

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΑΝΟΙΚΤΑ ΑΚΑΔΗΜΑΪΚΑ ΜΑΘΗΜΑΤΑ

ανοικτάμαθήματα
opencourses

Ανάκτηση Πληροφορίας

Στατιστικά Συλλογής,
Συμπίεση Ευρετηρίου

Διδάσκων: Καθ. Ε. Πιτουρά

CC BY SA

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

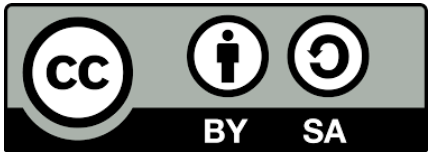
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Αδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Introduction to

Information Retrieval

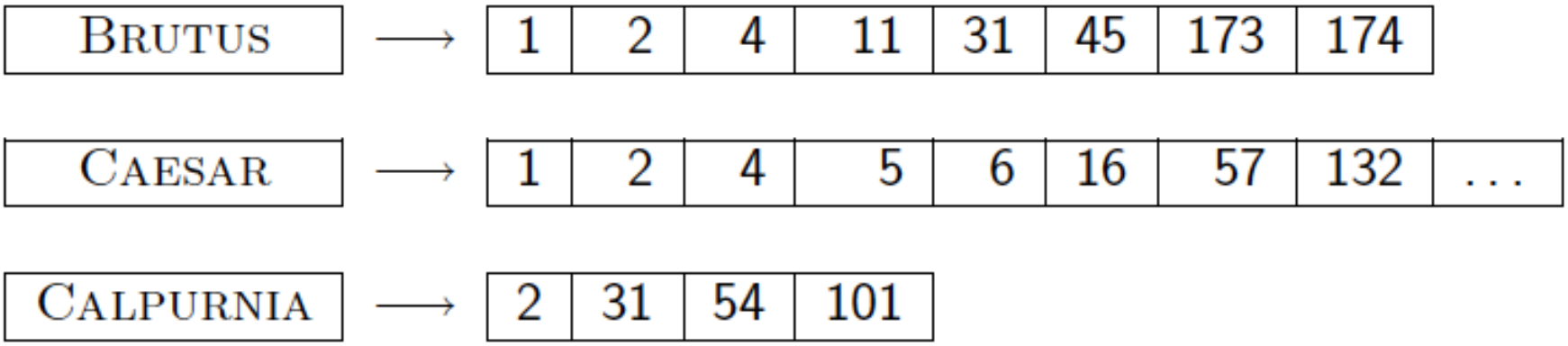
ΠΛΕ70: Ανάκτηση Πληροφορίας

Διδάσκουσα: Ευαγγελία Πιτουρά

Διάλεξη 6: Στατιστικά Συλλογής, Συμπίεση Ευρετηρίου

ΣΤΑΤΙΣΤΙΚΑ ΣΥΛΛΟΓΗΣ

Στατιστικά στοιχεία



- Πιο λεπτομερή στατιστικά για τη συλλογή RCV1
  - Πόσο μεγάλο είναι το λεξικό και οι καταχωρίσεις;

Στατιστικά για τη συλλογή Reuters RCV1

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

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

- Πόσο μεγάλο είναι το λεξιλόγιο όρων;
  - Δηλαδή, πόσες είναι οι διαφορετικές λέξεις;
- Υπάρχει κάποιο άνω όριο;

Π.χ., το Oxford English Dictionary 600,000 λέξεις, αλλά στις πραγματικά μεγάλες συλλογές έχουμε και ονόματα προσώπων, προϊόντων, κλπ.

✓ Στην πραγματικότητα, το λεξιλόγιο συνεχίζει να μεγαλώνει με το μέγεθος της συλλογής.

Λεξιλόγιο και μέγεθος συλλογής

Ο **νόμος του Heaps**:

$$M = kT^b$$

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

περιγράφει πως μεγαλώνει το λεξιλόγιο όσο μεγαλώνει η συλλογή.

- Συνηθισμένες 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 κλίση περίπου ½.

