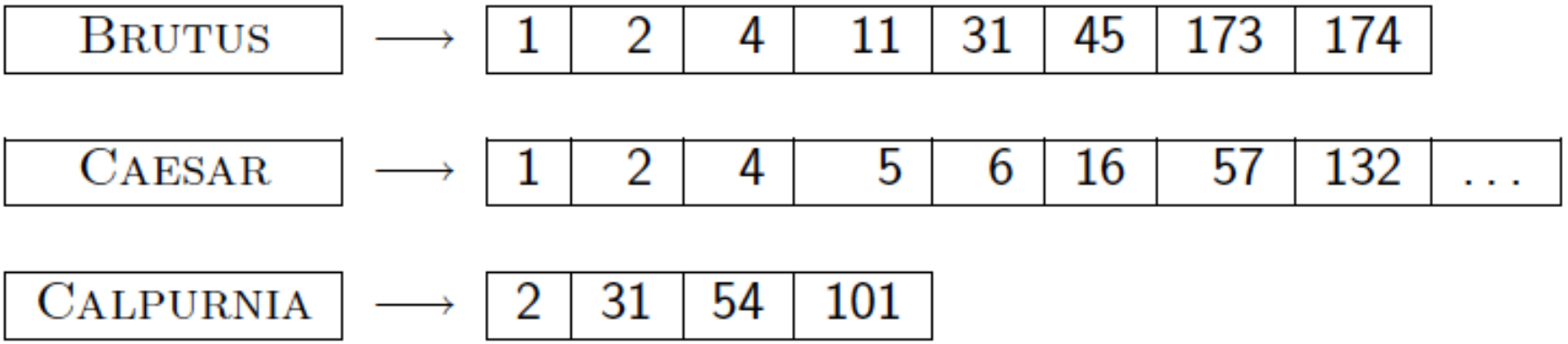
Τι θα δούμε σήμερα

- Κατασκευή ευρετηρίου
- Στατιστικά για τη συλλογή
- Συμπίεση

ΣΤΑΤΙΣΤΙΚΑ

Στατιστικά

- Πόσο μεγάλο είναι το λεξικό και οι καταχωρήσεις;



Η συλλογή RCV1

- Η συλλογή με τα άπαντα του Shakespeare δεν είναι αρκετά μεγάλη για το σκοπό της σημερινής διάλεξης.
- Η συλλογή που θα χρησιμοποιήσουμε δεν είναι στην πραγματικότητα πολύ μεγάλη, αλλά είναι διαθέσιμη στο κοινό.
- Θα χρησιμοποιήσουμε τη συλλογή RCV1.
  - Είναι ένας χρόνος του κυκλώματος ειδήσεων του Reuters (Reuters newswire) (μέρος του 1995 και 1996)
  - 1GB κειμένου

Ένα έγγραφο της συλλογής Reuters RCV1

REUTERS

You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enoug

Extreme conditions create rare Antarctic clouds

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprints

[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian meteorological base at Mawson Station on July 25.

Η συλλογή RCV1: στατιστικά

N	documents	800,000
L	tokens per document	200
M	terms (= word types)	400,000
	bytes per token (incl. spaces/punct.)	6
	bytes per token (without spaces/punct.)	4.5
	bytes per term (= word type)	7.5
T	non-positional postings	100,000,000

- *Γιατί κατά μέσο ένα term είναι μεγαλύτερο από ένα token;*

Μέγεθος ευρετηρίου

size of	word types (terms)			non-positional postings			positional postings		
	dictionary			non-positional index			positional index		
	Size (K)	Δ%	cumul %	Size (K)	Δ %	cumul %	Size (K)	Δ %	cumul %
Unfiltered	484			109,971			197,879		
No numbers	474	-2	-2	100,680	-8	-8	179,158	-9	-9
Case folding	392	-17	-19	96,969	-3	-12	179,158	0	-9
30 stopwords	391	-0	-19	83,390	-14	-24	121,858	-31	-38
150 stopwords	391	-0	-19	67,002	-30	-39	94,517	-47	-52
stemming	322	-17	-33	63,812	-4	-42	94,517	0	-52

Λεξιλόγιο και μέγεθος συλλογής

Ο **νόμος του Heaps**:

$$M = k\,T^b$$

M είναι το μέγεθος του λεξιλογίου (αριθμός όρων), T ο αριθμός των tokens στη συλλογή

περιγράφει πως μεγαλώνει το λεξιλόγιο όσο μεγαλώνει η συλλογή

- Συνήθης τιμές: $30 \leq k \leq 100$ (εξαρτάται από το είδος της συλλογής) και $b \approx 0.5$
- Σε log-log plot του μεγέθους M του λεξιλογίου με το T, ο νόμος προβλέπει γραμμή με κλίση περίπου ½

Λεξιλόγιο και μέγεθος συλλογής

Νόμος του Heaps:

$$M = k\,T^b$$

M είναι το μέγεθος του λεξιλογίου (αριθμός όρων), T ο αριθμός των tokens στη συλλογή

- περιγράφει πόσο μεγαλώνει το λεξιλόγιο όσο μεγαλώνει η συλλογή (το συνολικό μήκος των εγγράφων)

- Συνήθης τιμές: $30 \leq k \leq 100$ (εξαρτάται από το είδος της συλλογής) και $b \approx 0.5$

