ΜΥΕ003: Ανάκτηση Πληροφορίας

Διδάσκουσα: Ευαγγελία Πιτουρά

Κεφάλαια 6, 7: Βαθμολόγηση. Στάθμιση όρων. Το μοντέλο διανυσματικού χώρου.

Τι θα δούμε σήμερα;

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

Boolean Μοντέλο

- 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

Boolean Μοντέλο

- Κατάλληλο *για ειδικούς* με σαφή κατανόηση των αναγκών τους και γνώση της συλλογής
  - Επίσης, καλό *για εφαρμογές*: οι εφαρμογές μπορούν να επεξεργαστούν χιλιάδες αποτελεσμάτων.
- Αλλά, όχι κατάλληλο 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

- 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* ή με βάση *τη συνάφεια τους με το ερώτημα*

- Πως διατάσουμε-διαβαθμίζουμε τα έγγραφα μιας συλλογής με βάση ένα ερώτημα;
  - Αναθέτουμε ένα βαθμό (score) – ας πούμε στο [0, 1] – σε κάθε έγγραφο
  - score(*d*, *q*): 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 κανένα 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*

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

- Θεωρούμε το ερώτημα και το έγγραφο ως *σύνολα όρων*

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*

Εναλλακτικός τρόπος κανονικοποιήσης του μήκους:

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

Παράδειγμα

Ποιο είναι ο βαθμός για τα παρακάτω ζεύγη χρησιμοποιώντας jaccard;

q1: [information on cars]

d1: “all you’ve ever wanted to know about cars”

q1: [information on cars]

d2: “information on trucks, information on planes, information on trains”

q2: [red cars and red trucks]

d3: “cops stop red cars more often”

Προβλήματα

- Η ομοιότητα 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.

Βαθμός εγγράφου και ερώτησης

Μέτρο βαθμολογίας επικάλυψης (overlap score measure)

$$\text{score}(q, d) = \sum_{t \in q \cap d} \text{w}(\text{t}, \text{d})$$

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]

d1: “all you’ve ever wanted to know about cars”

d2: “information on trucks, information on planes, information on trains”

q: [red cars and red trucks]

d3: “cops stop red cars more often”

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: το 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
  - …

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

Στάθμιση tf-idf

Ερώτημα (q) a b

Έγγραφα

d_1 a ….. b …

d_2 a … a …

d_3 a … a … b

d_4 b ….. b … b

d_5 a … a … b … b

d_6 a ….

Διάταξη??

Η επίδραση του idf στη διάταξη

- Το idf δεν επηρεάζει τη διάταξη για ερωτήματα *με ένα μόνο όρο*, όπως iPhone

- Το idf επηρεάζει μόνο τη διάταξη για ερωτήματα *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*

Στάθμιση tf-idf

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

Υπάρχουν πολλές άλλες παραλλαγές

- Πως υπολογίζεται το “tf” (με ή χωρίς log)
- Αν δίνεται βάρος και στους όρους του ερωτήματος
- ..

Συχνότητα όρου - Term frequency (tf)

Υπενθύμιση: 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 συνάφεια (relevance) δεν αυξάνει αναλογικά με τη συχνότητα εμφάνισης 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, κλπ*

- 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

Στάθμιση με log tf

Ερώτημα (q) a b

Έγγραφα

d_1 a ….. b …

d_2 a … a …

d_3 a … a … b

d_4 b ….. b … b

d_5 a … a … b … b

d_6 a ….

Διάταξη??

Βάρος idf

Χρησιμοποιούμε $\log(N/\mathrm{df}_t)$ αντί για N/df_t για να «ομαλοποιήσουμε» την επίδραση του idf.

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

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.

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

- 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

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*

Τι θα δούμε σήμερα;

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

Στάθμιση tf-idf

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

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

- 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

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 $R^{|v|}$

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

Αποθήκευση

- Που υπάρχει αυτή η πληροφορία στο σύστημα ανάκτησης πληροφορίας;

Ομοιότητα διανυσμάτων

Πρώτη προσέγγιση: απόσταση μεταξύ δυο διανυσμάτων

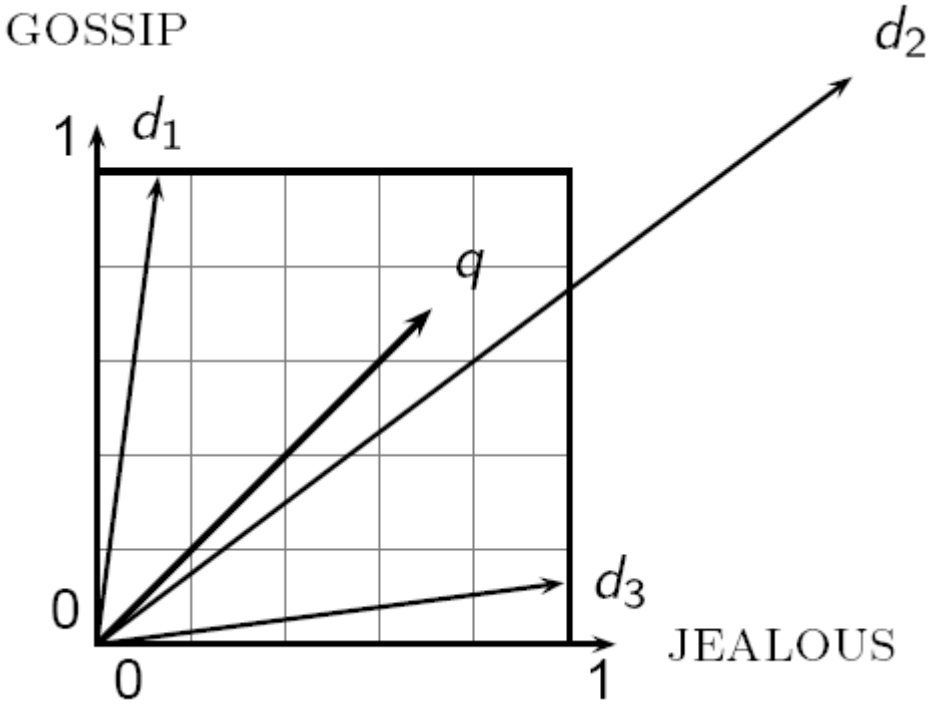
- Ευκλείδεια απόσταση;
  - Δεν είναι καλή ιδέα – είναι μεγάλη για διανύσματα διαφορετικού μήκους

Χρήση της γωνίας αντί της απόστασης

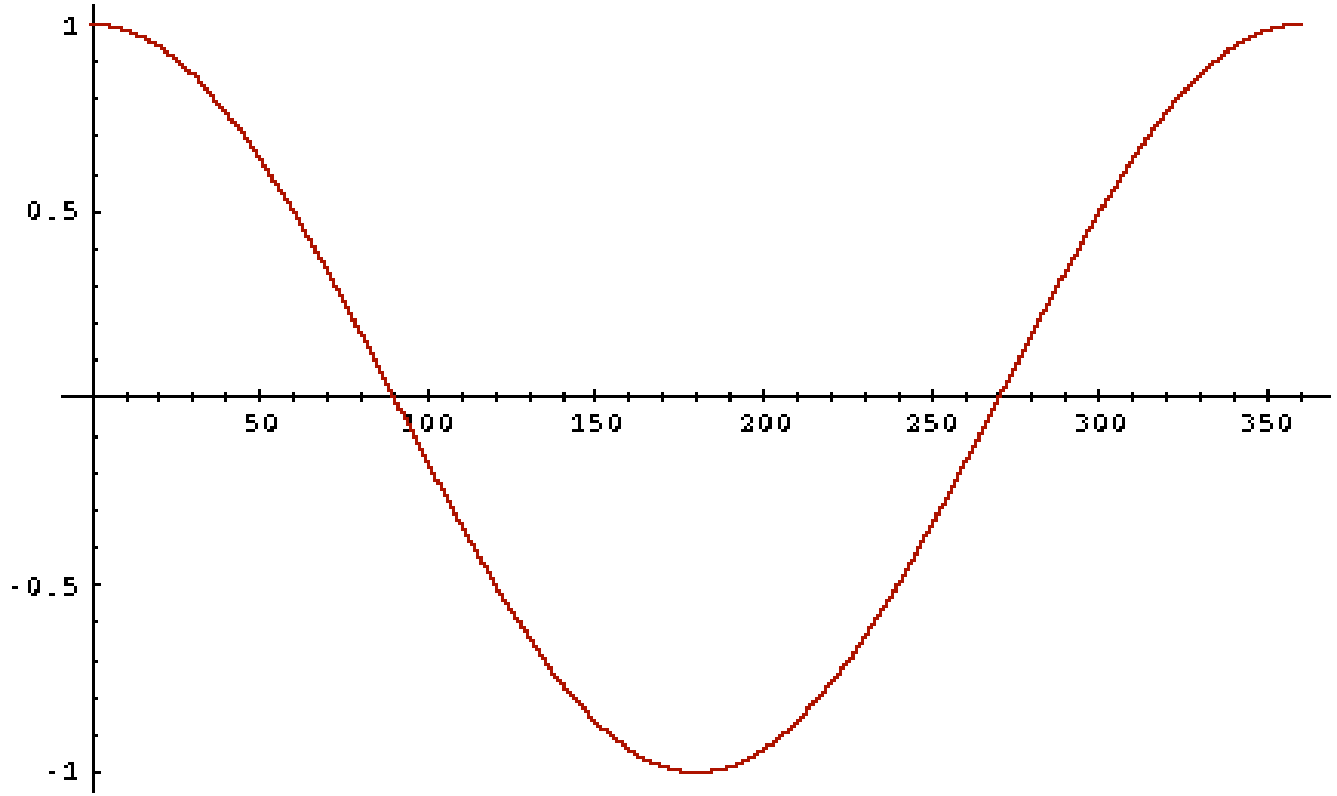
- Έστω ένα έγγραφο *d*. Υποθέστε ότι κάνουμε append το *d* στον εαυτό του και έστω *d'* το κείμενο που προκύπτει.
- "Σημασιολογικά" το *d* και το *d'* έχουν το ίδιο περιεχόμενο
- Η Ευκλείδεια απόσταση μεταξύ τους μπορεί να είναι πολύ μεγάλη
- Η γωνία όμως είναι 0 (αντιστοιχεί στη μεγαλύτερη ομοιότητα)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