Heaps' Law

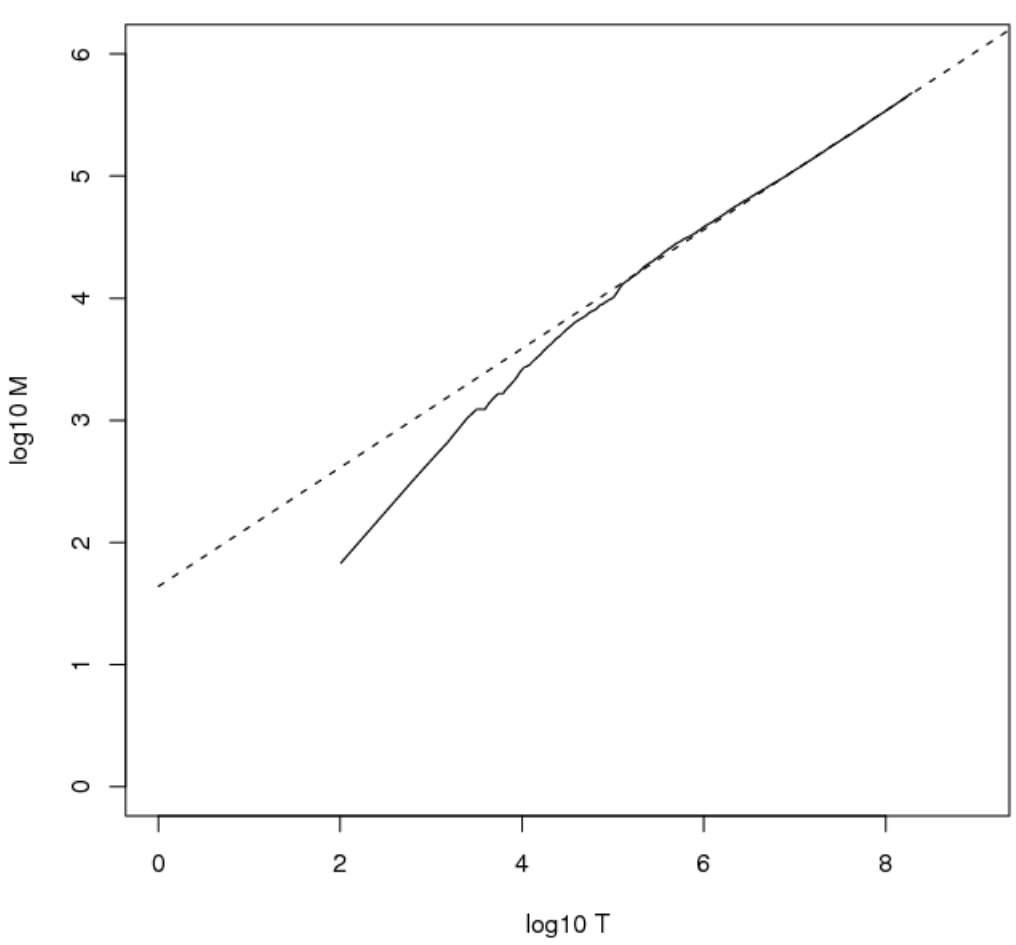
Για το RCV1, η διακεκομμένη γραμμή

$\log_{10} M = 0.49 \log_{10} T + 1.64$ είναι το 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 τα πρώτα 1,000,020 tokens, ο νόμος προβλέπει 38,323 όρους, στην πραγματικότητα 38,365.



Ο νόμος του Zipf

- ✓ Ο νόμος του Heaps μας δίνει το μέγεθος του λεξιλογίου μιας 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* .