Λεξιλόγιο και μέγεθος συλλογής

- Diminishing returns: μπορούμε γρήγορα να καλύψουμε μέρος του λεξιλογίου, αλλά γίνεται όλο και πιο δύσκολο να το καλύψουμε όλο

- Σε log-log plot του μεγέθους M του λεξιλογίου με το T, ο νόμος προβλέπει γραμμή με κλίση περίπου ½

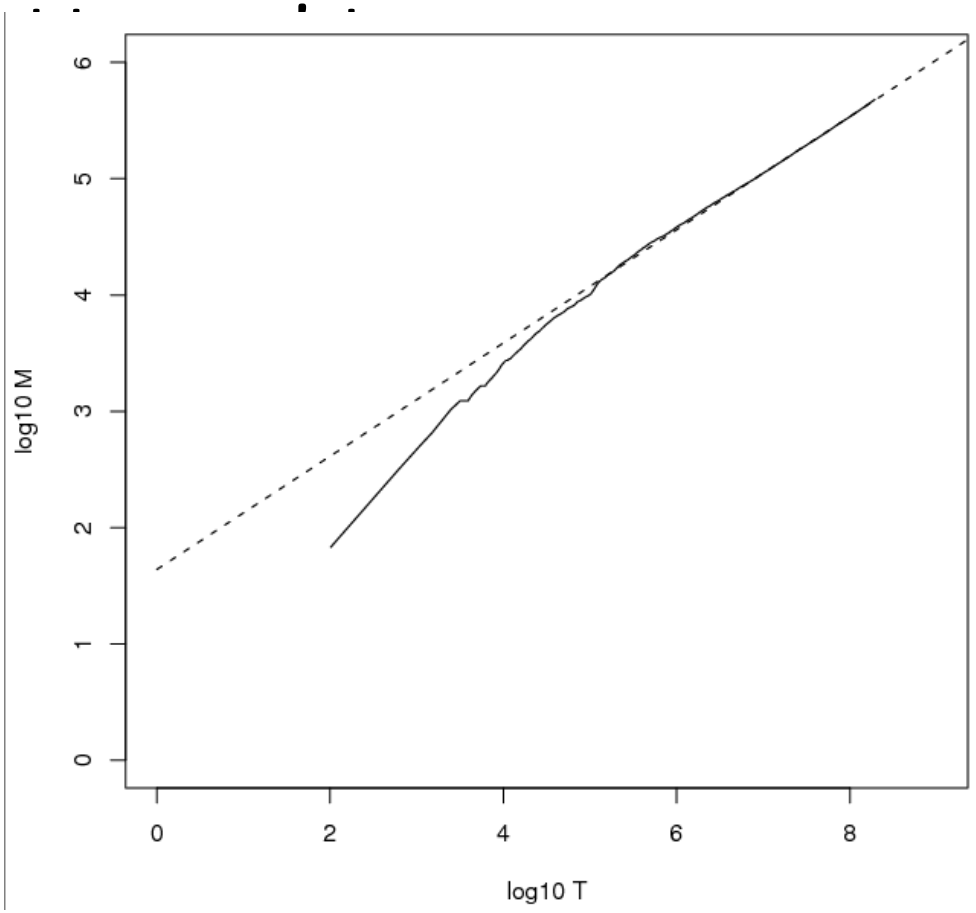
Για το RCV1, η διακεκομμένη γραμμή

$\log_{10} M = 0.49 \log_{10} T + 1.64$ (best least squares fit)

Οπότε, $M = 10^{1.64} T^{0.49}$, άρα $k = 10^{1.64} \approx 44$ and $b = 0.49$.

Καλή προσέγγιση για το Reuters RCV1!

Για το πρώτα 1,000,020 tokens, ο νόμος προβλέπει 38,323 όρους, στην πραγματικότητα 38,365



Ο νόμος του Heaps

Τα παρακάτω επηρεάζουν το μέγεθος του λεξικού (και την παράμετρο k):

- Stemming
- Including numbers
- Spelling errors
- Case folding

Ο νόμος του Zipf

✓ Ο νόμος του Heaps μας δίνει το μέγεθος του λεξιλογίου μιας συλλογής (σε συνάρτηση του μεγέθους της συλλογής)

Θα εξετάσουμε τη *σχετική συχνότητα* των όρων

- Στις φυσικές γλώσσες, υπάρχουν λίγοι πολύ συχνοί όροι και πάρα πολύ σπάνιοι

Ο νόμος του Zipf

Ο **νόμος του Zipf:** Ο i-οστός πιο συχνός όρος έχει συχνότητα ανάλογη του 1/*i* .

$$cf_i \propto 1/i = K/i$$

cf_i collection frequency: ο αριθμός εμφανίσεων του όρου t_i στη συλλογή

K μια normalizing constant

Η συχνότητα εμφάνισης ενός όρου είναι αντιστρόφως ανάλογη της θέσης του στη διάταξη με βάση τις συχνότητες

- Αν ο πιο συχνός όρος (ο όρος *the*) εμφανίζεται cf_1 φορές
- Τότε ο δεύτερος πιο συχνός (*of*) εμφανίζεται $cf_1/2$ φορές
- Ο τρίτος (*and*) $cf_1/3$ φορές
- …

