Introduction to

Information Retrieval

ΠΛΕ70: Ανάκτηση Πληροφορίας

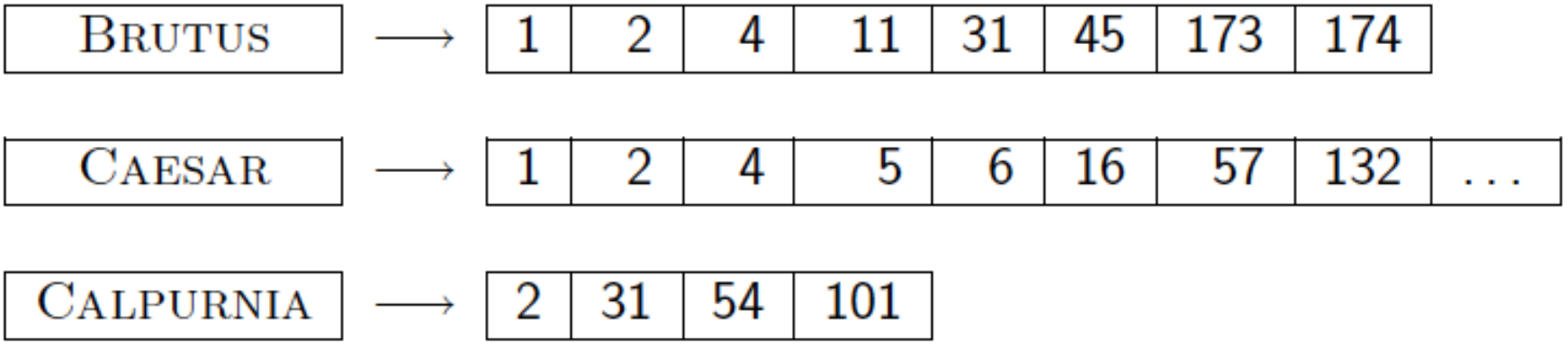
Διδάσκουσα: Ευαγγελία Πιτουρά

Διάλεξη 5(α): Συμπίεση Ευρετηρίου

ΣΤΑΤΙΣΤΙΚΑ ΣΥΛΛΟΓΗΣ

Στατιστικά στοιχεία

- Πόσο μεγάλο είναι το λεξικό και οι καταχωρήσεις;



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 ονόματα προσώπων, προϊόντων, κλπ

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

Λεξιλόγιο και μέγεθος συλλογής

Ο **νόμος του Heaps**:

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

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

Heaps' Law

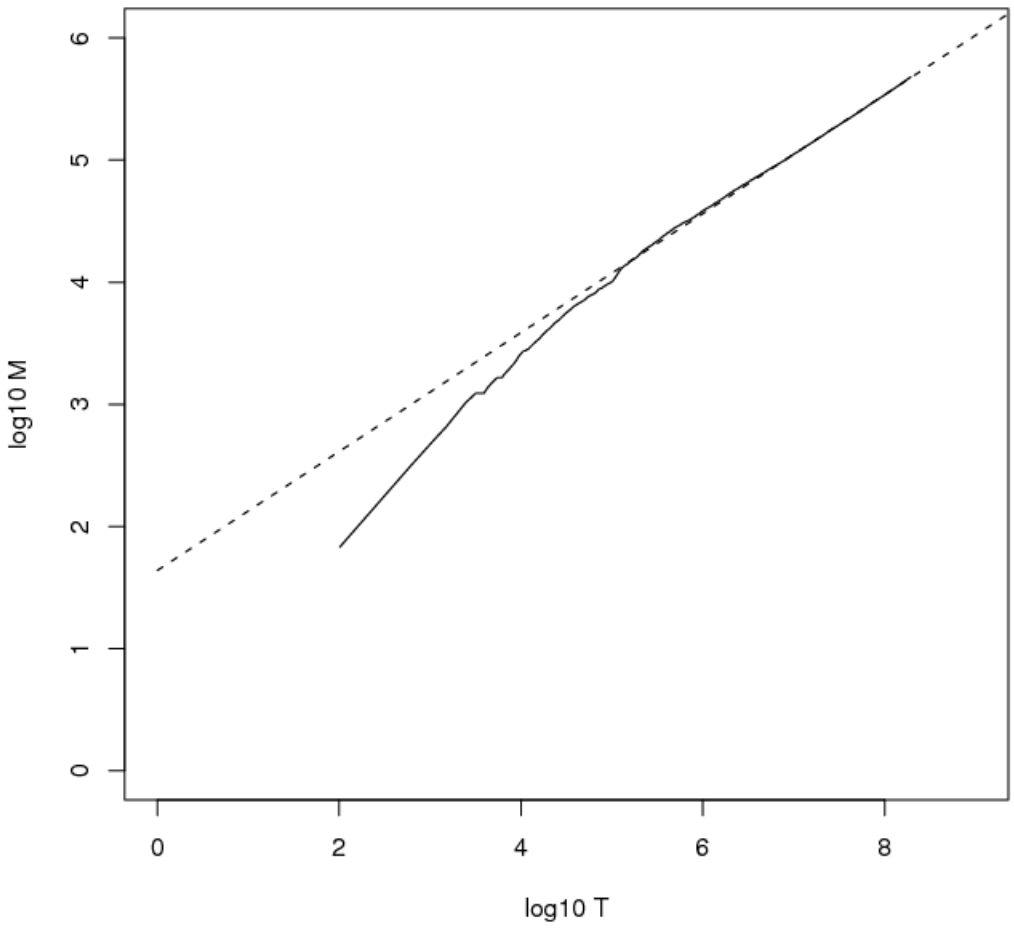
Για το RCV1, η διακεκομμένη γραμμή

$\log_{10} M = 0.49 \log_{10} T + 1.64$ (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:

- Stemming
- Including numbers
- Spelling errors
- Case folding

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* .