$cf_i \propto 1/i = K/i$ όπου K μια normalizing constant

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

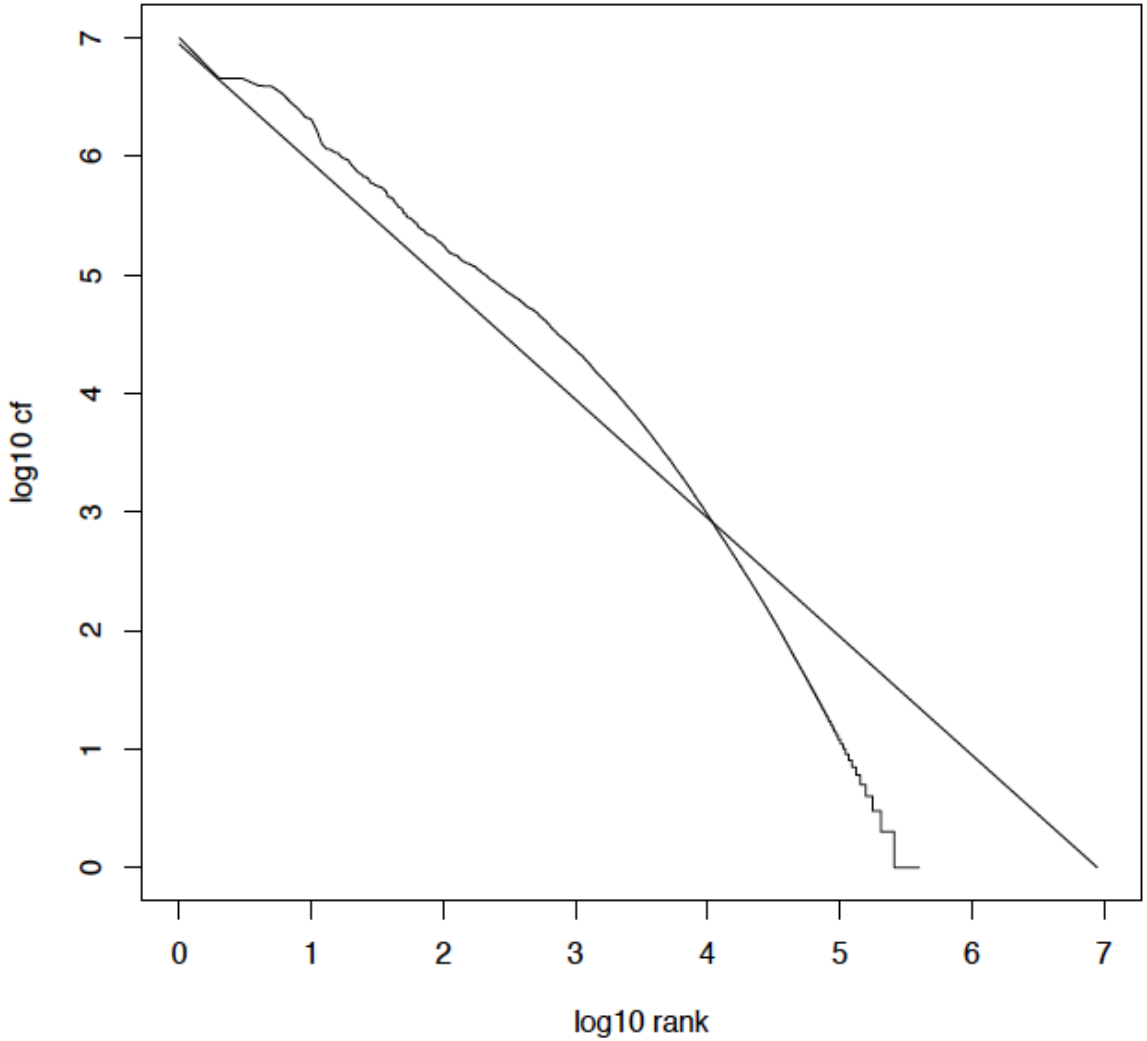
- Αν ο πιο συχνός όρος (ο όρος *the*) εμφανίζεται cf_1 φορές,
- τότε ο δεύτερος πιο συχνός (*of*) εμφανίζεται $cf_1/2$ φορές,
- ο τρίτος (*and*) $cf_1/3$ φορές …

$\log cf_i = \log K - \log i$

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

power law σχέση (εκθετικός νόμος)

Zipf's law for Reuters RCV1



Τι θα δούμε σήμερα

- Συμπίεση Ευρετηρίου

Συμπίεση

- Θα δούμε μερικά θέματα για τη συμπίεση το λεξιλογίου και των καταχωρίσεων.
- Βασικό Boolean 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 και αποσυμπίεσε] γρηγορότερο από [διάβασε μη συμπιεσμένα δεδομένα]
  - Προϋπόθεση: Γρήγοροι αλγόριθμοι αποσυμπίεσης.