Ο νόμος του Zipf

$cf_i = mi^{-k}$

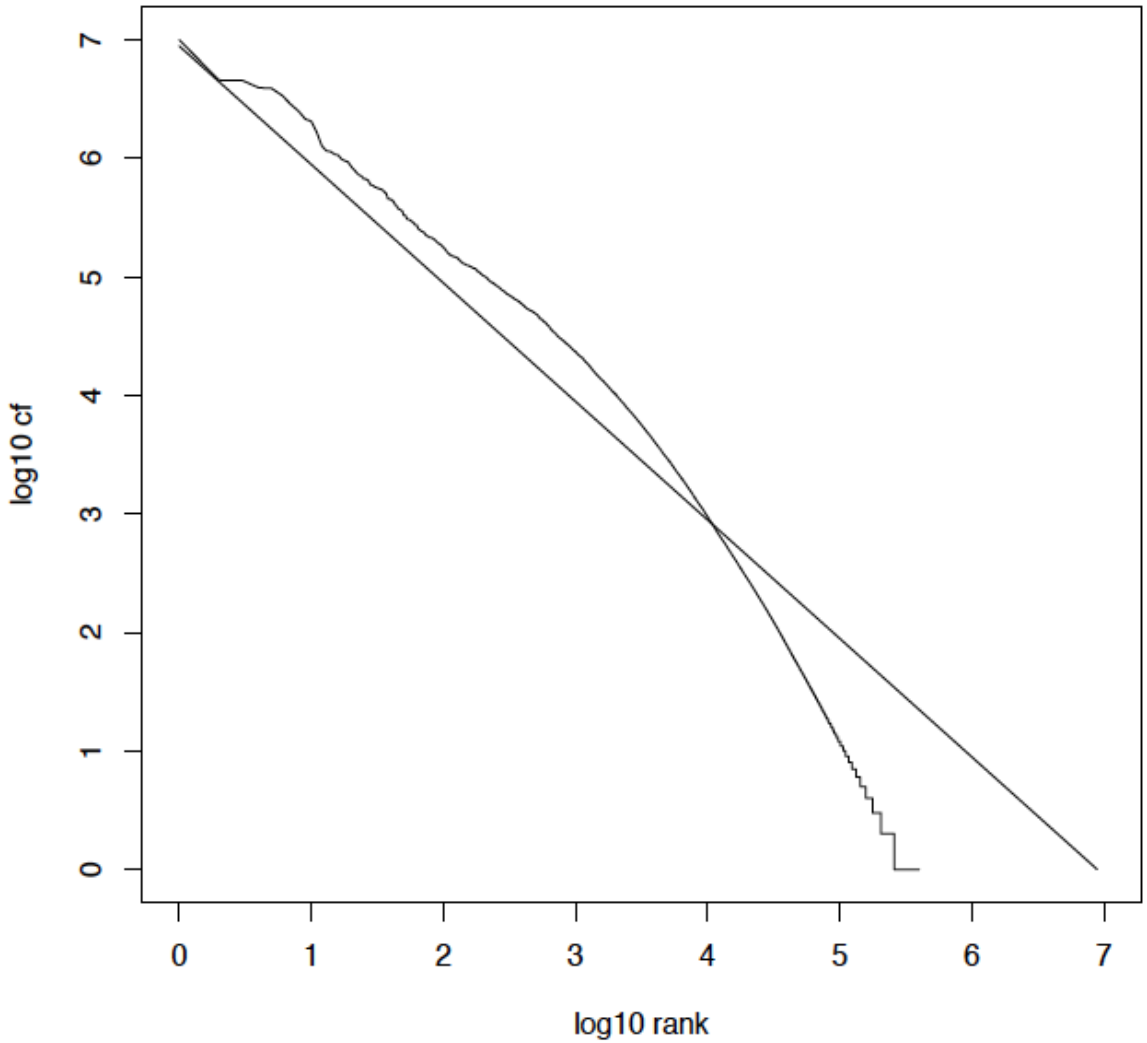
$\log cf_i = \log m - k \log i$

- Γραμμική σχέση μεταξύ $\log cf_i$ και $\log i$

$cf_i = m\, i^{-k}$, $k = 1$

power law σχέση (εκθετικός νόμος)

Zipf's law for Reuters RCV1



ΣΥΜΠΙΕΣΗ

Συμπίεση

- Θα δούμε μερικά θέματα για τη συμπίεση το *λεξικού* και των *λιστών καταχωρήσεων*
- Βασικό Boolean ανεστραμμένο ευρετήριο, χωρίς πληροφορία θέσης κλπ

Γιατί συμπίεση;

- Λιγότερος *χώρος στη μνήμη*
  - Λίγο πιο οικονομικό
- Κρατάμε περισσότερα πράγματα στη μνήμη
  - Αύξηση της ταχύτητας
- Αύξηση της ταχύτητας μεταφοράς δεδομένων από το δίσκο στη μνήμη
  - [διάβασε τα συμπιεσμένα δεδομένα| αποσυμπίεσε] γρηγορότερο από [διάβασε μη συμπιεσμένα δεδομένα]
  - Προϋπόθεση: Γρήγοροι αλγόριθμοι αποσυμπίεσης