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

Όπου 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
- …

Ο νόμος του Zipf

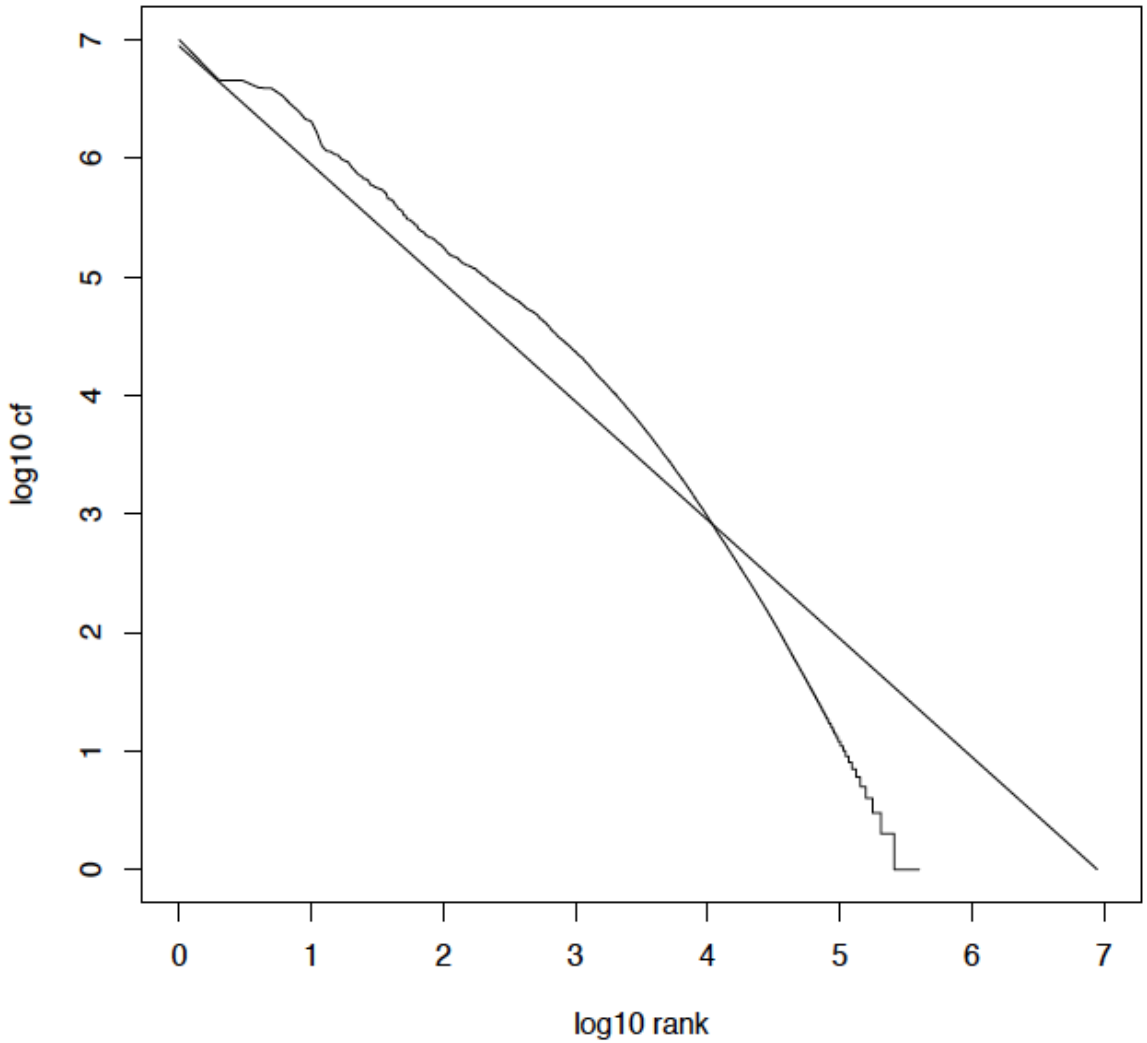
$\log cf_i = \log c - \log i$

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

$cfi = c\ i^K$, $k = -1$

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 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 lossy compression
  - Μπορεί να είναι αποδεκτή στην περίπτωση π.χ., που μας ενδιαφέρουν μόνο τα κορυφαία από τα σχετικά έγγραφα

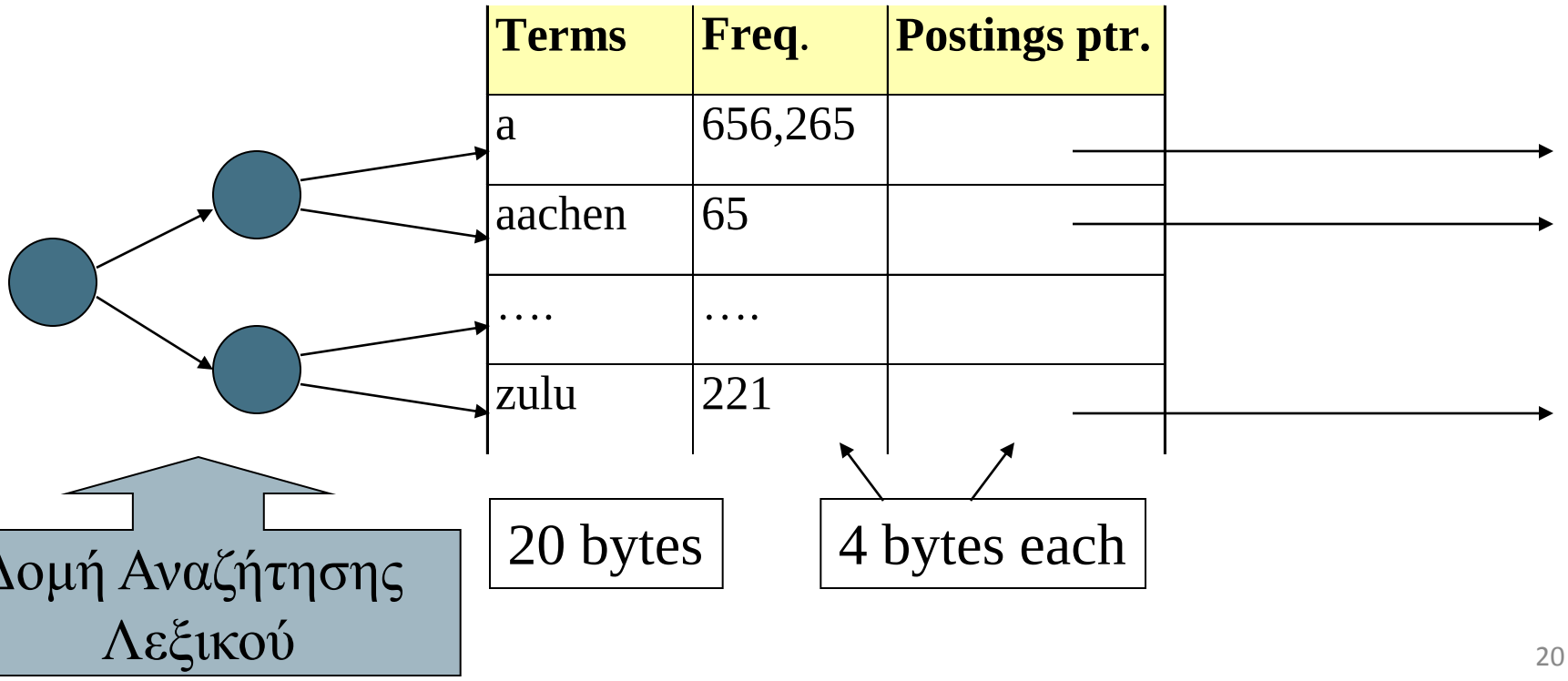
ΣΥΜΠΙΕΣΗ ΛΕΞΙΚΟΥ

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 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.