Χρήση της γωνίας αντί της απόστασης

Η Ευκλείδεια απόσταση μεταξύ του $\vec{q}$ και του $\vec{d}_2$ είναι μεγάλη αν και η κατανομή των όρων είναι παρόμοια



Από γωνίες σε συνημίτονα



Συνημίτονο μονότονα φθίνουσα συνάρτηση στο διάστημα [0^o, 180^o]

Ομοιότητα εγγράφων

Dot product

Unit vectors

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

d_i (d'$_i$) είναι το tf-idf βάρος του *i*-οστού όρου στο έγγραφο *d* (*d'*)

cos($\vec{d}'$, $\vec{d}$) η ομοιότητα συνημιτόνου μεταξύ $\vec{d}'$ and $\vec{d}$ … ή, Ισοδύναμα, το συνημίτονο της γωνίας μεταξύ των $\vec{d}'$ και $\vec{d}$.

Κανονικοποίηση του μήκους

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

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

- Διαιρώντας ένα διάνυσμα με την L_2 νόρμα το κάνει μοναδιαίο
  - *Ως αποτέλεσμα, μικρά και μεγάλα έγγραφα έχουν συγκρίσιμα βάρη*

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

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

Παράδειγμα

Ποια είναι οι ομοιότητα μεταξύ των έργων

SaS: *Sense and Sensibility*

PaP: *Pride and Prejudice*, and

WH: *Wuthering Heights*?

Συχνότητα όρων (μετρητές)

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

Παράδειγμα (συνέχεια)

Για απλοποίηση δε θα χρησιμοποιήσουμε τα idf βάρη

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

Log frequency βάρος (log tf)

όρος	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

Μήκος

$$\text{SAS} = \sqrt{3.06^2 + 2.00^2 + 1.3^2 + 0^2} \approx 3.88$$

Μετά την κανονικοποίηση

όρος	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

Παράδειγμα (συνέχεια)

όρος	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

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

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

Γιατί cos(SaS,PaP) > cos(SaS,WH)?

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

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

cosine(query, document)

Dot product

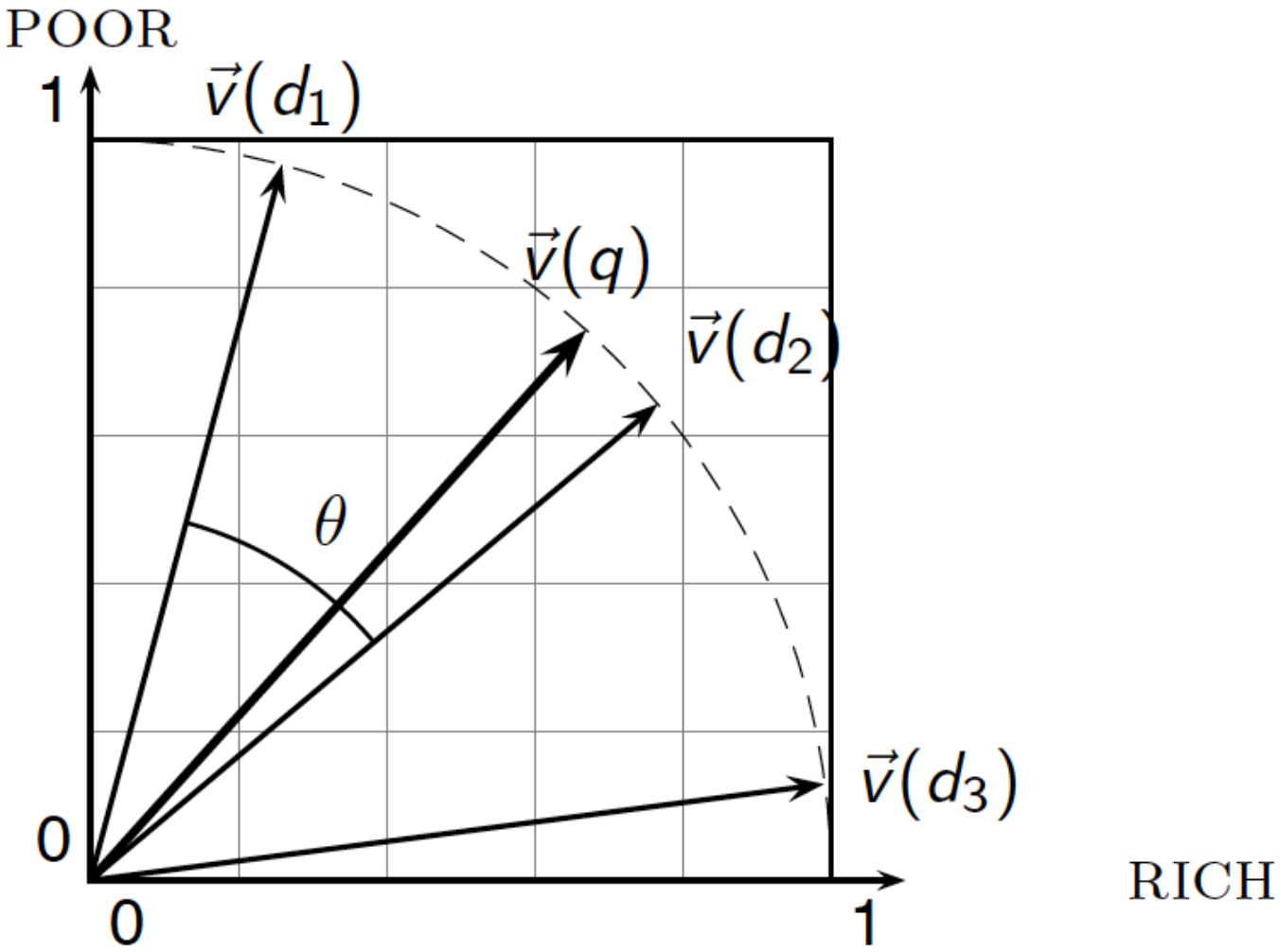
Unit vectors

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

$\vec{q}_i$ είναι το tf-idf βάρος του όρου *i* στην ερώτηση

$\vec{d}_i$ είναι το tf-idf βάρος του όρου *i* στο έγγραφο

Ομοιότητα συνημίτονου



Περίληψη βαθμολόγησης στο διανυσματικό χώρο

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

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;

Κανονικοποίηση με μέγιστη συχνότητα όρου

Έστω *τ* ο ποιο συχνός όρος σε *ένα* έγγραφο *d* και *tfmax(d)* η συχνότητα του

Διαιρούμε τη συχνότητα $tf_{t,d}$ κάθε όρου *t* στο *d* με αυτήν την τιμή

Γιατί;
Στα μεγάλα έγγραφα μεγάλες συχνότητες όρων απλώς γιατί υπάρχει επανάληψη

Προβλήματα:

- Ασταθής (πχ τροποποίηση stopwords)
- Ιδιαίτερη λέξη (outlier) που εμφανίζεται συχνά
- Πρέπει να υπάρχει διαφορά ανάμεσα σε έγγραφα με ομοιόμορφη και skewed κατανομή

Κανονικοποίηση με μέγιστη συχνότητα όρου

$$\mathrm{ntf}_{t,d} = a + (1-a)\frac{\mathrm{tf}_{t,d}}{\mathrm{tf}_{\max}(d)}$$

(augmented) Το a είναι ένας τελεστές στάθμισης (εξομάλυνσης) – smoothing factor (συχνά και 0.4 αντί 0.5)