Απωλεστική και μη συμπίεση

- Lossless compression: (μη απωλεστική συμπίεση) Διατηρείτε όλη η πληροφορία
  - Αυτή που κυρίως χρησιμοποιείται σε ΑΠ
- Lossy compression: (απωλεστική συμπίεση) Κάποια πληροφορία χάνεται
  - Πολλά από τα *βήματα προ-επεξεργασίας* (μετατροπή σε μικρά, stop words, stemming, number elimination) μπορεί να θεωρηθούν ως απωλεστική συμπίεση
  - Μπορεί να είναι αποδεκτή στην περίπτωση π.χ., που μας ενδιαφέρουν μόνο τα κορυφαία από τα σχετικά έγγραφα

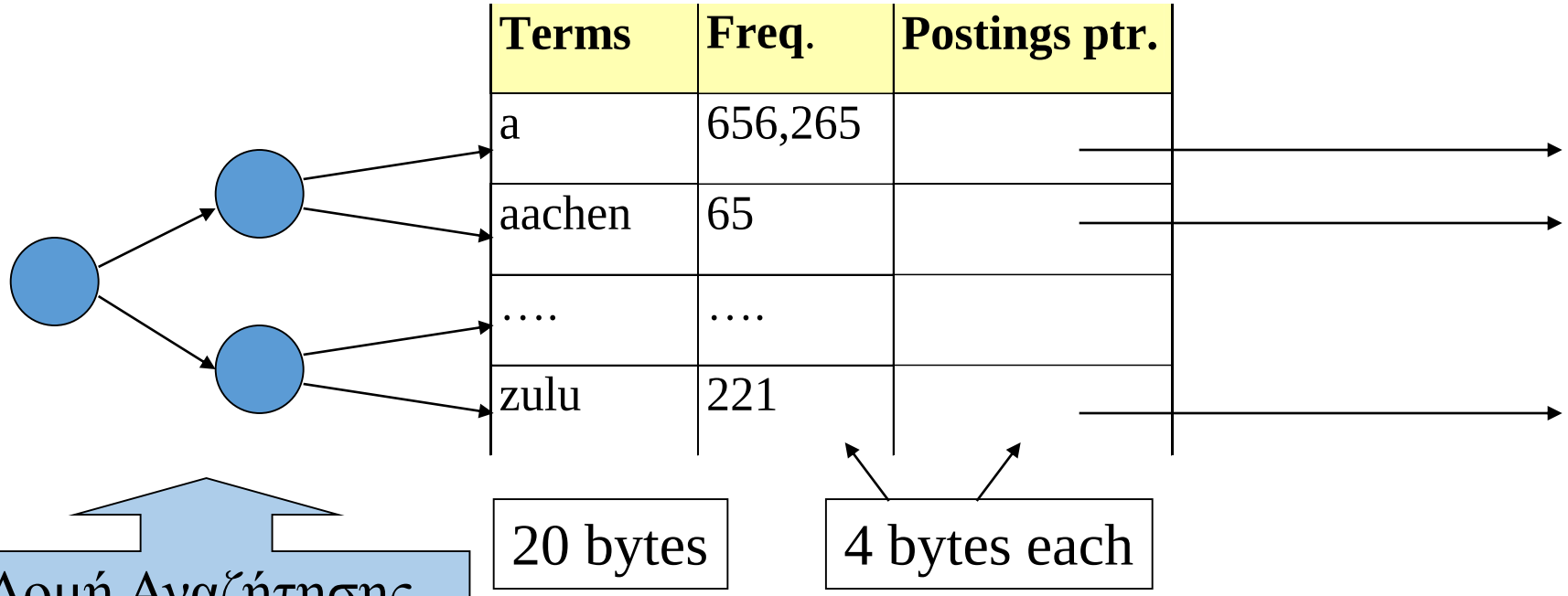
ΣΥΜΠΙΕΣΗ ΛΕΞΙΚΟΥ

Συμπίεση λεξικού

- Η αναζήτηση αρχίζει από το λεξικό -> Θα θέλαμε να το κρατάμε στη μνήμη
- Συνυπάρχει με άλλες εφαρμογές (memory footprint competition)
- Κινητές/ενσωματωμένες συσκευές μικρή μνήμη
- Ακόμα και αν όχι στη μνήμη, θα θέλαμε να είναι μικρό για γρήγορη αρχή της αναζήτησης

Αποθήκευση λεξικού

- Κάθε εγγραφή: τον όρο, συχνότητα εμφάνισης, δείκτη
- Θα θεωρήσουμε *την πιο απλή αποθήκευση*, ως ταξινομημένο πίνακα εγγραφών σταθερού μεγέθους (array of fixed-width entries)
  - ~400,000 όροι; 28 bytes/term = 11.2 MB.



(20+4+4) 400,000=11,2MB

Θα την αγνοήσουμε

4 bytes pointers -> 4GB address space (more bytes may be needed for larger collections)