Αποθήκευση λεξικού

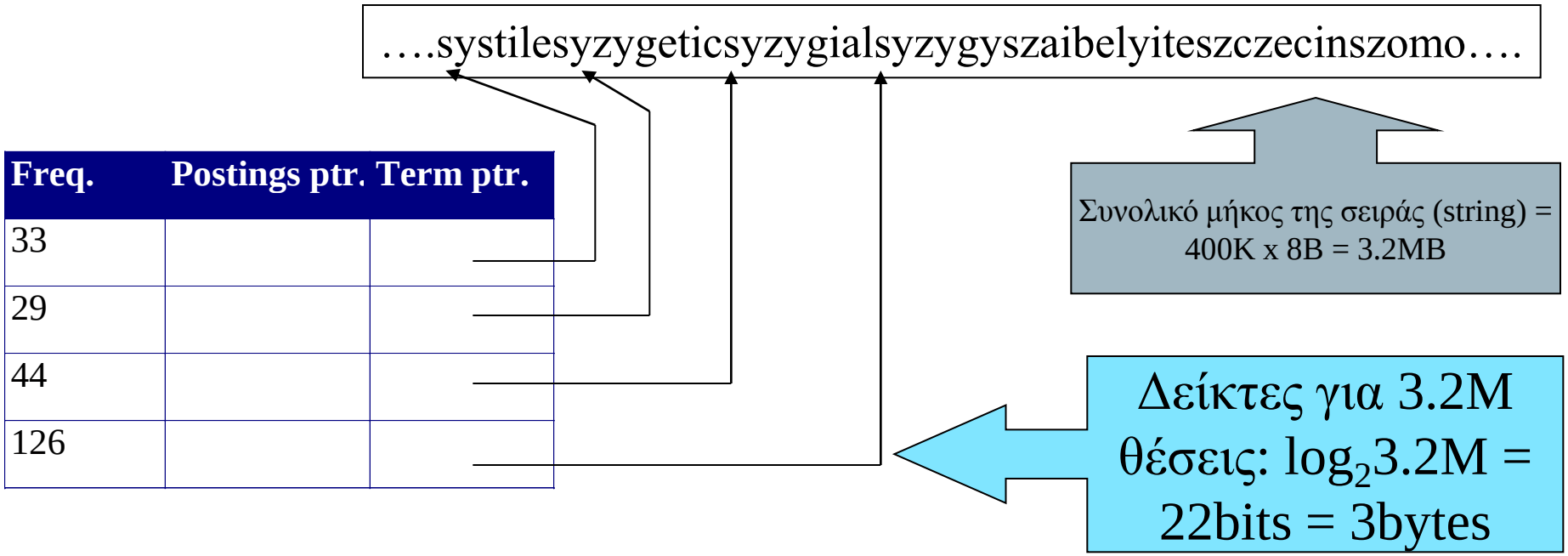
Σπατάλη χώρου

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

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



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 $\Rightarrow$ 7.6 MB (έναντι 11.2MB για σταθερό μήκος λέξης)

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		

Κερδίζουμε 3 bytes για k - 1 δείκτες.

Χάνουμε 4 (k) bytes για το μήκος του όρου

Blocking

Συνολικό όφελος για block size $k = 4$

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

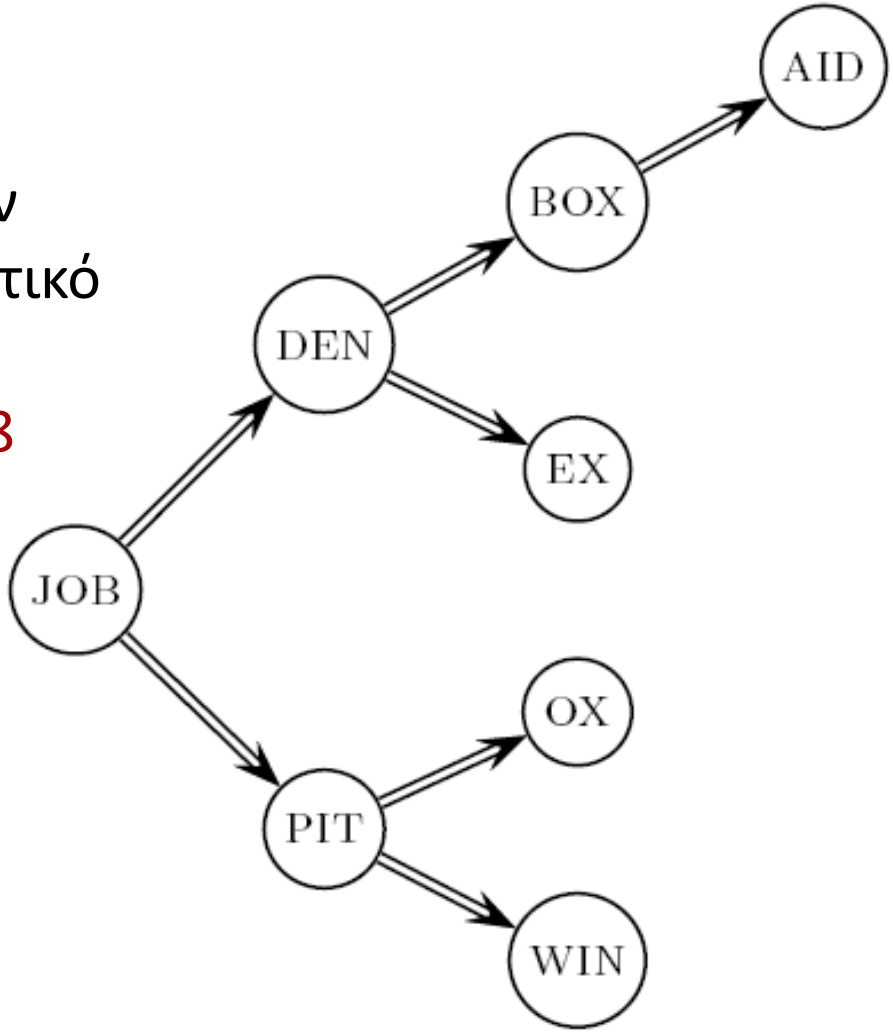
Τώρα 3 + 4 = 7 bytes.