Συμπίεση

- Λεξικό
  - Αρκετά μικρό για να το έχουμε στην κύρια μνήμη
  - Ακόμα μικρότερο ώστε να έχουμε επίσης και κάποιες καταχωρίσεις στην κύρια μνήμη.
- Αρχείο Καταχωρίσεων
  - Μείωση του χώρου στο δίσκο
  - Μείωση του χρόνου που χρειάζεται για να διαβάσουμε τις λίστες καταχωρίσεων από το δίσκο
  - Οι μεγάλες μηχανές αναζήτησης διατηρούν ένα μεγάλο τμήμα των καταχωρίσεων στη μνήμη.

Lossless vs. lossy συμπίεση

- Lossless compression: (μη απωλεστική συμπίεση) Διατηρείται 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 lossy compression
  - Μπορεί να είναι αποδεκτή στην περίπτωση π.χ., που μας ενδιαφέρουν μόνο τα κορυφαία από τα σχετικά έγγραφα.

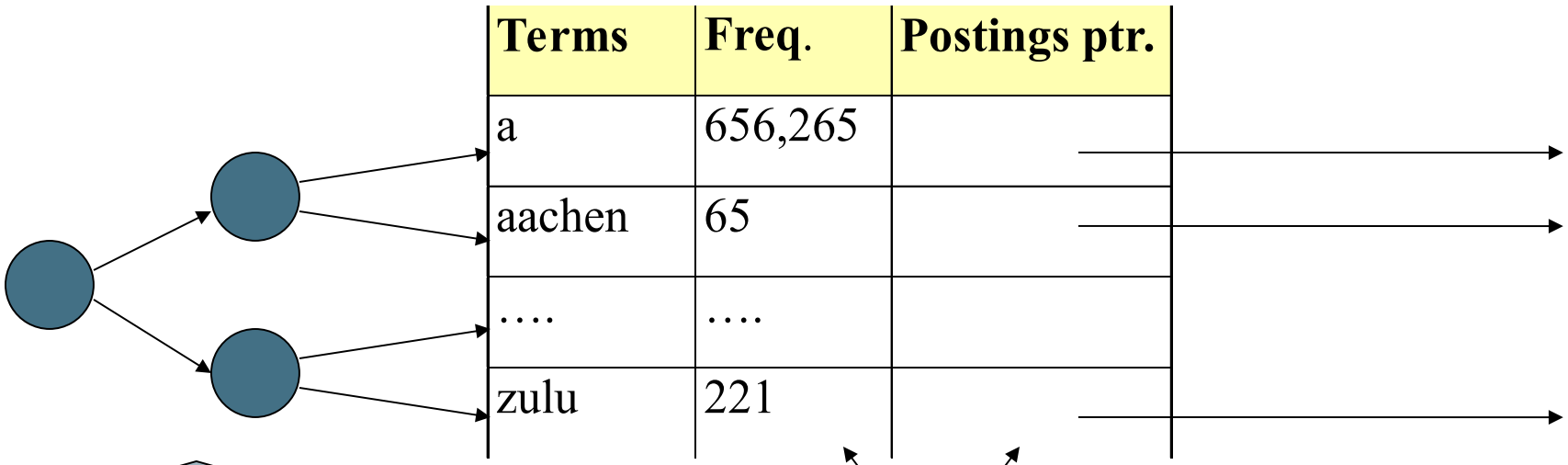
ΣΥΜΠΙΕΣΗ ΛΕΞΙΚΟΥ

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 δεν το έχουμε στη μνήμη, θα θέλαμε να είναι μικρό για γρήγορη έναρξη της αναζήτησης.