Αποθήκευση λεξικού

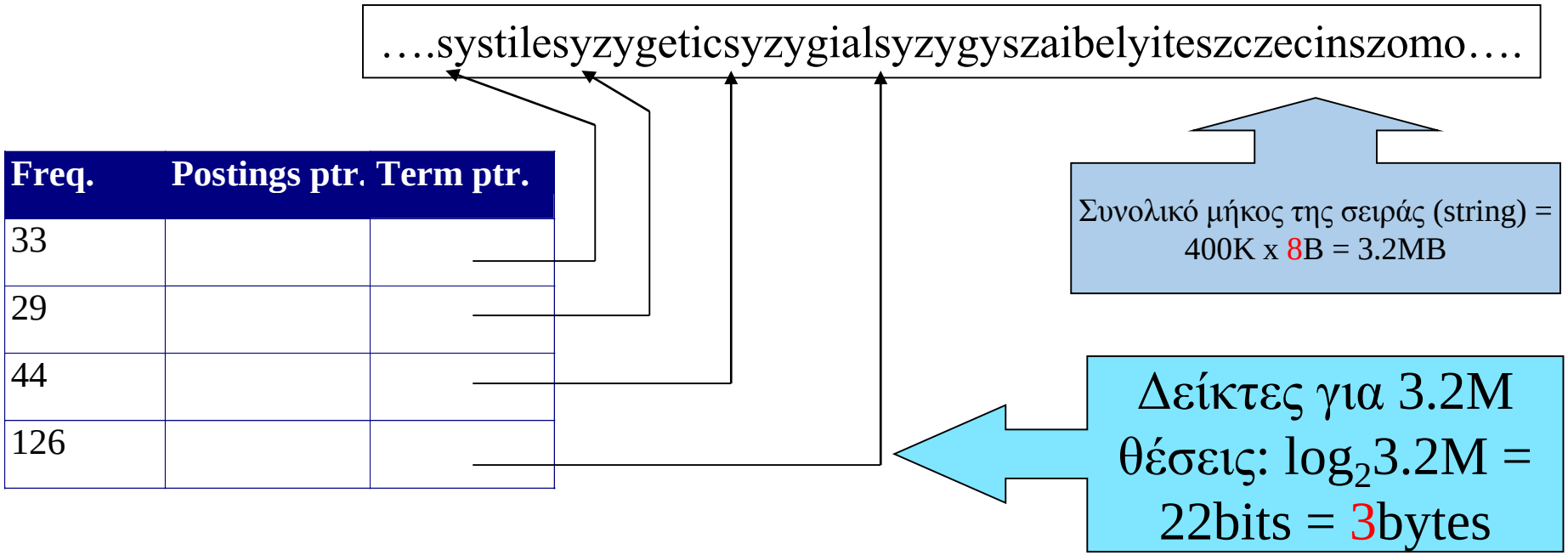
Σπατάλη χώρου

- Πολλά από τα bytes στη στήλη **Term** *δεν* χρησιμοποιούνται – δίνουμε 20 bytes για όρους με 1 χαρακτήρα
  - Και δε μπορούμε να χειριστούμε το *supercalifragilisticexpialidocious* ή *hydrochlorofluorocarbons (λέξεις με πάνω από 20 χαρακτήρες)*
- Μέσος όρος στο γραπτό λόγο για τα Αγγλικά είναι ~4.5 χαρακτήρες/λέξη.
- Μέσος όρος των λέξεων στο λεξικό για τα Αγγλικά: ~8 χαρακτήρες
- Οι μικρές λέξεις κυριαρχούν στα tokens αλλά όχι στους όρους.

Συμπίεση της λίστας όρων: Λεξικό-ως-Σειρά-Χαρακτήρων

Αποθήκευσε το λεξικό ως ένα (μεγάλο) string χαρακτήρων:

- Ένας δείκτης δείχνει στο τέλος της τρέχουσας λέξης (αρχή επόμενης)
- Εξοικονόμηση 60% του χώρου



δυαδική αναζήτηση όπως πριν, τώρα στο string

Χώρος για το λεξικό ως string

- 4 bytes ανά όρο για το Freq.
- 4 bytes ανά όρο για δείκτες σε Postings.
- 3 bytes ανά term pointer

} Κατά μέσο όρο:11bytes /term

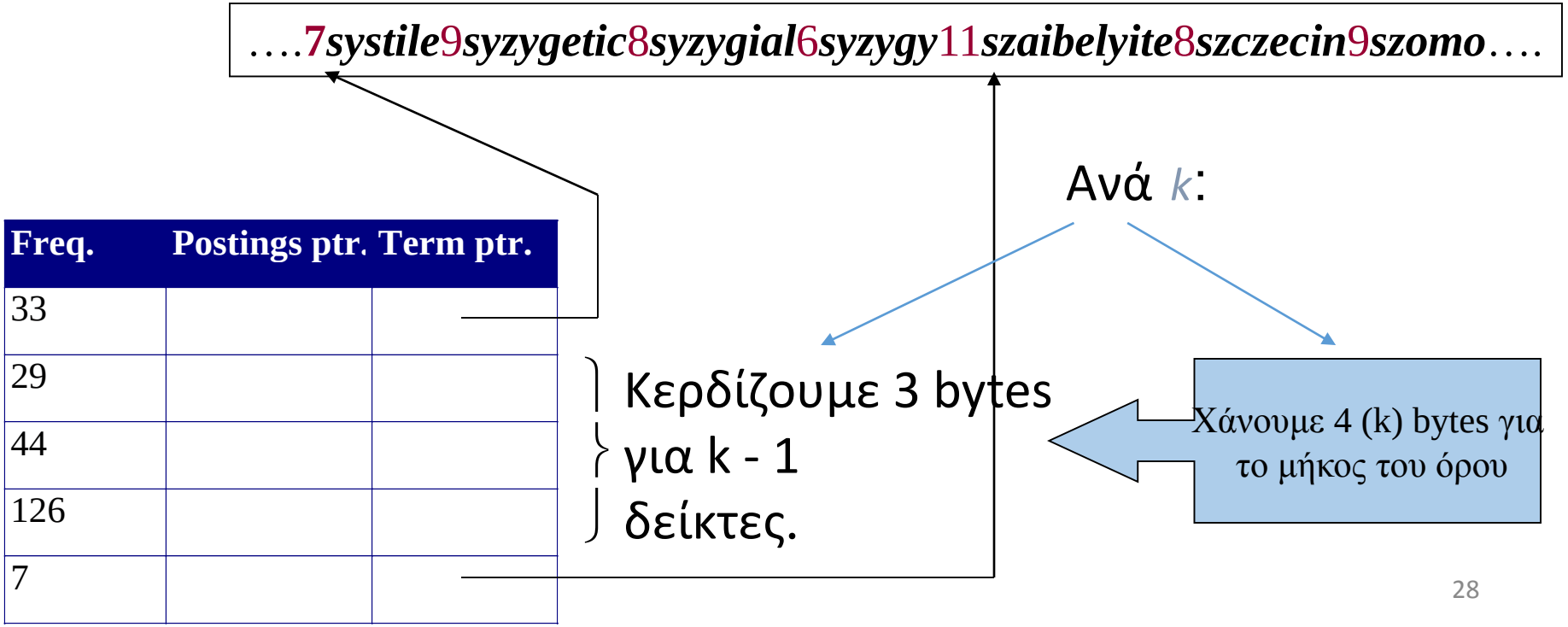
- Κατά μέσο όρο 8 bytes ανά όρο στο string (3.2MB)