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



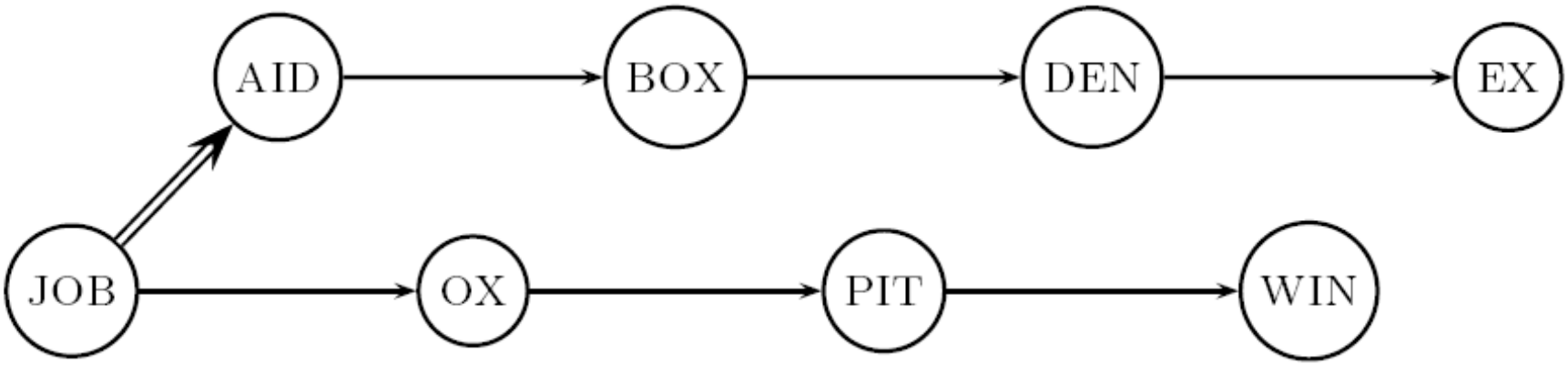
Άσκηση: σκεφτείτε ένα καλύτερο τρόπο αναζήτησης αν δεν έχουμε ομοιόμορφη κατανομή των όρων

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 (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

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

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;

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, 47,1 54, 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 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)
  - Είναι 0 σε όλα τα bytes εκτός από το τελευταίο, όπου είναι 1
    - 0, αν ακολουθεί και δεύτερο byte
    - 1, αλλιώς
  - 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*

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 πρώτου μέρους 5ου Μαθήματος

Ερωτήσεις?

Χρησιμοποιήθηκε κάποιο υλικό των:

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

Introduction to

Information Retrieval

ΠΛΕ70: Ανάκτηση Πληροφορίας

Διδάσκουσα: Ευαγγελία Πιτουρά

Διάλεξη 5(β) : Βαθμολόγηση. Στάθμιση όρων. Το μοντέλο διανυσματικού χώρου.

Τι άλλο θα δούμε σήμερα;

- Βαθμολόγηση και κατάταξη εγγράφων
- Στάθμιση όρων (term weighting)
- Αναπαράσταση εγγράφων και ερωτημάτων ως διανύσματα

Κατάταξη εγγράφων (Ranked retrieval)

- Μέχρι τώρα, τα ερωτήματα που είδαμε ήταν Boolean.
  - Τα έγγραφα είτε ταιριάζουν στο ερώτημα, είτε όχι

- Τα Boolean ερωτήματα συχνά έχουν είτε *πολύ λίγα* (=0) είτε *πάρα πολλά* (χιλιάδες) αποτελέσματα ("feast or famine")
  - Ερώτημα 1: "*standard user dlink 650*" → 200,000 hits
  - Ερώτημα 2: "*standard user dlink 650 no card found*": 0 hits
- Χρειάζεται επιδεξιότητα για να διατυπωθεί μια ερώτηση που έχει ως αποτέλεσμα ένα διαχειρίσιμο αριθμό ταιριασμάτων
  - AND πολύ λίγα - OR πάρα πολλά

Διάταξη εγγράφων (Ranked retrieval)

- Κατάλληλη *για ειδικούς* με σαφή κατανόηση των αναγκών τους και της συλλογής
  - Επίσης, καλή *για εφαρμογές*: οι εφαρμογές μπορούν να επεξεργαστούν χιλιάδες αποτελεσμάτων.
- Αλλά, όχι κατάλληλη για την πλειοψηφία των χρηστών
  - Είναι δύσκολο για τους περισσότερους χρήστες να διατυπώσουν Boolean ερωτήματα
  - Οι περισσότεροι χρήστες δεν θέλουν να διαχειριστούν χιλιάδες αποτελέσματα.
    - Ιδιαίτερα στην περίπτωση των αναζητήσεων στο web

Μοντέλα διαβαθμισμένης ανάκτησης

- Αντί ενός **συνόλου** εγγράφων που ικανοποιούν το ερώτημα, η ***διαβαθμισμένη ανάκτηση (ranked retrieval)*** επιστρέφει μια **διάταξη** των (κορυφαίων) για την ερώτηση εγγράφων της συλλογής
- Όταν το σύστημα παράγει ένα διατεταγμένο σύνολο αποτελεσμάτων, τα μεγάλα σύνολα δεν αποτελούν πρόβλημα
  - Δείχνουμε απλώς τα ***κορυφαία*** (top) *k* (≈ 10) αποτελέσματα
  - Δεν παραφορτώνουμε το χρήστη

Προϋπόθεση: ο αλγόριθμος διάταξης να δουλεύει σωστά

Μοντέλα διαβαθμισμένης ανάκτησης

- Αν και διαφορετικά θέματα, η διαβαθμισμένη ανάκτηση συνήθως με ερωτήματα ελεύθερου κειμένου
  - Ερωτήματα ελεύθερου κειμένου (Free text queries): Μία ή περισσότερες λέξεις σε μια φυσική γλώσσα (αντί για μια γλώσσα ερωτημάτων με τελεστές και εκφράσεις)

Βαθμολόγηση ως βάση της διαβαθμισμένης ανάκτησης

- Θέλουμε να επιστρέψουμε τα αποτελέσματα διατεταγμένα με βάση *το πόσο πιθανό είναι να είναι χρήσιμα στο χρήστη*
- Πως διατάσουμε-διαβαθμίζουμε τα έγγραφα μιας συλλογής με βάση ένα ερώτημα
  - Αναθέτουμε ένα βαθμό (score) – ας πούμε στο [0, 1] – σε κάθε έγγραφο
- Αυτός ο βαθμός μετρά πόσο καλά το έγγραφο *d* "ταιριάζει" (match) με το ερώτημα *q*

Βαθμός ταιριάσματος ερωτήματος-εγγράφου

- Χρειαζόμαστε ένα τρόπο για να αναθέσουμε ένα βαθμό σε κάθε ζεύγος ερωτήματος(q)/εγγράφου(d)

  score(d, q)

- ✓ Αν ο όρος του ερωτήματος δεν εμφανίζεται στο έγγραφο, τότε ο βαθμός θα πρέπει να είναι 0
- ✓ Όσο *πιο συχνά* εμφανίζεται ο όρος του ερωτήματος σε ένα έγγραφο, *τόσο μεγαλύτερος* θα πρέπει να είναι ο βαθμός