Αποθήκευση λεξικού

- Το πιο απλό: ως πίνακας εγγραφών σταθερού μεγέθους (array of fixed-width entries)
  - ~400,000 όροι; 28 bytes/term = 11.2 MB.



Αποθήκευση λεξικού

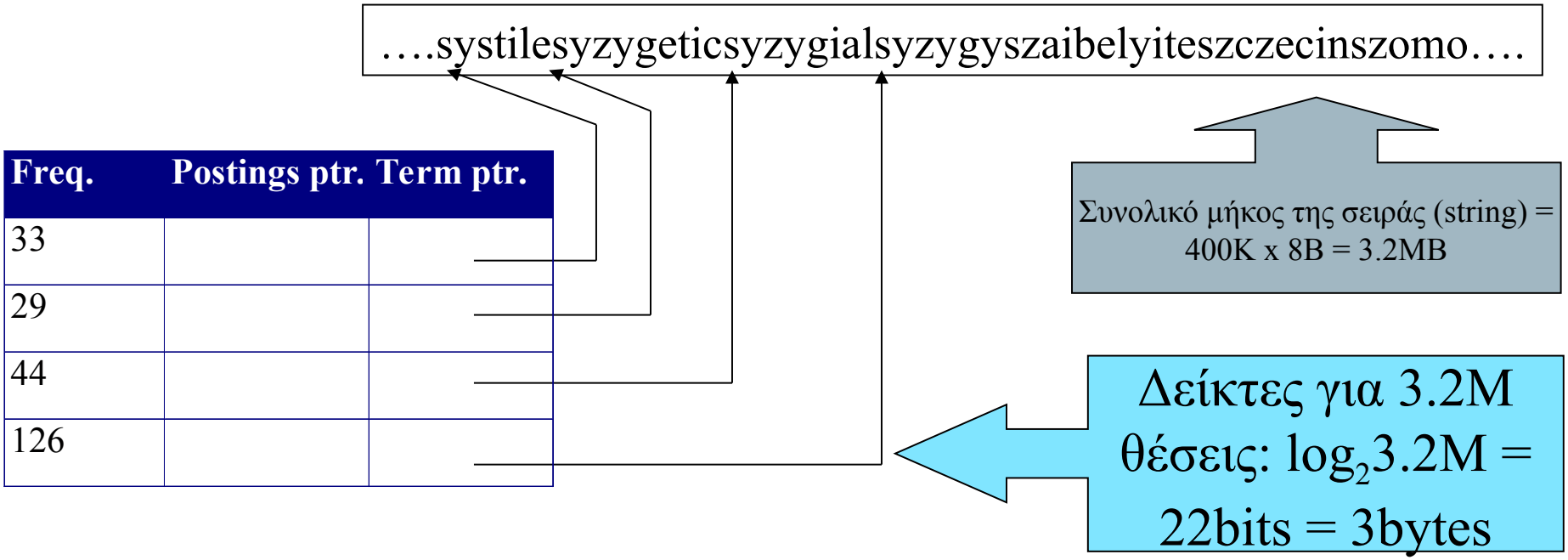
Σπατάλη χώρου

- Πολλά από τα bytes στη στήλη **Terms** δεν χρησιμοποιούνται - δίνουμε 20 bytes για όρους με 1 γράμμα
  - Και δε μπορούμε να χειριστούμε το *supercalifragilisticexpialidocious* ή το *hydrochlorofluorocarbons.*
- 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:

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



Χώρος για το λεξικό ως string

- 4 bytes per term for Freq.
- 4 bytes per term for pointer to Postings.
- 3 bytes per term pointer

} Now avg. 11 bytes/term

- Avg. 8 bytes per term in term string (3.2MB)
- 400K terms x 19 ⇒ 7.6 MB (against 11.2MB for fixed width).

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)



Freq.	Postings ptr.	Term ptr.
33		
29		
44		
126		
7		

Save 9 bytes on 3 pointers.

Lose 4 bytes on term lengths.