- 400K όροι x 19 ⇒ 7.6 MB (έναντι 11.2MB για σταθερό μήκος λέξης)

11+8

Blocking (Δείκτες σε ομάδες)

- Διαίρεσε το string σε ομάδες (blocks) των *k* όρων
- Διατήρησε ένα δείκτη σε κάθε ομάδα
  - Παράδειγμα: *k* = 4.
- Χρειαζόμαστε και το μήκος του όρου (1 extra byte)



Blocking

Συνολικό όφελος για block size $k = 4$

- Χωρίς blocking 3 bytes/pointer
  - 3 x 4 = 12 bytes, (ανά block)

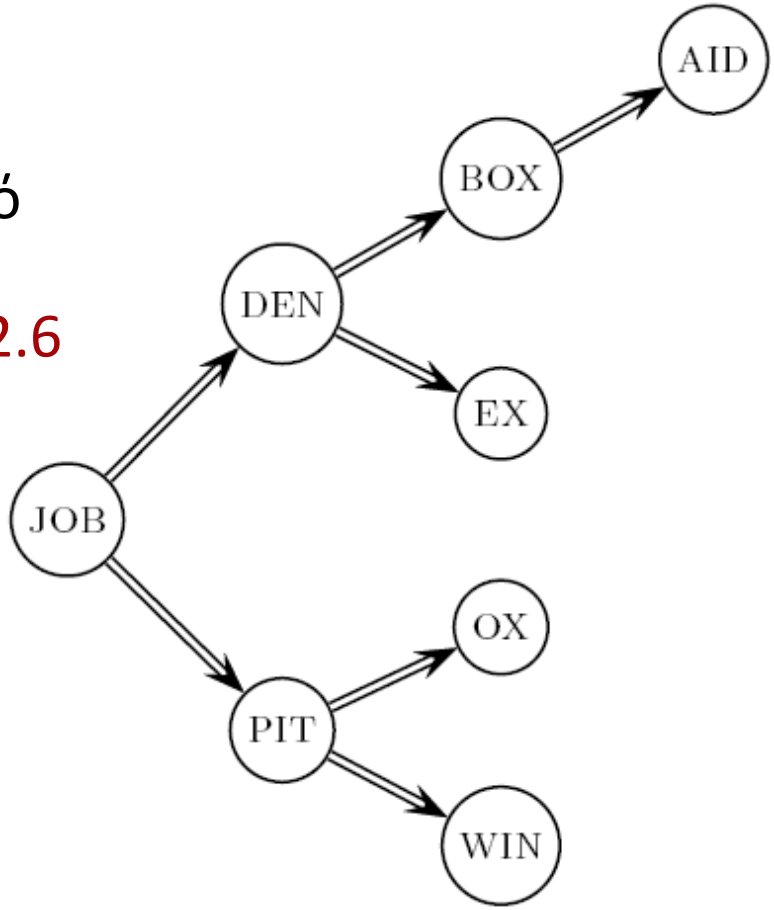
Τώρα 3 + 4 = 7 bytes.

Εξοικονόμηση ακόμα ~0.5MB. Ελάττωση του μεγέθους του ευρετηρίου από 7.6 MB σε 7.1 MB.