- Θα εξετάσουμε κάποιες εναλλακτικές για αυτό

Προσπάθεια 1: Συντελεστής Jaccard

Υπενθύμιση: συνηθισμένη μέτρηση της επικάλυψης δύο συνόλων A και B

$$\text{jaccard}(A,B) = |A \cap B| \,/\, |A \cup B|$$

- jaccard$(A,A) = 1$
- jaccard$(A,B) = 0$ if $A \cap B = 0$

- *Τα* A και B δεν έχουν απαραίτητα το ίδιο μέγεθος
- Αναθέτει πάντα έναν αριθμό μεταξύ του 0 και του 1

Συντελεστής Jaccard: Παράδειγμα βαθμολόγησης

- Ποιος είναι ο βαθμός ταιριάσματος ερωτήματος-εγγράφου με βάση το συντελεστή Jaccard για τα παρακάτω;
  - Ερώτημα (q): *ides of march*
  - Έγγραφο 1 (d1): *caesar died in march*
  - Έγγραφο 2 (d2): *the long march*

Θα δούμε και έναν πιο πλήρη τρόπο κανονικοποιήσης του μήκους:

$$|A \cap B| / \sqrt{|A \cup B|}$$

Παράδειγμα

Ποιο είναι ο βαθμός για τα παρακάτω ζεύγη χρησιμοποιώντας jaccard;

q: [information on cars]

d: “all you’ve ever wanted to know about cars”

q: [information on cars]

d: “information on trucks, information on planes, information on trains”

q: [red cars and red trucks]

d: “cops stop red cars more often”

Συχνότητα όρου - Term frequency (tf)

- Η συχνότητα όρου $tf_{t,d}$ του όρου t σε ένα έγγραφο d ορίζεται ως ο αριθμός των φορών που το t εμφανίζεται στο d (το πλήθος των εμφανίσεων του όρου t στο έγγραφο d)

Παράδειγμα

Ποιο είναι ο βαθμός για τα παρακάτω ζεύγη χρησιμοποιώντας tf;

q: [information on cars]

d: “all you’ve ever wanted to know about cars”

q: [information on cars]

d: “information on trucks, information on planes, information on trains”

q: [red cars and red trucks]

d: “cops stop red cars more often”

Προβλήματα

- Jaccard δεν λαμβάνει υπ’ όψιν την *συχνότητα όρου* (*term frequency*): πόσες φορές εμφανίζεται ο όρος στο έγγραφο

- Αγνοεί το γεγονός πως οι *σπάνιοι όροι* περιέχουν περισσότερη πληροφορία από ό,τι οι συχνοί.

Συχνότητα εγγράφου (Document frequency)

Οι σπάνιοι όροι παρέχουν περισσότερη πληροφορία από τους συχνούς όρους

- Θυμηθείτε τα stop words (διακοπτόμενες λέξεις)

- Θεωρείστε έναν όρο σε μια ερώτηση που είναι σπάνιος στη συλλογή (π.χ., *arachnocentric*)
- Το έγγραφο που περιέχει αυτόν τον όρο είναι πιο πιθανό να είναι περισσότερο συναφές με το ερώτημα από ένα έγγραφο που περιέχει ένα λιγότερο σπάνιο όρο του ερωτήματος

→ Θέλουμε να δώσουμε μεγαλύτερο βάρος στους σπάνιους όρους – αλλά πως; **df**

Βάρος idf

- df_t είναι η συχνότητα εγγράφων του t: ο αριθμός (πλήθος) των εγγράφων της συλλογής που περιέχουν το t
  - df_t είναι η αντίστροφη μέτρηση της πληροφορίας που παρέχει ο όρος t
  - $df_t \leq N$
- Ορίζουμε την αντίστροφη συχνότητα εγγράφων idf (inverse document frequency) του t ως

$$\text{idf}_t = N/\text{df}_t$$

Βαθμός εγγράφου και ερώτησης

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

- Μεγάλο για όρους που εμφανίζονται πολλές φορές σε λίγα έγγραφα (μεγάλη *διακριτική δύναμη* (discriminating power) σε αυτά τα έγγραφα
- Μικρότερο όταν ο όρος εμφανίζεται λίγες φορές σε ένα έγγραφο ή όταν εμφανίζεται σε πολλά έγγραφα
- Το μικρότερο για όρους που εμφανίζονται σχεδόν σε όλα τα έγγραφα

- Υπάρχουν πολλές άλλες παραλλαγές
  - Πως υπολογίζεται το “tf” (με ή χωρίς log)
  - Αν δίνεται βάρος και στους όρους του ερωτήματος
  - …

Συχνότητα συλλογής και εγγράφου

- Η *συχνότητα συλλογής* ενός όρου *t* είναι ο αριθμός των εμφανίσεων του *t* στη συλλογή, μετρώντας και τις πολλαπλές εμφανίσεις

Παράδειγμα:

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

- Ποια λέξη είναι καλύτερος όρος αναζήτησης (και πρέπει να έχει μεγαλύτερο βάρος)?

Bag of words model

- Η tf-idf διαβάθμιση δεν εξετάζει τη διάταξη των λέξεων σε ένα έγγραφο
  - *John is quicker than Mary* και
  - *Mary is quicker than John*

  Έχουν τα ίδια διανύσματα
- Αυτό λέγεται μοντέλο σάκου λέξεων (bag of words model) – έχει σημασία ο αριθμός των εμφανίσεων αλλά όχι η διάταξη
- Θα εισάγουμε πληροφορία θέσης *αργότερα*

Στάθμιση tf-idf

Ποιο είναι το idf ενός όρου που εμφανίζεται σε κάθε έγγραφο (ποια η σχέση με stop words);

- tf-idf των παρακάτω όρων:

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

Η επίδραση του idf στη διάταξη

- Το idf δεν επηρεάζει τη διάταξη ερωτημάτων με ένα όρο, όπως
  - iPhone
- Το idf επηρεάζει μόνο τη διάταξη εγγράφων με τουλάχιστον δύο όρους
  - Για το ερώτημα capricious person, η idf στάθμιση έχει ως αποτέλεσμα οι εμφανίσεις του capricious να μετράνε περισσότερο στην τελική διάταξη των εγγράφων από ότι οι εμφανίσεις του person.

(ένα έγγραφο που περιέχει μόνο το capricious είναι πιο σημαντικό από ένα που περιέχει μόνο το person)

Συχνότητα όρου - Term frequency (tf)

- Η συχνότητα όρου $tf_{t,d}$ του όρου t σε ένα έγγραφο d ορίζεται ως ο αριθμός των φορών που το t εμφανίζεται στο d.

Φτάνει μόνο η συχνότητα

- Ένα έγγραφο με 10 εμφανίσεις ενός όρου είναι πιο σχετικό από ένα έγγραφο με 1 εμφάνιση του όρου .. Αλλά είναι 10 φορές πιο σχετικό;

- Η σχετικότητα (relevance) δεν αυξάνει *ανάλογα* με τη συχνότητα όρου

Στάθμιση με Log-συχνότητας

- Η στάθμιση με χρήση του λογάριθμου της συχνότητας (log frequency weight) του όρου *t* στο *d* είναι

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

  - 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.