Blocking

Συνολικό όφελος για block size $k = 4$

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

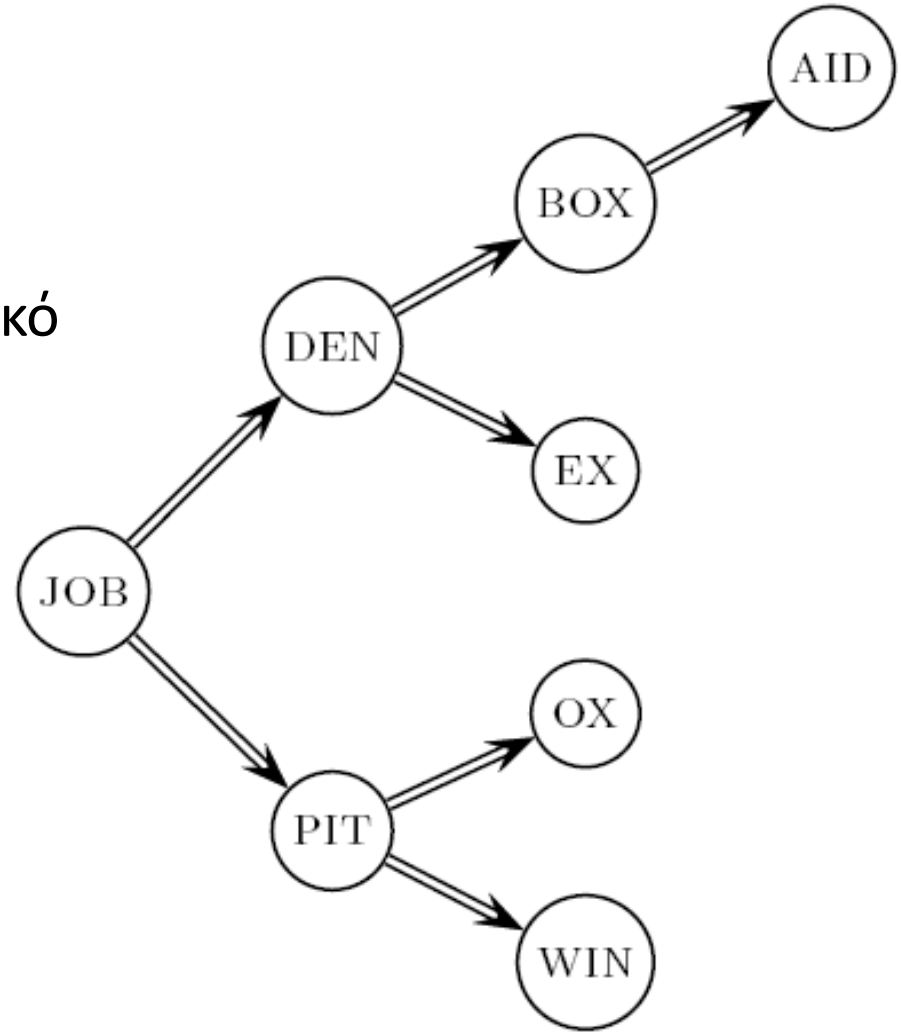
Τώρα 3 + 4 = 7 bytes.

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

- Γιατί όχι ακόμα μικρότερο 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.