- Γιατί όχι ακόμα μεγαλύτερο *k*;
- Σε τι χάνουμε;

Αναζήτηση στο λεξικό χωρίς Blocking

- Ας υποθέσουμε *δυαδική αναζήτηση* και ότι κάθε όρος ισοπίθανο να εμφανιστεί στην ερώτηση (όχι και τόσο ρεαλιστικό στη πράξη) μέσος αριθμός συγκρίσεων = (1+2·2+4·3+4)/8 ~2.6



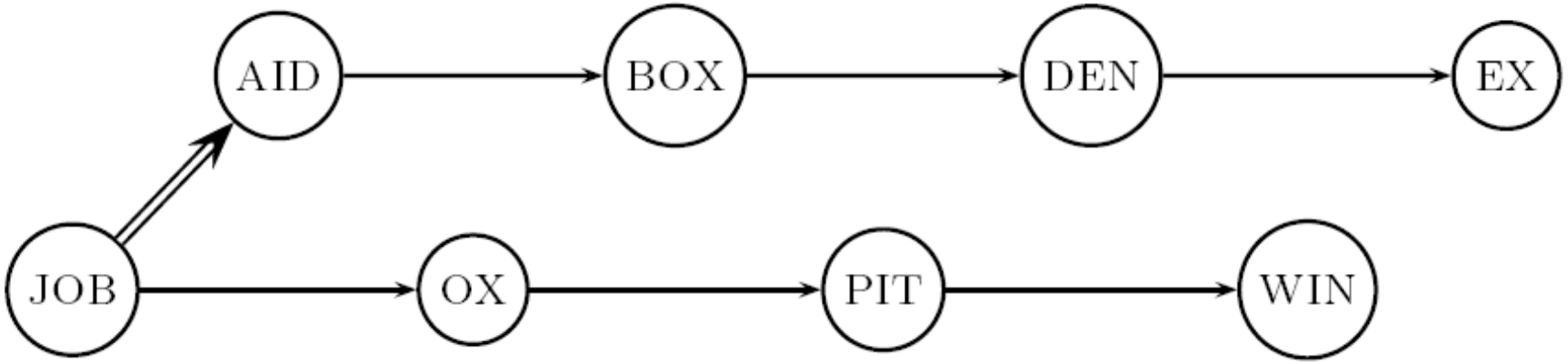
Άσκηση: σκεφτείτε ένα καλύτερο τρόπο αναζήτησης αν δεν έχουμε ομοιόμορφη κατανομή των όρων

Αναζήτηση στο λεξικό με Blocking

Δυαδική αναζήτηση μας οδηγεί σε ομάδες (block) από $k = 4$ όρους

Μετά *γραμμική αναζήτηση* στους $k = 4$ αυτούς όρους.

Μέσος όρος $(1+2\cdot2+2\cdot3+2\cdot4+5)/8 = 3$



Εμπρόσθια κωδικοποίηση (Front coding)

Οι λέξεις συχνά έχουν μεγάλα κοινά προθέματα – αποθήκευση μόνο των διαφορών

8*automata*8*automate*9*automatic*10*automation*

→8*automat***a*1◊*e*2◊*ic*3◊*ion*

Encodes ***automat***

Extra length beyond ***automat.***

Εμπρόσθια κωδικοποίηση (Front coding)

Αν στο δίσκο, μπορούμε να έχουμε ένα B-δέντρο με τον πρώτο όρο σε κάθε σελίδα

Κατακερματισμός ελαττώνει το μέγεθος αλλά πρόβλημα με ενημερώσεις

Περίληψη συμπίεσης για το λεξικό του RCV1

Τεχνική	Μέγεθος σε MB
Fixed width	11.2
Dictionary-as-String with pointers to every term	7.6
Also, blocking $k = 4$	7.1
Also, Blocking + front coding	5.9

ΣΥΜΠΙΕΣΗ ΤΩΝ ΚΑΤΑΧΩΡΗΣΕΩΝ

Συμπίεση των καταχωρήσεων

- Το αρχείο των καταχωρήσεων είναι *πολύ μεγαλύτερο* αυτού του λεξικού - τουλάχιστον 10 φορές.
- Βασική επιδίωξη: *αποθήκευση κάθε καταχώρησης συνοπτικά*
- Στην περίπτωση μας, μια καταχώρηση είναι το αναγνωριστικό ενός εγγράφου (docID).
  - Για τη συλλογή του Reuters (800,000 έγγραφα), μπορούμε να χρησιμοποιήσουμε 32 bits ανά docID αν έχουμε ακεραίους 4-bytes.
  - Εναλλακτικά, $\log_2$ 800,000 ≈ 20 bits ανά docID.
- Μπορούμε λιγότερο από 20 bits ανά docID;

Η συλλογή RCV1: στατιστικά

N	documents	800,000
L	tokens per document	200
M	terms (= word types)	400,000
	bytes per token (incl. spaces/punct.)	6
	bytes per token (without spaces/punct.)	4.5
	bytes per term (= word type)	7.5
T	non-positional postings	100,000,000

- *Γιατί κατά μέσο ένα term είναι μεγαλύτερο από ένα token;*

Συμπίεση των καταχωρήσεων

- Μέγεθος της συλλογής
  - 800,000 (έγγραφα)×200 (token)× 6 bytes = 960 MB
- Μέγεθος του αρχείου καταχωρήσεων
  - 100,000,000 (καταχωρήσεις)×20/8 bytes = 250MB

Συμπίεση των καταχωρήσεων

- Αποθηκεύουμε τη λίστα των εγγράφων σε αύξουσα διάταξη των docID.
  - ***computer***: 33, 47, 154, 159, 202 …
- Συνέπεια: αρκεί να αποθηκεύουμε τα διάκενα (*gaps*).
  - 33, 14, 107, 5, 43 …

- Γιατί; Τα περισσότερα διάκενα μπορεί να κωδικοποιηθούν/αποθηκευτούν με πολύ λιγότερα από 20 bits.