- Ο βαθμός για ένα ζεύγος εγγράφου-ερωτήματος: άθροισμα όλων των κοινών όρων :

$$score = \sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d}) idf_t$$

  - Ο βαθμός είναι 0 όταν κανένας από τους όρους του ερωτήματος δεν εμφανίζεται στο έγγραφο

Βάρος idf

- Χρησιμοποιούμε log (N/df$_t$) αντί για N/df$_t$ για να «ομαλοποιήσουμε» την επίδραση του idf.

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

Παράδειγμα idf, έστω N = 1 εκατομμύριο

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

- ✓ Κάθε όρος στη συλλογή έχει μια τιμή idf
- ✓ Ολική μέτρηση (επίσης, αλλάζει συνεχώς)

Στάθμιση tf-idf

- Το tf-idf βάρος ενός όρου είναι το γινόμενο του βάρους tf και του βάρους idf.

$$w_{t,d} = (1 + \log_{10} \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Overlap score measure
- Το *πιο γνωστό* σχήμα διαβάθμισης στην ανάκτηση πληροφορίας
  - Εναλλακτικά ονόματα: tf.idf, tf x idf
- Αυξάνει με τον αριθμό εμφανίσεων του όρου στο έγγραφο
- Αυξάνει με τη σπανιότητα του όρου στη συλλογή

Δυαδική μήτρα σύμπτωσης (binary term-document incidence matrix)

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Κάθε έγγραφο αναπαρίσταται ως ένα δυαδικό διάνυσμα $\in \{0,1\}^{|V|}$ (την αντίστοιχη στήλη)

Ο πίνακας με μετρητές

- Θεωρούμε τον tf, αριθμό (πλήθος) των εμφανίσεων ενός όρου σε ένα έγγραφο:
  - Κάθε έγγραφο είναι ένα διάνυσμα μετρητών στο $\mathbb{N}^{|v|}$: μια στήλη παρακάτω

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Ο πίνακας με βάρη

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Θεωρούμε το tf-idf βάρος του όρου:

- Κάθε έγγραφο είναι ένα διάνυσμα tf-idf βαρών στο $R^{|v|}$

Αποθήκευση

- Που υπάρχει αυτή η πληροφορία στο σύστημα ανάκτησης πληροφορίας;

Τα έγγραφα ως διανύσματα (vector space model)

Έχουμε ένα |V|-διάστατο διανυσματικό χώρο

- Οι όροι είναι οι άξονες αυτού του χώρου
- Τα έγγραφα είναι σημεία ή διανύσματα σε αυτόν τον χώρο

- Πολύ μεγάλη διάσταση: δεκάδες εκατομμύρια διαστάσεις στην περίπτωση της αναζήτησης στο web
- Πολύ αραιά διανύσματα – οι περισσότεροι όροι είναι 0

Τα ερωτήματα ως διανύσματα

- Βασική ιδέα 1: Εφαρμόζουμε το ίδιο και για τα ερωτήματα, δηλαδή, *αναπαριστούμε και τα ερωτήματα ως διανύσματα στον ίδιο χώρο*

- Βασική ιδέα 2: Διαβάθμιση των εγγράφων με βάση το πόσο κοντά είναι στην ερώτηση σε αυτό το χώρο
  - Κοντινά = ομοιότητα διανυσμάτων
  - Ομοιότητα ≈ αντίθετο της απόστασης

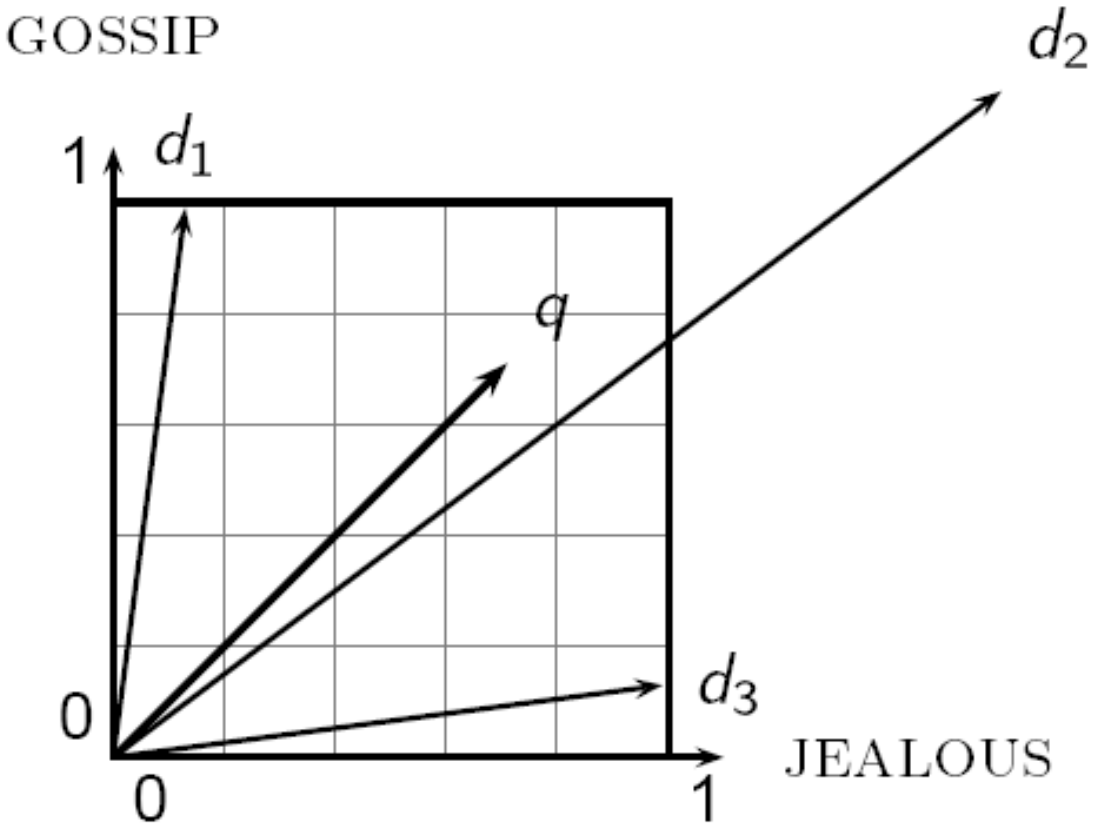
Ομοιότητα διανυσμάτων

Πρώτη προσέγγιση απόστασης μεταξύ δυο διανυσμάτων

- Ευκλείδεια απόσταση;
  - Δεν είναι καλή ιδέα – είναι μεγάλη για διανύσματα διαφορετικού μήκους