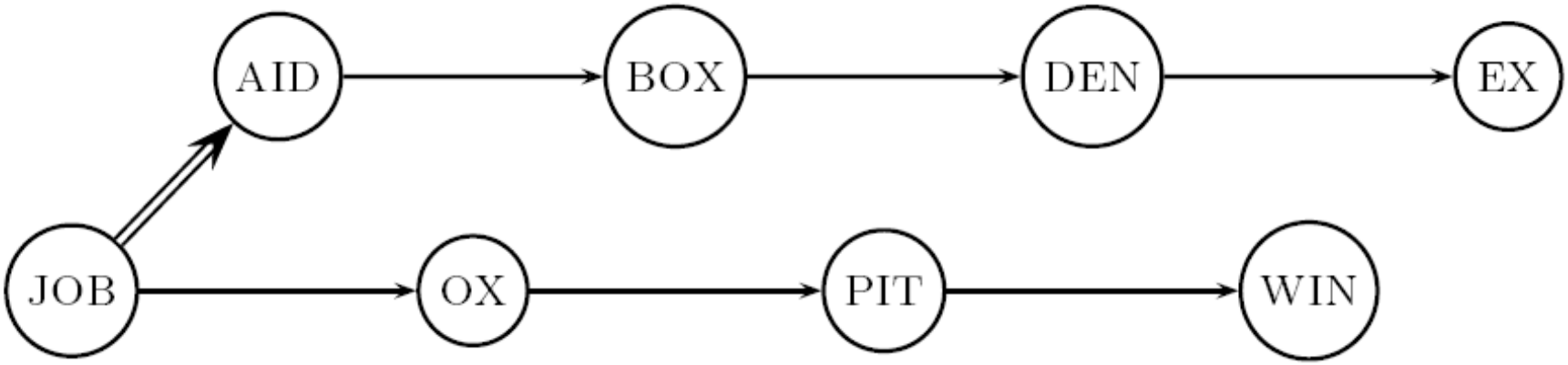
Ασκηση: σκεφτείτε ένα καλύτερο τρόπο αναζήτησης αν δεν έχουμε ομοιόμορφη κατανομή των όρων.

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 (δυαδικό δέντρο)= (1+2·2+2·3+2·4+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.***

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 ΚΑΤΑΧΩΡΙΣΕΩΝ

Συμπίεση των καταχωρίσεων

- Το αρχείο των καταχωρίσεων είναι πολύ μεγαλύτερο από το αρχείο του λεξικού - τουλάχιστον 10 φορές.
- Βασική επιδίωξη: *αποθήκευση κάθε καταχώρισης συνοπτικά.*
- Στην περίπτωση μας, μια καταχώριση 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;

Συμπίεση των καταχωρίσεων

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

- Γιατί; Τα περισσότερα κενά 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								

Συμπίεση των καταχωρίσεων

- Ενας όρος όπως ***arachnocentric*** εμφανίζεται ίσως σε ένα έγγραφο στο εκατομμύριο.
- Ενας όρος όπως ***the*** εμφανίζεται σχεδόν σε κάθε έγγραφο, άρα τα 20 bits/εγγραφή είναι πολλά!

Κωδικοποίηση μεταβλητού μεγέθους (Variable length encoding)

Στόχος:

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

- Αν το μέσο gap για έναν όρο είναι *G*, θέλουμε να χρησιμοποιήσουμε ~$\log_2 G$ 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.

Unary code

- Αναπαράστησε το n ως n 1 με ένα 0 στο τέλος.
- Unary code για το 3 είναι το 1110.
- Unary code για το 40 είναι το

1111111111111111111111111111111111111110 .

- Unary code για το 80 είναι το:

11111111111111111111111111111111111111111111
11111111111111111111111111111110

Δεν δείχνει πολλά υποσχόμενος, αλλά...
Είναι βέλτιστος όταν $P(n) = 2^{-n}$

- Μπορούμε να τον αξιοποιήσουμε στη λύση μας

Gamma codes

- Πετυχαίνουμε καλύτερη συμπίεση με bit-level codes
  - Ο Gamma code είναι ο πιο γνωστός τέτοιος κώδικας.
- Αναπαράστησε ένα gap *G* ως ένα ζεύγος *length* και *offset*
- *offset* είναι το *G* σε δυαδική μορφή χωρίς το πρώτο bit (most significant – πάντα άσσος)
  - Για παράδειγμα 13 → 1101 → 101
- *length* είναι το μήκος του offset
  - Για το 13 (offset 101), είναι 3.
- Κωδικοποιούμε το *length* με *unary code*: 3=1110.
- Gamma code για το 13 είναι η ένωση (concatenation) των *length* και *offset*: 1110101