Στάθμιση ερωτημάτων και εγγράφων

- Πολλές μηχανές αναζήτησης σταθμίζουν διαφορετικά τις ερωτήσεις από τα έγγραφα
- Συμβολισμό: *ddd.qqq*, με χρήση των ακρωνύμων του πίνακα (πρώτα 3 γράμματα έγγραφο- επόμενα 3 γράμματα ερώτημα) συχνότητα όρου-συχνότητα εγγράφων-κανονικοποίηση
- Συχνό σχήμα : lnc.ltc
  - Έγγραφο: logarithmic tf (l), no idf (n), cosine normalization (c)
  - Ερώτημα: logarithmic tf (l), idf (t), cosine normalization (c)

Παράδειγμα

Ερώτημα: *best car insurance*

Έγγραφο: *car insurance auto insurance*

N = 1000K

Inc.ltc

Όρος	Ερώτημα (Query)						Έγγραφο				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

Μήκος Ερωτήματος = $\sqrt{0^2 + 1.3^2 + 2.0^2 + 3^2} \approx 3.8$

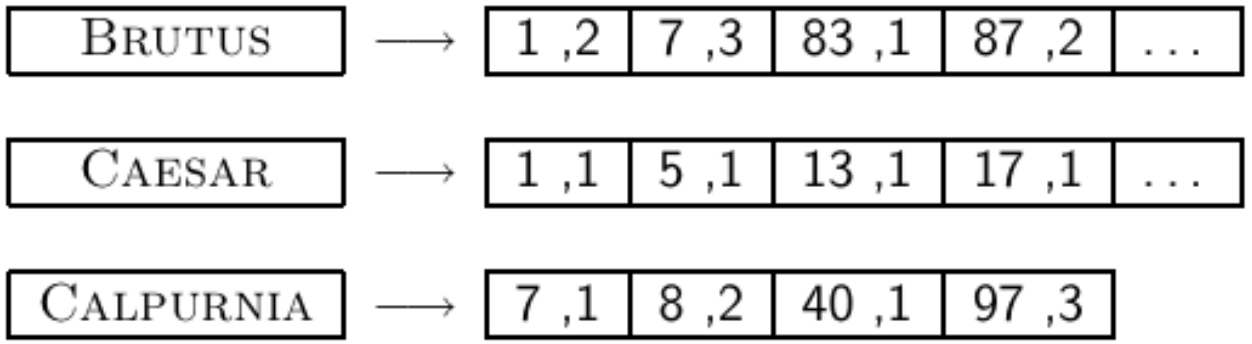
Μήκος Εγγράφου = $\sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$

Score = 0+0+(0.52*0.52=)27+(0.78*0.68=)0.53 = 0.8

Θέματα

- Που αποθηκεύουμε τις συχνότητες;
- Μια καλή δομή για τον υπολογισμό του top-*k*?
- Βοηθάει η διάταξη των εγγράφων με βάση το id; Πιο χρήσιμη διάταξη;

Επέκταση καταχωρήσεων



- Συχνότητες όρων

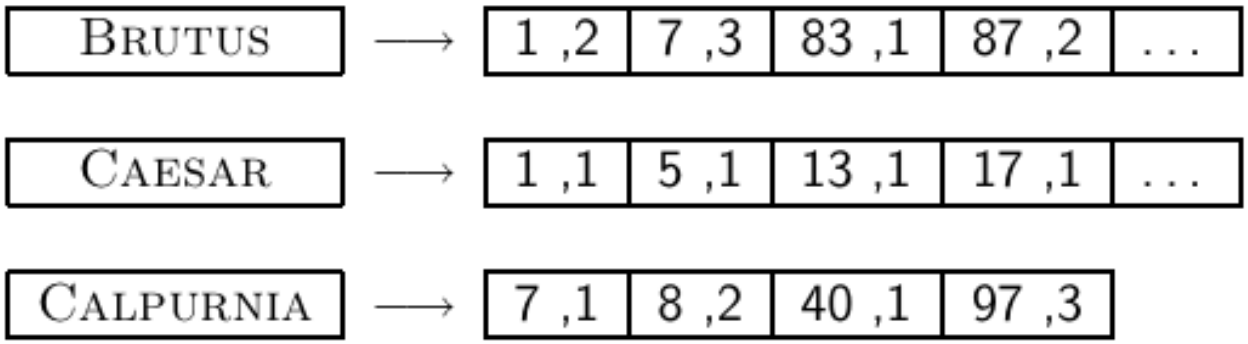
Σε κάθε καταχώρηση, αποθήκευση του $tf_{t,d}$ επιπρόσθετα του docID $_d$

- *Η συχνότητα idf_t αποθηκεύεται στο λεξικό μαζί με τον όρο t (το μήκος της αντίστοιχης λίστας καταχωρήσεων)*

Υπολογισμός ανά έγγραφο (document-at-a-time)

- (document-at-a-time) Μπορούμε να διατρέχουμε τις λίστες των όρων του ερωτήματος *παράλληλα όπως στην περίπτωση της Boolean ανάκτησης* (merge sort)
  - Αυτό έχει ως αποτέλεσμα λόγω της ίδιας διάταξης των εγγράφων στις λίστες καταχωρίσεων τον υπολογισμό του βαθμού ανά έγγραφο

Παράδειγμα



- Ερώτημα: [Brutus Caesar]
- Διατρέχουμε παράλληλα τις λίστες για το Brutus και Caesar

Υπολογισμός βαρών

Αν συνημίτονο:

- Εξαρτάται από τη μέθοδο – ίσως χρειαστεί να *αποθηκεύσουμε και το μήκος του εγγράφου* (για κανονικοποίηση) ή να αποθηκεύσουμε τις κανονικοποιημένες τιμές (αντί του tf)
- Τροποποιήσεις εγγράφων: τι αλλάζει;

- Η σχετική διάταξη των εγγράφων δεν επηρεάζεται από την κανονικοποίηση ή όχι του διανύσματος του *q*
- Αν κάθε όρος μόνο μια φορά στο ερώτημα, το $w_{t,q}$ μπορεί να αγνοηθεί, οπότε μπορούμε απλώς να αθροίζουμε τα $w_{t,d}$

Υπολογισμός *k*-κορυφαίων αποτελεσμάτων

Σε πολλές εφαρμογές, δε χρειαζόμαστε την πλήρη διάταξη, αλλά μόνο τα κορυφαία *k (top-k),* για κάποιο μικρό *k*, π.χ., *k* = 100

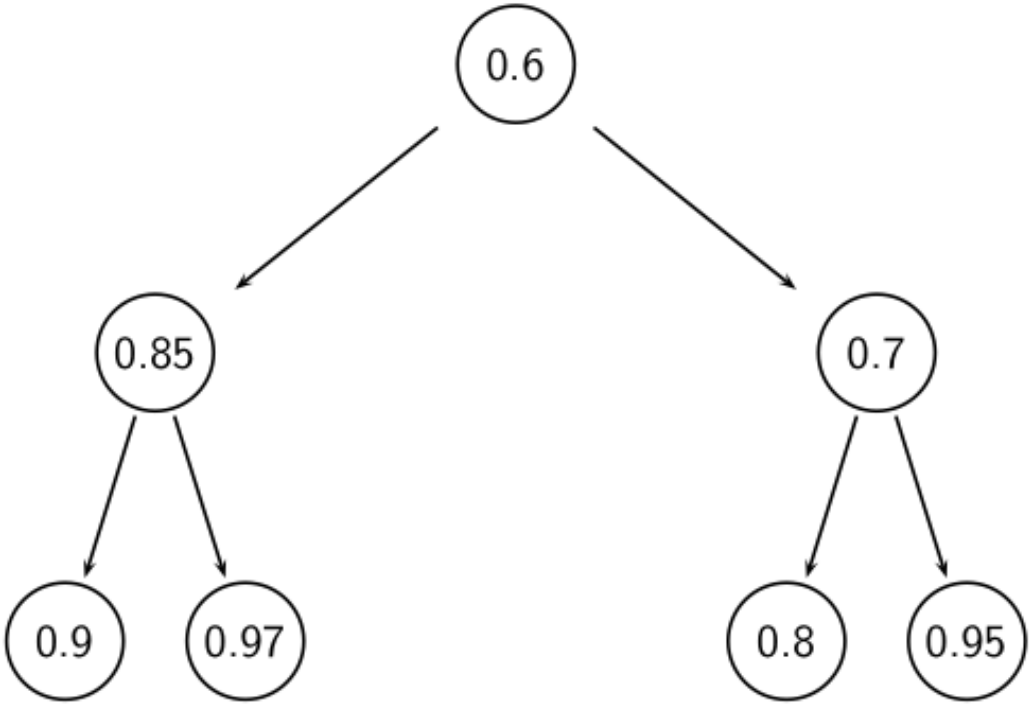
- Απλοϊκός τρόπος:
  - Υπολόγισε τους βαθμούς για όλα τα N έγραφα
  - Sort
  - Επέστεψε τα κορυφαία *k*