Γιατί η απόσταση δεν είναι καλή ιδέα

Η Ευκλείδεια απόσταση μεταξύ του $\vec{q}$ και του $\vec{d_2}$ είναι μεγάλη αν και η κατανομή των όρων είναι παρόμοια



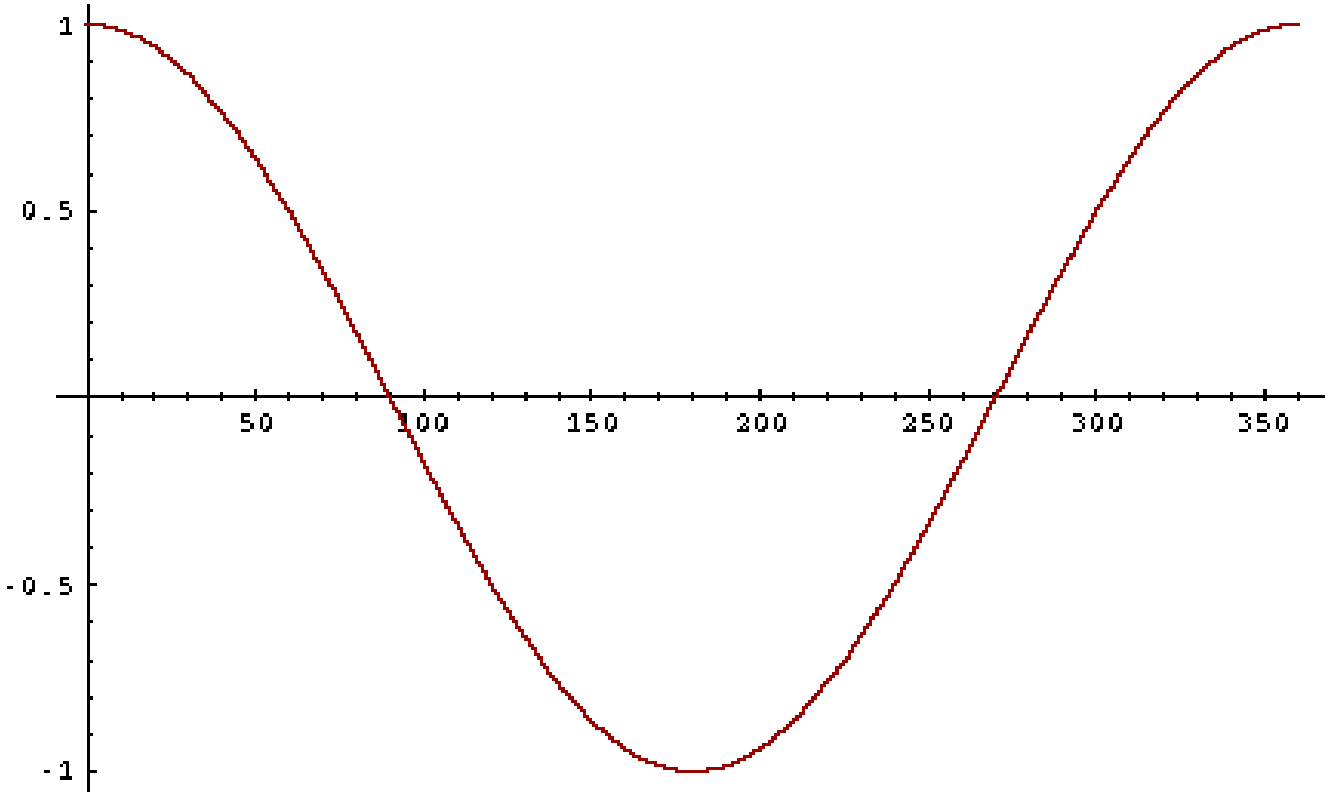
Χρήση της γωνίας αντί της απόστασης

- Έστω ένα έγγραφο d. Ως παράδειγμα, υποθέστε ότι κάνουμε append το d στον εαυτό του και έστω d’ το κείμενο που προκύπτει.
- “Σημασιολογικά” το d και το d’ έχουν το ίδιο περιεχόμενο
- Η Ευκλείδεια απόσταση μεταξύ τους μπορεί να είναι πολύ μεγάλη
- Η γωνία όμως είναι 0 (αντιστοιχεί στη μεγαλύτερη ομοιότητα) => χρήση της γωνίας

Από γωνίες σε συνημίτονα

- Οι παρακάτω έννοιες είναι ισοδύναμες:
  - Διαβάθμιση των εγγράφων σε φθίνουσα διάταξη με βάση τη *γωνία* μεταξύ του εγγράφου και του ερωτήματος
  - Διαβάθμιση των εγγράφων σε αύξουσα διάταξη με βάση το *συνημίτονο της γωνίας* μεταξύ του εγγράφου και του ερωτήματος
- Συνημίτονο μονότονα φθίνουσα συνάρτηση στο διάστημα $[0^o, 180^o]$

Από γωνίες σε συνημίτονα



cosine(query,document)

Dot product

Unit vectors

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

q_i είναι το tf-idf βάρος του όρου i στην ερώτηση
d_i είναι το tf-idf βάρος του όρου i στο έγγραφο

$\cos(\vec{q},\vec{d})$ is the cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Κανονικοποίηση του μήκους

- Ένα διάνυσμα μπορεί να κανονικοποιηθεί διαιρώντας τα στοιχεία του με το μήκος του, με χρήση της L_2 νόρμας:

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

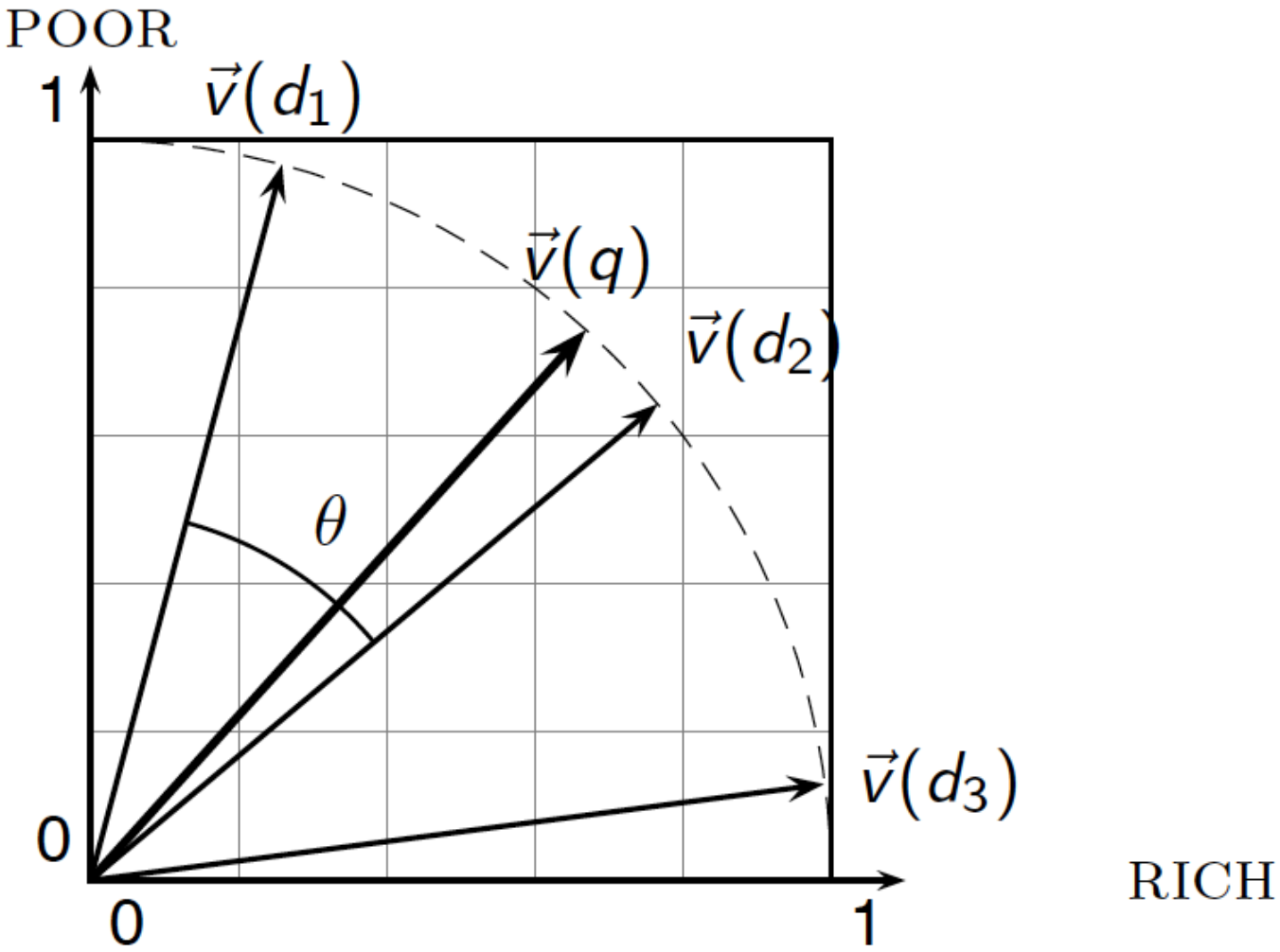
- Διαιρώντας ένα διάνυσμα με την L_2 νόρμα το κάνει μοναδιαίο
- Για παράδειγμα το d and d’ (d και μετά d) έχουν τα ίδια διανύσματα μετά την κανονικοποίηση μήκους
  - *Ως αποτέλεσμα, μικρά και μεγάλα έγγραφα έχουν συγκρίσιμα βάρη*

Συνημίτονο για κανονικοποιημένα διανύσματα

- Για διανύσματα που έχουμε κανονικοποιήσει το μήκος τους (length-normalized vectors) το συνημίτονο είναι απλώς το εσωτερικό γινόμενο (dot or scalar product):

$$\cos(\vec{q},\vec{d}) = \vec{q} \bullet \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$

Ομοιότητα συνημίτονου



Παράδειγμα

Ποια είναι οι ομοιότητα μεταξύ των έργων (εγγράφων)

SaS: *Sense and Sensibility*

PaP: *Pride and Prejudice*, and

WH: *Wuthering Heights*?

Συχνότητα όρων (μετρητές)

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Για απλοποίηση δε θα χρησιμοποιήσουμε τα idf βάρη

Παράδειγμα (συνέχεια)

Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

After length normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

cos(SaS,PaP) ≈
0.789 × 0.832 + 0.515 × 0.555 + 0.335 × 0.0 + 0.0 × 0.0
≈ 0.94
cos(SaS,WH) ≈ 0.79
cos(PaP,WH) ≈ 0.69

Why do we have cos(SaS,PaP) > cos(SaS,WH)?

Computing cosine scores

```
CosineScore(q)
 1  float Scores[N] = 0
 2  float Length[N]
 3  for each query term t
 4  do calculate w_{t,q} and fetch postings list for t
 5      for each pair(d, tf_{t,d}) in postings list
 6      do Scores[d] += w_{t,d} × w_{t,q}
 7  Read the array Length
 8  for each d
 9  do Scores[d] = Scores[d]/Length[d]
10  return Top K components of Scores[]
```

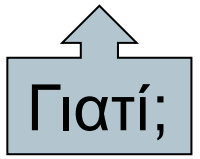
Παραλλαγές της tf-idf στάθμισης

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

Γιατί δεν έχει σημασία η βάση του λογαρίθμου;

Στάθμιση ερωτημάτων και εγγράφων

- Πολλές μηχανές αναζήτησης σταθμίζουνε διαφορετικά τις ερωτήσεις από τα έγγραφα
- Συμβολισμό: *ddd.qqq,* με χρήση των ακρονύμων του πίνακα
- Συχνό σχήμα : lnc.ltc
- Έγγραφο: logarithmic tf (l as first character), no idf, cosine normalization



- Ερώτημα: logarithmic tf (l in leftmost column), idf (t στη δεύτερη στήλη), no normalization

Περίληψη βαθμολόγησης στο διανυσματικό χώρο

- Αναπαράσταση του ερωτήματος ως ένα διαβαθμισμένο tf-idf διάνυσμα
- Αναπαράσταση κάθε εγγράφου ως ένα διαβαθμισμένο tf-idf διάνυσμα
- Υπολόγισε το συνημίτονο για κάθε ζεύγος ερωτήματος, εγγράφου
- Διάταξε τα έγγραφα με βάση αυτό το βαθμό
- Επέστρεψε τα κορυφαία K (π.χ., K =10) έγγραφα στο χρήστη

Παράδειγμα: lnc.ltc

Έγγραφο: *car insurance auto insurance*

Ερώτημα: *best car insurance*

Term	Query						Document				Prod
	tf-raw	tf-wt	df	idf	wt	n'lize	tf-raw	tf-wt	wt	n'lize	
auto	0	0	5000	2.3	0	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0.34	0	0	0	0	0
car	1	1	10000	2.0	2.0	0.52	1	1	1	0.52	0.27
insurance	1	1	1000	3.0	3.0	0.78	2	1.3	1.3	0.68	0.53

Doc length $= \sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$

Score = 0+0+0.27+0.53 = 0.8

ΤΕΛΟΣ δεύτερου μέρους 5ου Μαθήματος

Ερωτήσεις?

Χρησιμοποιήθηκε κάποιο υλικό των:
- ✓ *Pandu Nayak and Prabhakar Raghavan, CS276:Information Retrieval and Web Search (Stanford)*
- ✓ *Hinrich Schütze and Christina Lioma, Stuttgart IIR class*