Παραδείγματα Gamma code

number	length	offset	γ-code
0			none
1	0		0
2	10	0	10,0
3	10	1	10,1
4	110	00	110,00
9	1110	001	1110,001
13	1110	101	1110,101
24	11110	1000	11110,1000
511	111111110	11111111	111111110,11111111
1025	11111111110	0000000001	11111111110,0000000001

Ιδιότητες του Gamma code

- Το G κωδικοποιείται με $2\lfloor \log G \rfloor + 1$ bits
  - Μήκος offset $\lfloor \log G \rfloor$ bits
  - Μήκος length $\lfloor \log G \rfloor + 1$ bits
- Ολοι οι gamma κώδικες έχουν περιττό αριθμό bits
- Σχεδόν διπλάσιο μέγεθος από το καλύτερο δυνατό, $\log_2 G$
- Ο Gamma code είναι **uniquely prefix-decodable**
- Μπορεί να χρησιμοποιηθεί για οποιαδήποτε κατανομή
  - Βέλτιστος όταν $P(n) \approx 1/(2n^2)$
- Ο Gamma code είναι parameter-free

Ο Gamma code σπάνια χρησιμοποιείται στην πράξη

- Οι υπολογιστές έχουν word boundaries – 8, 16, 32, 64 bits
  - Πράξεις που διασχίζουν (cross) word boundaries είναι πιο αργές
- Γενικά όταν η συμπίεση διαχειρίζεται δεδομένα στο επίπεδο των bits μπορεί να είναι πολύ αργή
- Η σύγχρονη πρακτική είναι να χρησιμοποιούνται byte or word aligned codes
  - Το Variable byte encoding είναι γρηγορότερο, πιο απλό ως έννοια και με ικανοποιητική συμπίεση

Κωδικοί μεταβλητών Byte (Variable Byte (VB) codes)

- Κωδικοποιούμε κάθε gap με ακέραιο αριθμό από bytes.
- Το πρώτο bit κάθε byte χρησιμοποιείται ως bit συνέχισης c (continuation bit)
  - Είναι c=0 σε όλα τα bytes εκτός από το τελευταίο, όπου είναι c=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.

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

Αλλες κωδικοποιήσεις

- Αντί για bytes, άλλες μονάδες πχ 32 bits (words), 16 bits, 4 bits (nibbles).
- Με byte χάνουμε κάποιο χώρο αν τα gaps είναι πολύ μικρά – τα nibbles είναι καλύτερα σε αυτές τις περιπτώσεις.
- 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*

Περίληψη

- 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
  - Η εξοικονόμηση χώρου είναι μικρότερη στην πράξη
  - Αλλά, οι τεχνικές είναι παρόμοιες.

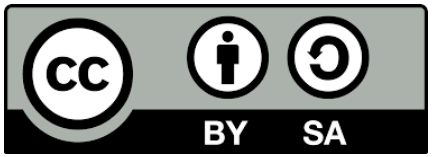
ΤΕΛΟΣ 6ου Μαθήματος

Ερωτήσεις;

Χρησιμοποιήθηκε κάποιο υλικό των:

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

Τέλος Ενότητας



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Ιωαννίνων**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ενωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Εργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Εχουν προηγηθεί οι κάτωθι εκδόσεις:

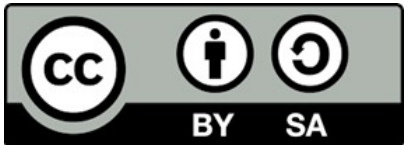
- Εκδοση 1.0 διαθέσιμη εδώ. http://ecourse.uoi.gr/course/view.php?id=1210.

Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Ιωαννίνων, Διδάσκων: Καθ. Ευαγγελία Πιτουρά. «Ανάκτηση Πληροφορίας. Συμπίεση Ευρετηρίου». Εκδοση: 1.0. Ιωάννινα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: http://ecourse.uoi.gr/course/view.php?id=1210.

Σημείωμα Αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Παρόμοια Διανομή, Διεθνής Εκδοση 4.0 [1] ή μεταγενέστερη.



- [1] https://creativecommons.org/licenses/by-sa/4.0/.