Αν δε χρειαζόμαστε όλη τη διάταξη, υπάρχει <u>πιο αποδοτικός</u> τρόπος να υπολογίσουμε *<u>μόνο</u>* τα κορυφαία k;

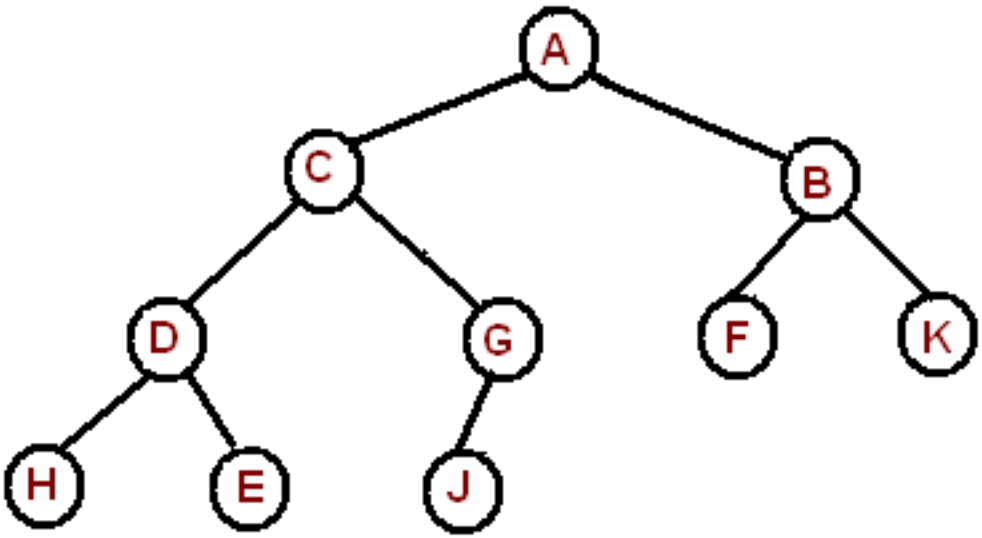
- *Έστω J τα έγγραφα με μη μηδενικό συνημίτονο. Μπορούμε να βρούμε τα K καλύτερα <u>χωρίς διάταξη όλων</u> των J εγγράφων;*

Χρήση min-heap

- Χρήση δυαδικού min heap
- Ένα δυαδικό min heap είναι ένα δυαδικό δέντρο που η τιμή ενός κόμβου είναι μικρότερη από την τιμή των δύο παιδιών του.



Αποθήκευση σε πίνακα



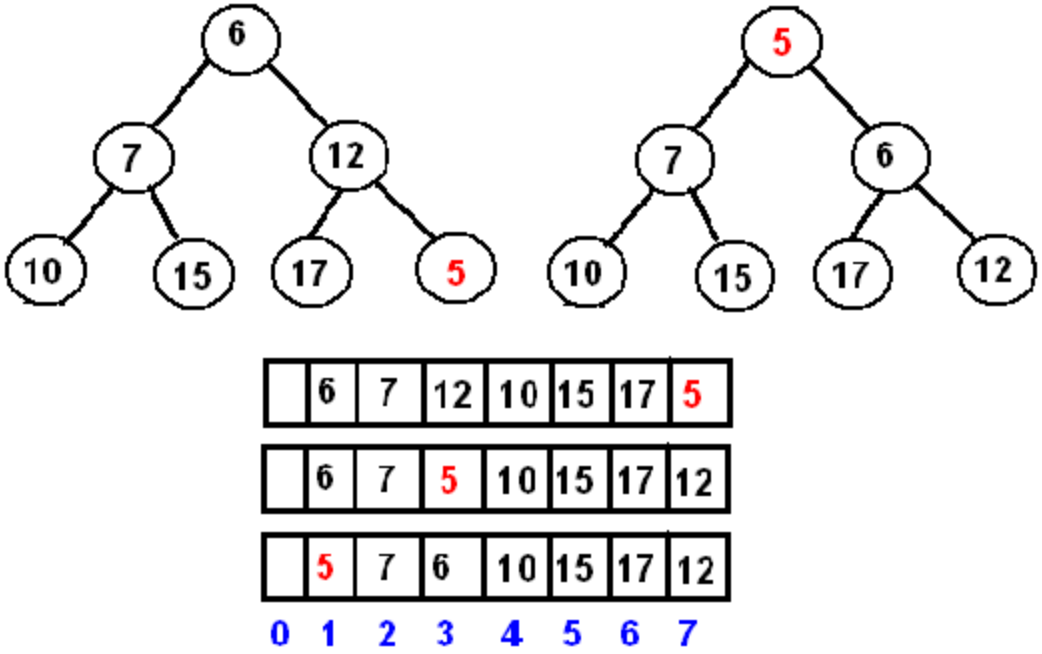
0	1	2	3	4	5	6	7	8	9	10
	A	C	B	D	G	F	K	H	E	J

Η ρίζα είναι στη θέση 1 του πίνακα. Για το i-οστό στοιχείο:

- Το αριστερό παιδί είναι στη θέση 2*i
- Το δεξί παιδί είναι στη θέση 2*i+1
- Ο γονέας στη θέση i/2

Εισαγωγή στοιχείου

Το νέο στοιχείο εισάγεται *ως το τελευταίο στοιχείο* (στο τέλος του heap)
Η ιδιότητα του heap εξασφαλίζεται με σύγκριση του στοιχείου με τον γονιό του και μετακίνηση του προς τα πάνω (swap with parent) μέχρι να συναντήσει στοιχείο ίσο ή μεγαλύτερο (*percolation up*).



Διαγραφή μικρότερου στοιχείου

Το μικρότερο στοιχείο βρίσκεται *στη ρίζα* (το πρώτο στοιχείο του πίνακα)

Το σβήνουμε από τη λίστα και το αντικαθιστούμε με το τελευταίο στοιχείο στη λίστα, εξασφαλίζοντας την ιδιότητα του heap συγκρίνοντας με τα παιδιά του (*percolating down*)

Επιλογή των κορυφαίων k σε $O(N \log k)$

Στόχος: Διατηρούμε τα καλύτερα k που έχουμε δει μέχρι στιγμής

- Χρήση δυαδικού min heap
- Για την επεξεργασία ενός νέου εγγράφου d' με score s':
  - Get *current minimum* h_m of heap *(O(1))*
  - If $s' < h_m$ skip to next document /* υπάρχουν k καλύτερα */
  - If $s' > h_m$ heap-delete-root *(O(log k))* /* καλύτερο, σβήσε τη ρίζα
    heap-add d'/s' *(O(log k))* και βάλτο στο heap */

Πιο αποδοτικός υπολογισμός;

Η ταξινόμηση (merge) έχει πολυπλοκότητα χρόνου $O(N)$ όπου N ο αριθμός των εγγράφων (ή, ισοδύναμα J).

Βελτιστοποίηση κατά ένα σταθερό όρο, αλλά ακόμα θέλουμε $O(N)$, $N > 10^{10}$ *(δηλαδή, πρέπει να «δούμε» όλα τα έγγραφα)*

Υπάρχουν sublinear αλγόριθμοι;

- Αυτό που ψάχνουμε στην πραγματικότητα αντιστοιχεί στο να λύνουμε το πρόβλημα των *k*-πλησιέστερων γειτόνων (*k*-nearest neighbor (kNN) problem) στο διάνυσμα του ερωτήματος (= query point).
- *Δεν υπάρχει γενική λύση σε αυτό το πρόβλημα που να είναι sublinear. (ειδικά για πολλές διαστάσεις)*

Ασφαλής (safe) και μη ασφαλής (non-safe) διάταξη

- Ο όρος ασφαλής διάταξη (safe ranking) χρησιμοποιείται για μεθόδους που εξασφαλίζουν ότι τα Κ έγγραφα που επιστέφονται είναι *ακριβώς* τα Κ έγγραφα με το μεγαλύτερο score

- Μη ασφαλής (ή inexact) διάταξη μας δίνει «καλά» Κ έγγραφα αλλά όχι απαραίτητα τα κορυφαία Κ
  - αποδεκτή αλλά πρέπει να εξασφαλίσουμε ότι δεν είμαστε «πολύ μακριά» από την ασφαλή διάταξη
  - Έτσι και αλλιώς, η tf.idf στάθμιση δεν είναι ακριβής αποτίμηση της συνάφειας, αλλά μια εκτίμηση της

Γενική προσέγγιση «ψαλιδίσματος» (pruning)

- Βρες ένα **σύνολο *A*** από υποψήφια έγγραφα (*contenders)*, όπου *K* < |*A*| << *N*
  - *Το A δεν περιέχει απαραίτητα όλα τα* top *K, αλλά περιέχει αρκετά καλά έγγραφα και πολλά από τα* top *K*
- Επέστρεψε τα top K έγγραφα του A