Παράδειγμα

	encoding	postings list									
THE	docIDs	...		283042		283043		283044		283045	...
	gaps				1		1		1		...
COMPUTER	docIDs	...		283047		283154		283159		283202	...
	gaps				107		5		43		...
ARACHNOCENTRIC	docIDs	252000		500100							
	gaps	252000	248100								

Παρόμοια ιδέα και για *positional indexes* (κωδικοποίηση των κενών ανάμεσα στις θέσεις)

Συμπίεση των καταχωρήσεων

- Ένας όρος όπως ***arachnocentric*** εμφανίζεται ίσως σε ένα έγγραφο στο εκατομμύριο.
- Ένας όρος όπως ***the*** εμφανίζεται σχεδόν σε κάθε έγγραφο, άρα 20 bits/εγγραφή πολύ ακριβό

Κωδικοποίηση μεταβλητού μεγέθους (Variable length encoding)

Στόχος:

- Για το ***arachnocentric***, θα χρησιμοποιήσουμε εγγραφές ~20 bits/gap.
- Για το ***the***, θα χρησιμοποιήσουμε εγγραφές ~1 bit/gap entry.

- Αν το μέσο κενό για έναν όρο είναι *G*, θέλουμε να χρησιμοποιήσουμε εγγραφές ~$\log_2 G$ bits/gap.
- Βασική πρόκληση: κωδικοποίηση κάθε ακεραίου (gap) *με όσα λιγότερα bits* είναι απαραίτητα για αυτόν τον ακέραιο.
- Αυτό απαιτεί κωδικοποίηση μεταβλητού μεγέθους -- *variable length encoding*
- Αυτό το πετυχαίνουμε χρησιμοποιώντας σύντομους κώδικες για μικρούς αριθμούς

Κωδικοί μεταβλητών Byte (Variable Byte (VB) codes)

- Κωδικοποιούμε κάθε διάκενο με ακέραιο αριθμό από bytes
- Το πρώτο bit κάθε byte χρησιμοποιείται ως *bit συνέχισης* (continuation bit)
  - 0, αν ακολουθεί και άλλο byte
  - 1, αλλιώς (αν το τελευταίο)
    - Είναι 0 σε όλα τα bytes εκτός από το τελευταίο, όπου είναι 1
  - Χρησιμοποιείται για να σηματοδοτήσει το τελευταίο byte της κωδικοποίησης

Κωδικοί μεταβλητών Byte (Variable Byte (VB) codes)

- Ξεκίνα με ένα byte για την αποθήκευση του *G*
- Αν *G* ≤127, υπολόγισε τη δυαδική αναπαράσταση με τα 7 διαθέσιμα bits and θέσε *c* =1
- Αλλιώς, κωδικοποίησε τα 7 lower-order bits του G και χρησιμοποίησε επιπρόσθετα bytes για να κωδικοποιήσεις τα higher order bits με τον ίδιο αλγόριθμο
- Στο τέλος, θέσε το bit συνέχισης του τελευταίου byte σε 1, c = 1 και στα άλλα σε 0, *c* = 0.

Παράδειγμα

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stored as the byte concatenation

000001101011100010000101000011010000110010110001

Key property: VB-encoded postings are uniquely *prefix-decodable*.

824

1100111000

For a small gap (5), VB uses a whole byte.

Άλλες κωδικοποιήσεις

- Αντί για bytes, δηλαδή 8 bits, άλλες μονάδες πχ 32 bits (words), 16 bits, 4 bits (nibbles).

Compression ratio vs speed of decompression

  - Με byte χάνουμε κάποιο χώρο αν πολύ μικρά διάκενα – nibbles καλύτερα σε αυτές τις περιπτώσεις.
  - Μικρές λέξεις, πιο περίπλοκος χειρισμός

- Οι κωδικοί VB χρησιμοποιούνται σε πολλά εμπορικά/ερευνητικά συστήματα

Συμπίεση του RCV1

Data structure	Size in MB
dictionary, fixed-width	11.2
dictionary, term pointers into string	7.6
with blocking, k = 4	7.1
with blocking & front coding	5.9
collection (text, xml markup etc)	3,600.0
collection (text)	960.0
Term-doc incidence matrix	40,000.0
postings, uncompressed (32-bit words)	400.0
postings, uncompressed (20 bits)	250.0
postings, variable byte encoded	116.0
postings, γ–encoded	*101.0*

Συμπεράσματα

- Μπορούμε να κατασκευάσουμε ένα ευρετήριο για Boolean ανάκτηση πολύ αποδοτικό από άποψη χώρου
- Μόνο 4% του συνολικού μεγέθους της συλλογής
- Μόνο το 10-15% του συνολικού κειμένου της συλλογής
- Βέβαια, έχουμε αγνοήσει την πληροφορία θέσης (positional indexes)
  - Η εξοικονόμηση χώρου είναι μικρότερη στην πράξη
  - Αλλά, οι τεχνικές είναι παρόμοιες – χρησιμοποίηση gaps και για τις θέσεις στο έγγραφο

ΤΕΛΟΣ 5ου Κεφαλαίου

Ερωτήσεις?

Χρησιμοποιήθηκε κάποιο υλικό των:

- ✓ *Pandu Nayak and Prabhakar Raghavan, CS276: Information Retrieval and Web Search (Stanford)*