Το A είναι ένα ψαλίδισμα (pruning) των μη υποψηφίων

Θα δούμε σχετικούς ευριστικούς

Περιορισμός του ευρετηρίου (index elimination)

- Ο βασικός αλγόριθμος υπολογισμού του συνημίτονου θεωρεί έγγραφα που περιέχουν *τουλάχιστον έναν όρο του ερωτήματος*

- Μπορούμε να επεκτείνουμε αυτήν την ιδέα;
  - Εξετάζουμε μόνο τους όρους του ερωτήματος με *μεγάλο idf*
  - Εξετάζουμε μόνο έγγραφα που περιέχουν *πολλούς από τους όρους του ερωτήματος*

Μόνο όροι με μεγάλο idf

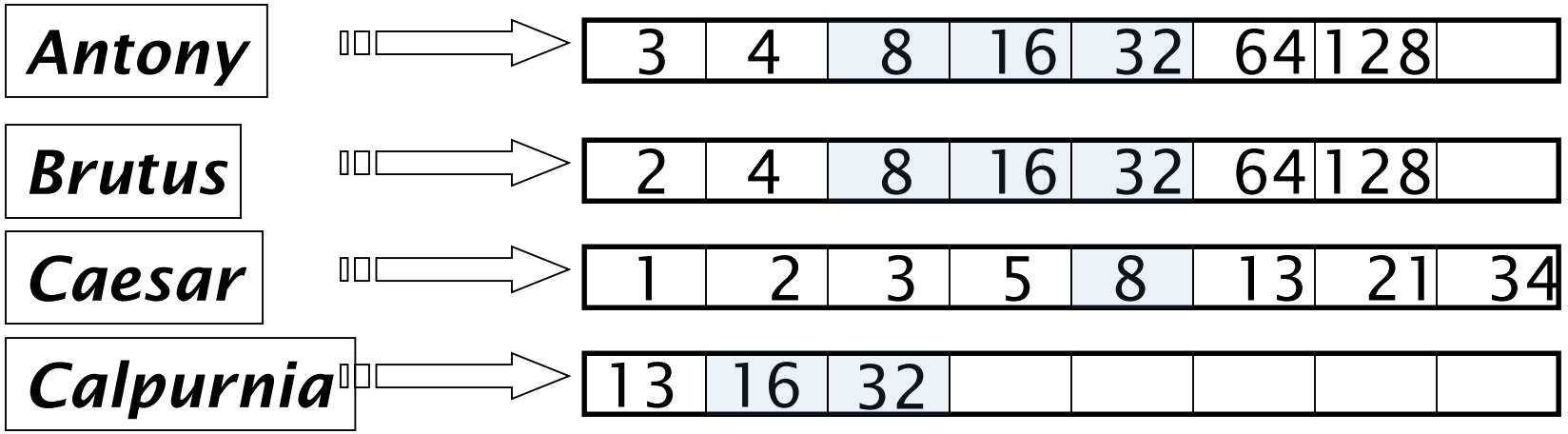
- Παράδειγμα: Για το ερώτημα: *catcher in the rye*
  - Αθροίζουμε μόνο το βαθμό για τους όρους catcher και rye
  - Γιατί; οι όροι **in** και **the** έχουν μικρή συνεισφορά στο βαθμό και άρα δεν αλλάζουν σημαντικά τη διάταξη

- Όφελος:
  - Οι καταχωρήσεις των όρων με μικρά idf περιέχουν *πολλά έγγραφα (μεγάλες λίστες καταχωρήσεων)* → αυτά τα (πολλά) έγγραφα δε μπαίνουν ως υποψήφια στο σύνολο Α

Έγγραφα με πολλούς όρους του ερωτήματος

- Κάθε έγγραφο που έχει *τουλάχιστον έναν όρο* του ερωτήματος είναι υποψήφιο για τη λίστα με τα κορυφαία Κ έγγραφα
- Για ερωτήματα με *πολλούς όρους*, υπολογίζουμε τους βαθμούς μόνο των εγγράφων που περιέχουν *αρκετούς από τους όρους του ερωτήματος*
  - Για παράδειγμα, τουλάχιστον 3 από τους 4 όρους
  - Παρόμοιο με ένα είδος *μερικής σύζευξης* (“soft conjunction”) στα ερωτήματα των μηχανών αναζήτησης (αρχικά στη Google)
- Εύκολα να υλοποιηθεί κατά τη διάσχιση των καταχωρήσεων

Παράδειγμα



3 από τους 4 όρους του ερωτήματος

Υπολογισμοί βαθμών μόνο για τα έγγραφα 8, 16 και 32

Λίστες πρωταθλητών

- **Προ-υπολογισμός** για κάθε όρο *t* του λεξικού, των ***r*** εγγράφων με το μεγαλύτερο βάρος ανάμεσα στις καταχωρήσεις του *t* -> *λίστα πρωταθλητών* (champion list , fancy list ή top docs για το *t*) (για κάθε *t*, τα καλύτερα *r* έγγραφα)
  - Αν tf.idf, είναι αυτά *με το καλύτερο tf*
- *Κατά την ώρα του ερωτήματος*, πάρε ως Α την ένωση των λιστών πρωταθλητών για τους όρους του ερωτήματος, υπολόγισε μόνο τους βαθμούς για τα έγγραφα της Α και διάλεξε τα Κ ανάμεσα τους
- Το *r* πρέπει να επιλεγεί κατά τη διάρκεια της κατασκευής του ευρετηρίου
  - Έτσι, είναι πιθανόν ότι $r < K$

Υπολογισμός ανά όρο

Υπολογισμός ανά-όρο (ένας-όρος-τη-φορά - a-term-at-a-time)

- Επεξεργαζόμαστε *όλη* τη λίστα καταχωρήσεων για τον *πρώτο όρο* του ερωτήματος
- Δημιουργούμε ένα *συσσωρευτή* των βαθμών για κάθε docID εγγράφου που βρίσκουμε
- Μετά επεξεργαζόμαστε πλήρως τη λίστα καταχωρήσεων για τον *δεύτερο όρο* κοκ

Υπολογισμός ανά όρο (term-at-a-time)

```
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[]
```

Για κάθε όρο *t* του ερωτήματος *q*

Λέμε τα στοιχεία του πίνακα Scores, συσσωρευτές (accumulators)

Διάταξη καταχωρήσεων του t με βάση το $wf_{t,d}$

Διατάσσουμε τα έγγραφα στις λίστες καταχωρήσεων με βάση το βάρος (weight) $wf_{t,d}$

- Η απλούστερη περίπτωση, normalized tf-idf weight

Τα «καλά» έγγραφα για έναν όρο είναι στην αρχή της λίστας

Όχι κοινή διάταξη των εγγράφων σε όλες τις λίστες

Αλλά, δε μπορούμε να υπολογίσουμε ένα συνολικό βαθμό για κάθε έγγραφο με merge sort

→ "συσσωρεύουμε" τους βαθμούς για τα έγγραφα ανά όρο

Διάταξη καταχωρήσεων του t με βάση το $wf_{t,d}$

Προσέγγιση: δεν επεξεργαζόμαστε τις καταχωρήσεις που θα συνεισφέρουν *λίγο* στον τελικό βαθμό

- Τα κορυφαία k έγγραφα είναι πιθανόν να βρίσκονται *στην αρχή* αυτών των ταξινομημένων λιστών.

→ γρήγορος τερματισμός ενώ επεξεργαζόμαστε τις λίστες καταχωρήσεων *μάλλον* δε θα αλλάξει τα κορυφαία k έγγραφα

Υπολογισμός ανά όρο

```
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[]
```

Μη φέρεις όλη τη λίστα καταχωρήσεων, μόνο τα πρώτα στοιχεία της

1. Πρόωρος τερματισμός

- Κατά τη διάσχιση των καταχωρήσεων ενός όρου *t,* σταμάτα νωρίς αφού:
  - Δεις ένα προκαθορισμένο αριθμό *r* από έγγραφα
  - Το $wf_{t,d}$ πέφτει κάτω από κάποιο κατώφλι
- Πάρε την ένωση του συνόλου των εγγράφων που προκύπτει
  - Ένα σύνολο για κάθε όρο
- Υπολόγισε τους βαθμούς μόνο αυτών των εγγράφων

2. idf-διατεταγμένοι όροι

Κατά την επεξεργασία των όρων του ερωτήματος

- Εξετάζουμε τους όρους με φθίνουσα διάταξη ως προς idf
  - Όροι με μεγάλο idf πιθανών να συνεισφέρουν περισσότερο στο βαθμό
- Καθώς ενημερώνουμε τη συμμετοχή στο βαθμό κάθε όρου
  - Σταματάμε αν ο βαθμός των εγγράφων δεν μεταβάλλεται πολύ

Επεξεργασία Ανά-Έγγραφο και Ανά-Όρο

- *Υπολογισμός ανά-όρο (term-at-a-time processing): Υπολογίζουμε για κάθε όρο της ερώτησης, για κάθε έγγραφο που εμφανίζεται στη λίστας καταχώρησης του ένα βαθμό και μετά συνεχίζουμε με τον επόμενο όρο της ερώτησης*
- *Υπολογισμός Ανά Έγγραφο (document-at-a-time processing): Τελειώνουμε τον υπολογισμό του βαθμού ομοιότητας ερωτήματος-εγγράφου για το έγγραφο d_i πριν αρχίσουμε τον υπολογισμό βαθμού ομοιότητας ερωτήματος-εγγράφου για το έγγραφο d_{i+1}.*

Με βάση την «ποιότητα» του εγγράφου (*g*(d))

Συχνά υπάρχει ένας *ανεξάρτητος του ερωτήματος (στατικός)* χαρακτηρισμός της καταλληλότητας ("goodness", authority) του εγγράφου – έστω *g*(d)

Για παράδειγμα:

- Στις μηχανές αναζήτησης (στο Google) το PageRank *g*(*d*) μιας σελίδας *d* μετρά το πόσο «καλή» είναι μια σελίδα με βάση το πόσες «καλές» σελίδες δείχνουν σε αυτήν, ή
- αριθμός hits (δημοφιλές έγγραφο) ή
- wikipedia σελίδες ή
- άρθρα σε μια συγκεκριμένη εφημερίδα, κλπ

Με βάση την «ποιότητα» του εγγράφου (g(d))

Αν υπάρχει μια διάταξη της καταλληλότητας τότε ο συγκεντρωτικός βαθμός (net-score) ενός εγγράφου ***d*** και μιας ερώτησης ***q*** είναι ένας συνδυασμός της ποιότητας του εγγράφου (που έστω ότι δίνεται από μια συνάρτηση *g* στο [0, 1]) και της συνάφειας του με το ερώτημα *q (πχ με χρήση tf-idf)*:

$$\text{net-score}(q, d) = g(d) + \text{score}(q, d)$$

Θέλουμε να επιλέξουμε σελίδες που είναι και γενικά σημαντικές (authoritative) και συναφείς ως προς την ερώτηση (το οποίο μας δίνει το score)

- Στην πράξη (βάρη) $\textit{net-score}(q, d) = w_1\, g(d) + w_2\, \textit{score}(q, d)$
- Υπόθεση: κανονικοποίηση ώστε score επίσης στο [0, 1]

Με βάση την «ποιότητα» του εγγράφου (g(d))

Θέλουμε διάταξη με βάση το net-score

Πως *μπορούμε να επιτύχουμε γρήγορο τερματ*ισμό (early termination); Δηλαδή να μην επεξεργαστούμε όλη τη λίστα καταχωρήσεων για να βρούμε τα καλύτερα k;

Με βάση την «ποιότητα» του εγγράφου (g(d))

- Διατάσουμε τις λίστες καταχωρήσεων με βάση την καταλληλότητα (π.χ., PageRank) των εγγράφων:

$$g(d_1) > g(d_2) > g(d_3) > \ldots$$

Η διάταξη των εγγράφων είναι ίδια για όλες τις λίστες καταχωρήσεων

✓ Τα «καλά» έγγραφα στην αρχή της κάθε λίστας, οπότε αν θέλουμε να βρούμε γρήγορα καλά αποτελέσματα μπορούμε να δούμε μόνο την αρχή της λίστας

Με βάση την «ποιότητα» του εγγράφου (*g*(d))

Υπενθύμιση net-score(*q*, *d*) = *g*(*d*) + score(*q*, *d*) και τα έγγραφα σε κάθε λίστα σε διάταξη με βάση το *g*

Επεξεργαζόμαστε ένα έγγραφο τη φορά – δηλαδή, για κάθε έγγραφο υπολογίζουμε πλήρως το net-score του (για όλους τους όρους του ερωτήματος)

- Έστω *g* → [0, 1],

  το τελευταίο *k*-κορυφαίο έγγραφο έχει βαθμό **1.2**

  και για το έγγραφο *d* που επεξεργαζόμαστε *g(d) < 0.1*, άρα και για όλα τα υπόλοιπα συνολικός βαθμός < 1.1 (στην καλύτερη περίπτωση έχουν score ίσο με 1 που δεν αρκεί όμως).

  => δε χρειάζεται να επεξεργαστούμε το υπόλοιπο των λιστών

Περίληψη

- Ορισμός score(q, d)
  - tf-idf
  - Διανυσματικό μοντέλο

- Υπολογισμός top-*k* συναφών εγγράφων
  - Χρήση *heap*
  - *Ασφαλής* και *μη ασφαλής* τερματισμός (ακριβής – μη ακριβής υπολογισμός)
  - Τεχνικές
    - Ανά έγγραφο – *Document At A Time (DAAT)*
    - Ανά όρο – *Term At A Time (TAAT)*
    - Επηρεάζει τον τρόπο διάταξης των εγγράφων στις λίστες καταχωρήσεων του ανεστραμμένου ευρετηρίου

Περίληψη

Αλγόριθμοι υπολογισμού

1. Διάταξη με *doc-id*
   - υπολογισμό *ανά έγγραφο*
   - μη ασφαλή (pruning): (1) μικρό idf, (2) τουλάχιστον m1 από τους m2 (1 < m1 < m2) όρους του ερωτήματος, (3) λίστες πρωταθλητών
2. Διάταξη με $tf_{t,d}$
   - Υπολογισμός *ανά όρο*
   - Μη ασφαλής – γρήγορος τερματισμός: (1) prune για κάθε όρο (τα πρώτα *r* έγγραφα ή όλα πάνω κάποιου tf) (2) εξέταση με βάση idf, σταμάτα αν όχι μεγάλη αλλαγή

Περίληψη

Αλγόριθμοι υπολογισμού (συνέχεια)

3. *Αν υπάρχει g(d) (κάποια βαθμολογία των εγγράφων ανεξάρτητη του ερωτήματος),* διάταξη με *g(d)*
   - υπολογισμό *ανά έγγραφο*
   - ασφαλή: κατώφλι
   - μη ασφαλή: γρήγορος τερματισμός

Κλάδεμα συστάδων

Προ-επεξεργασία: συσταδοποίηση (clustering) εγγράφων

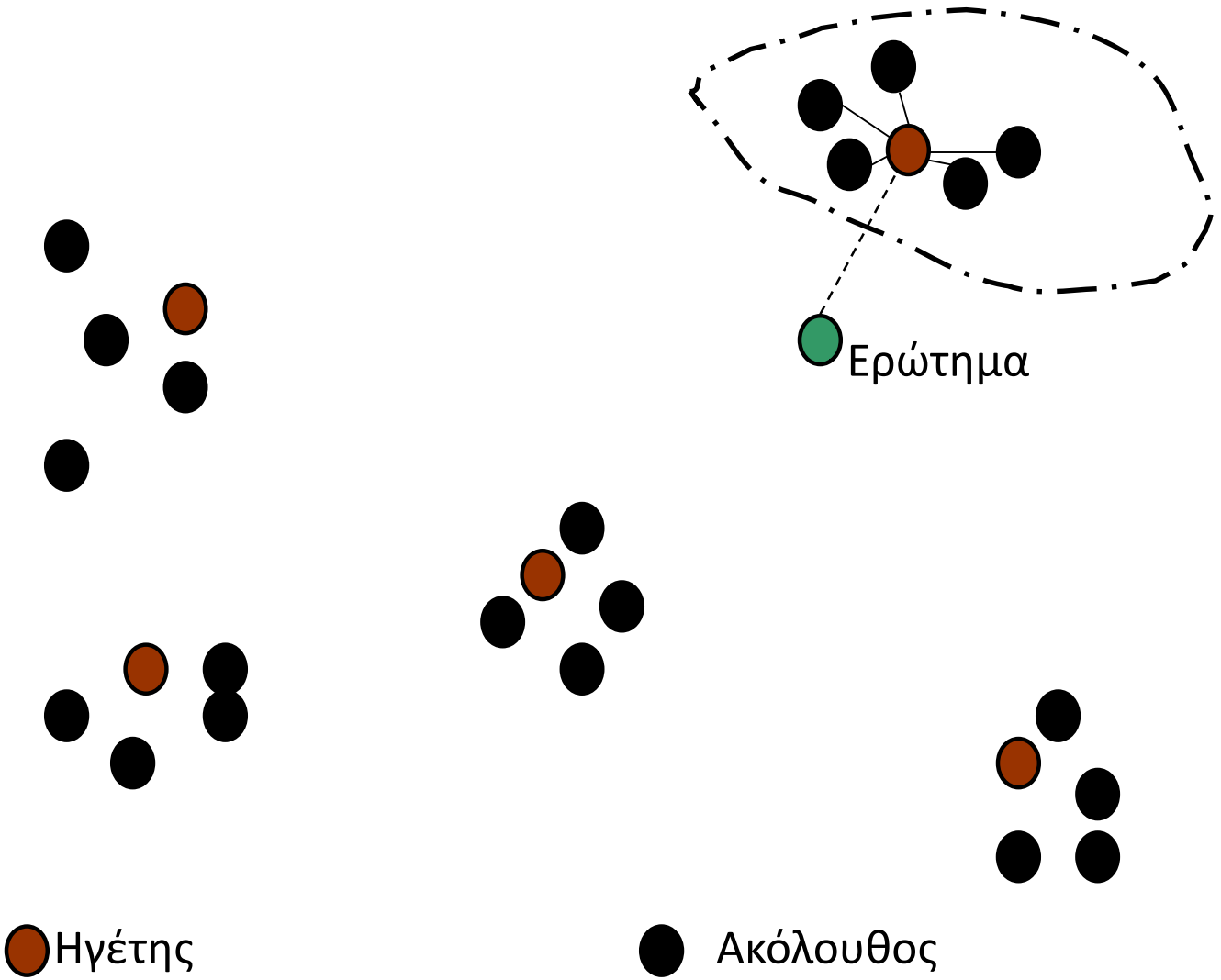
- Επέλεξε τυχαία K (πχ, √N) έγγραφα: τα οποία τα ονομάζουμε ηγέτες (*leaders)*
- Για κάθε άλλο έγγραφο, προ-υπολογίζουμε τον κοντινότερο ηγέτη του (χρήση ομοιότητας συνημιτόνου)
  - Αυτά τα έγγραφα καλούντα ακόλουθοι (followers);
  - Ο αναμενόμενος αριθμός είναι: ~ √*N* ακόλουθοι ανά ηγέτη
- Τελικά √ N ομάδες με √ N έγγραφα

Κλάδεμα συστάδων

Για κάθε ερώτημα *q*

- Βρες τον πιο κοντινό ηγέτη *L.*
- Ψάξε για τα *K* πλησιέστερα έγγραφα ανάμεσα στους ακολούθους του *L (δηλαδή, στην ομάδα του L)*.

Κλάδεμα συστάδων



Κλάδεμα συστάδων

Γιατί τυχαία δείγματα;

- Γρήγορη
- Οι ηγέτες αντανακλούν την πραγματική κατανομή

Κλάδεμα συστάδων

Γενικές παραλλαγές (b1-b2)

- Κάθε ακόλουθος συνδέεται με *b1*=3 πλησιέστερους ηγέτες.
- Για ένα ερώτημα, βρες *b2* = 4 κοντινότερους ηγέτες και τους ακολούθους τους.

Τι (άλλο) θα δούμε σήμερα;

- Παραμετρικά Ευρετήρια
- Βαθμιδωτά Ευρετήρια
- Συνολικό Σύστημα

Παραμετρικά ευρετήρια και ευρετήρια ζώνης

- Μέχρι τώρα, ένα έγγραφο ως μια ακολουθία όρων
- Στην πραγματικότητα, τα *ευρετήρια είναι χωρισμένα σε τμήματα* με διαφορετική σημασία:
  - Συγγραφέας
  - Τίτλος
  - Ημερομηνία δημοσίευσης
  - Γλώσσα
  - κλπ
- Καλούνται και μεταδεδομένα (metadata) του εγγράφου

Παραμετρική αναζήτηση

Παραμετρικά ερωτήματα

- Συχνά αναζήτηση με βάση τα μεταδεδομένα
  - Π.χ., βρες όλα τα έγγραφα που έγγραψε ο William Shakespeare το 1601, που περιέχουν τις λέξεις *alas poor Yorick*
  - Year = 1601 είναι παράδειγμα ενός πεδίου (field)
  - Επίσης, author last name = shakespeare, κλπ

- Ερωτήματα με πεδία (παραμετρικά ερωτήματα) συνήθως ερμηνεύονται ως συζευκτικά (conjunction AND) πρέπει να ισχύουν όλα)

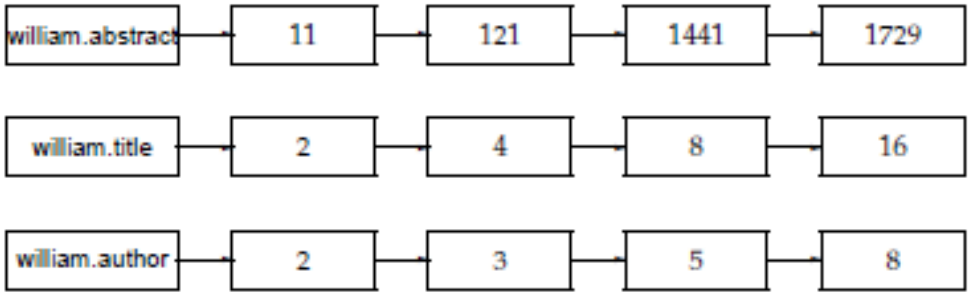
Ζώνη

- Η ζώνη (zone) είναι μια περιοχή ενός εγγράφου που περιέχει κείμενο, π.χ.,
  - Title (τίτλος)
  - Abstract (περίληψη)
  - References (αναφορές) …
- Πρέπει να τροποποιήσουμε τα ευρετήρια ώστε να επιτρέψουμε σχετικά ερωτήματα όπως
  - πχ, βρες έγγραφα με τον όρο «merchant» στον τίτλο τους για το ερώτημα «gentle rain»
- Επίσης χρήσιμα αν θέλουμε να δώσουμε *μεγαλύτερο βάρος* σε εμφανίσεις όρων στον τίτλο ή στην περίληψη

Πεδία (fields)

- Ευρετήριο πεδίου (Field index) ή παραμετρικό ευρετήριο (parametric index): καταχωρήσεις (postings) για κάθε πεδίο
  - Συχνά ειδικού τύπου (πχ δέντρα διαστήματος για ημερομηνίες)

Βασικό ευρετήριο ζώνης encoded στο λεξικό (διαφορετικές λίστες καταχωρήσεων)
Ένα ευρετήριο για κάθε ζώνη/πεδίο):



Ερωτήματα με ζώνες

Η πληροφορία ζώνης στις λίστες καταχώρησης:

- Το πρώτο καλύτερο για παραμετρικά ερωτήματα
- Το δεύτερο καλύτερο για υπολογισμό «ενιαίας» συνάφειας

Βαθμιδωτά (διαστρωματωμένα) ευρετήρια (Tiered indexes)

Βασική ιδέα:

- Κατασκευάζουμε διάφορα επίπεδα/βαθμίδες από ευρετήρια, όπου το καθένα αντιστοιχεί στη σημαντικότητα των όρων
- Κατά τη διάρκεια της επεξεργασίας του ερωτήματος,
  - Αρχίζουμε από την υψηλότερη βαθμίδα
  - Αν το ευρετήριο της υψηλότερης βαθμίδας, έχει τουλάχιστον k (π.χ., $k = 100$) αποτελέσματα: σταμάτα και επέστρεψε αυτά τα αποτελέσματα στο χρήστη
  - Αλλιώς, αν έχουμε βρει $< k$ ταιριάσματα: επανέλαβε την αναζήτηση στην επόμενη βαθμίδα

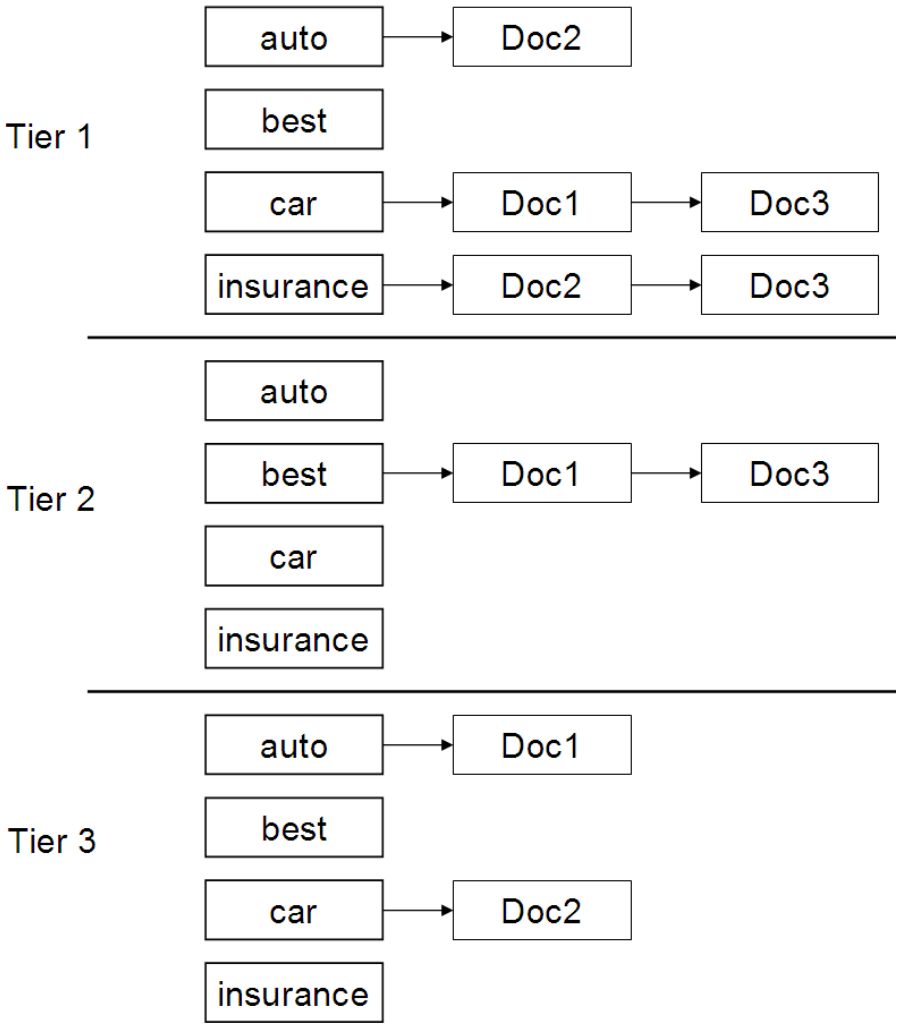
Βαθμιδωτά ευρετήρια

Παράδειγμα

Έστω 2 βαθμίδες

- Βαθμίδα 1: Ευρετήριο για όλους τους τίτλους ή με τα έγγραφα με μεγάλο tf.idf ή με τα έγγραφα με μεγάλο g(d)
- Βαθμίδα 2: Ευρετήριο για τα υπόλοιπο έγγραφα ή με τα έγγραφα με μικρό tf.idf ή με τα έγγραφα με μικρό g(d)

Βαθμιδωτά ευρετήρια



Βαθμιδωτά ευρετήρια

- Η χρήση βαθμιδωτών ευρετηρίων θεωρείται ως ένας από τους λόγους που η ποιότητα των αποτελεσμάτων του Google ήταν αρχικά σημαντικά καλύτερη (2000/01) από αυτήν των ανταγωνιστών τους.
- μαζί με το PageRank, τη χρήση του anchor text και περιορισμών θέσεων (proximity constraints)

Συνδυασμός διανυσματικής ανάκτησης

- Πως συνδυάζουμε την ανάκτηση φράσεων (και γενικά την εγγύτητα όρων – proximity queries) με τη διανυσματική ανάκτηση;
  - *Window*: το μικρότερο παράθυρο που περιέχονται όλοι οι όροι του ερωτήματος μετρημένο ως το πλήθος λέξεων του παραθύρου
  - Χρήση στη διάταξη του μεγέθους του παραθύρου – πως?
    - Με κάποιο σταθμισμένο άθροισμα?
- Πως συνδυάζουμε την Boolean ανάκτηση με τη διανυσματική ανάκτηση;
  - Π.χ., AND ή NOT
- Πως συνδυάζουμε τα * με τη διανυσματική ανάκτηση;
- Evidence accumulation

Πολλαπλοί παράγοντες

- Συνδυασμοί score πχ με βάρη

- Πως
  - Σε ορισμένες εφαρμογές από χρήστες
  - Machine learning (αλγόριθμοι μάθησης)

Επεξεργασία ερωτήματος

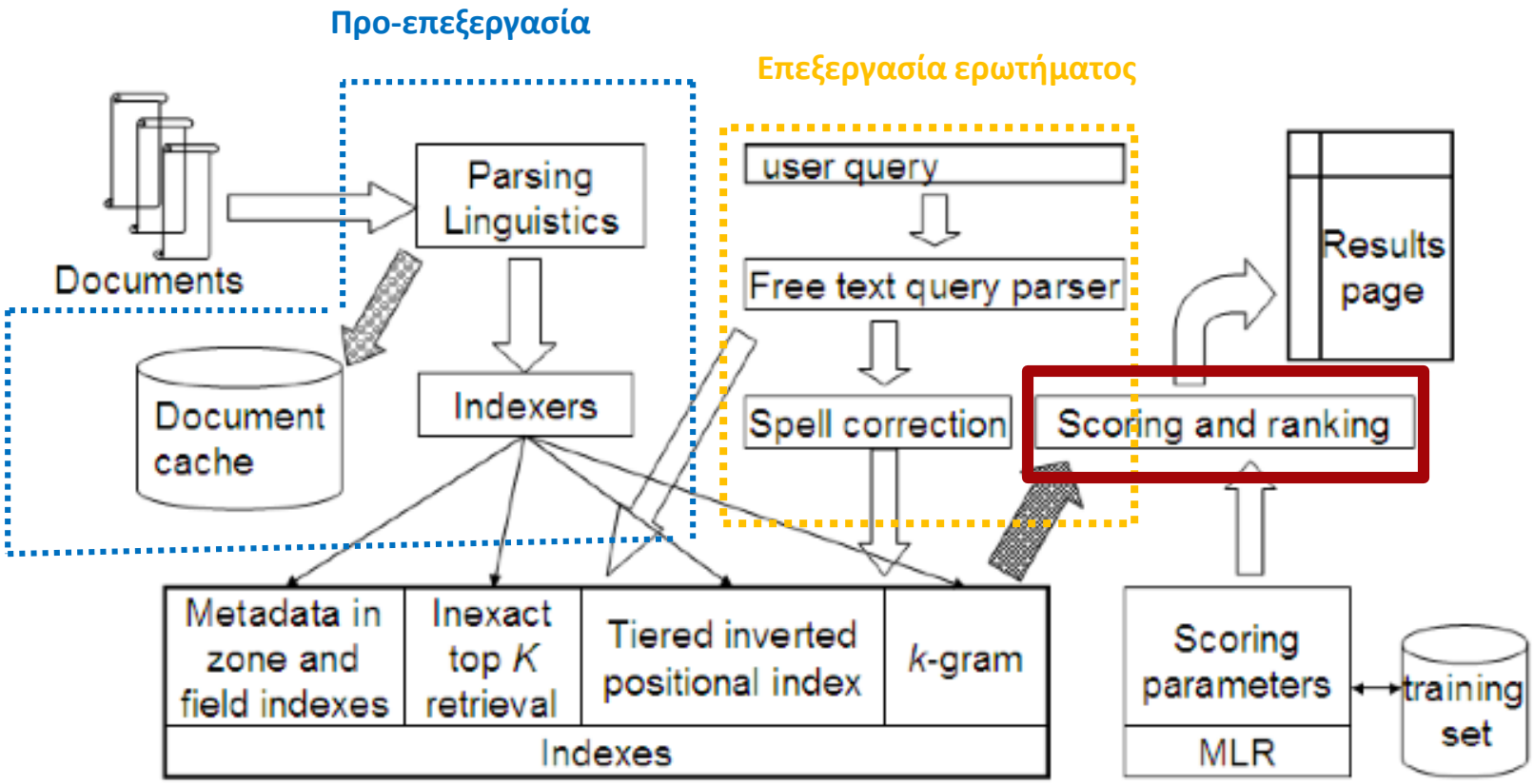
Αναλυτής ερωτημάτων (query parser)

Παράδειγμα rising interest rates

1. Εκτέλεσε την ερώτημα ως ερώτημα φράσης “rising interest rates” και κατάταξε τα αποτελέσματα χρησιμοποιώντας διανυσματική βαθμολόγηση
2. Αν δεν υπάρχουν αρκετά αποτελέσματα, εκτέλεσε το ερώτημα ως 2 ερωτήματα φράσεις: “rising interest” και “interest rates” και κατάταξε τα αποτελέσματα χρησιμοποιώντας διανυσματική βαθμολόγηση
3. Αν δεν υπάρχουν αρκετά αποτελέσματα, εκτέλεσε το ερώτημα ως διάνυσμα και κατάταξε τα αποτελέσματα χρησιμοποιώντας διανυσματική βαθμολόγηση

Μπορούμε τώρα για τα έγγραφα που εμφανίζονται σε παραπάνω από ένα από τα παραπάνω βήματα να συνδυάσουμε (αθροίσουμε) τους βαθμούς

Πλήρες σύστημα αναζήτησης



Ευρετήρια (παραλλαγές του αντεστραμμένου ευρετηρίου)

Πλήρες σύστημα αναζήτησης

Τι έχουμε ήδη δει:

- Προ-επεξεργασία των εγγράφων
- Ευρετήρια θέσεων (Positional indexes)
- Βαθμιδωτά ευρετήρια (Tiered indexes)
- Διορθώσεις ορθογραφικές (Spelling correction)
- *Ευρετήρα k*-γραμμάτων (για ερωτήματα με * και ορθογραφικές διορθώσεις)
- Επεξεργασία ερωτημάτων
- Βαθμολόγηση εγγράφων

ΤΕΛΟΣ 6ου - 7ου

Κεφαλαίου

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*