Ανάκληση Πληροφορίας

Information Retrieval

Διδάσκων –
Δημήτριος Κατσαρός

Διάλεξη 4η

Phrase queries

Ερωτήματα φράσεως

- Έστω ότι επιθυμούμε ν' απαντήσουμε ερωτήματα της μορφής "***stanford university***" – δηλαδή οι δυο όροι ως ενιαία φράση
- Επομένως, η πρόταση *"I went to university at Stanford"* δεν είναι έγκυρη απάντηση στο ερώτημα αυτό
  - Η έννοια των ερωτημάτων φράσεως είναι εύκολα κατανοητή από τους χρήστες: περίπου το 10% των ερωτημάτων που υποβάλλονται στο Web είναι ερωτήματα φράσεως
- Συνεπώς, δεν επαρκεί ν' αποθηκεύσουμε μόνο ζεύγη της μορφής <*term* : *docs*>

Μια πρώτη προσέγγιση: Biword indexes

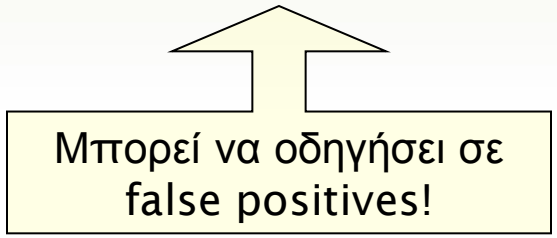
- Φτιάχνουμε ευρετήριο για κάθε ζεύγος συνεχόμενων όρων στο έγγραφο
- Για παράδειγμα, από το κείμενο “Friends, Romans, Countrymen” θα παραγόταν τα ακόλουθα *biwords*
  - ***friends romans***
  - ***romans countrymen***
- Κάθε ένα από αυτά τα biwords θα εισαχθεί πλέον ως όρος στο dictionary
- Τώρα, η επεξεργασία ερωτημάτων που εμπλέκουν φράσεις αποτελούμενες από δυο μόνο όρους είναι εφικτή

Μεγαλύτερα ερωτήματα φράσεως

- Μεγαλύτερες φράσεις θα τις επεργαστούμε με τον ίδιο τρόπο όπως και τα wild-cards (στην επόμενη διάλεξη):
- ***stanford university palo alto*** μπορεί να σπάσει στο Boolean ερώτημα πάνω σε biwords:

stanford university AND university palo AND palo alto

Εάν δεν έχουμε στην διάθεσή μας τα πραγματικά έγγραφα, δεν μπορούμε να πιστοποιήσουμε ότι τα έγγραφα που ταιριάξανε με το προηγούμενο Boolean ερώτημα πράγματι περιέχουν την φράση



Εκτεταμένα biwords

- Κάνουμε parse το indexed κείμενο και εκτελούμε part-of-speech-tagging (POST)
- Ομαδοποιούμε τους όρους σε (έστω) Nouns (N) και articles/prepositions (X)
- Τώρα, όποια ακολουθία όρων της μορφής NX*N ορίζεται ως extended biword
  - Κάθε extended biword εισάγεται ως όρος στο dictionary
- Παράδειγμα : ***catcher in the rye***

  N X X N
- Επεξεργασία ερωτήματος: κάνε parse σε N και X
  - Σπάζουμε το ερώτημα σε enhanced biwords
  - Ψάχνουμε στον index

Ζητήματα με τους biword indexes

- False positives (όπως προαναφέραμε)
- Εκθετική αύξηση του μεγέθους του index, εξαιτίας του μεγαλύτερου dictionary
- Σε έναν extended biword index, κάνουμε parse μεγάλα ερωτήματα σε συζεύξεις:
  - Π.χ., το ερώτημα ***tangerine trees and marmalade skies*** γίνεται parse στο:
  - ***tangerine trees*** *AND* ***trees and marmalade*** *AND* ***marmalade skies***

- Δεν υπάρχει μια λύση για όλα (για κάθε biword)

Λύση 2: Positional indexes

- Αποθηκεύουμε για κάθε ***term***, πλειάδεις της μορφής:

  <number of docs containing ***term***;

  doc1: position1, position2 … ;

  doc2: position1, position2 … ;

  etc.>

Παράδειγμα positional index

<***be***: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, …>

Ποια από τα έγγραφα 1,2,4,5 μπορεί να περιέχουν την φράση "***to be or not to be***"?

- Μπορούμε να συμπιέσουμε τις position values/offsets
- Παρόλ' αυτά, το μέγεθος των postings αυξάνει *ουσιαστικά*

Επεξεργασία ενός phrase query

- Βρίσκουμε από τον inverted index τις λίστες για κάθε διακριτό όρο: ***to, be, or, not***
- Συγχωνεύουμε τις λίστες τους *doc:position* για ν' απαριθμήσουμε όλες τις θέσεις με το "***to be or not to be***".
  - ***to***:
    - *2*:1,17,74,222,551; *4*:8,16,190,429,433; *7*:13,23,191; ...
  - ***be***:
    - *1*:17,19; *4*:17,191,291,430,434; *5*:14,19,101; ...
- Η ίδια γενική μέθοδος για ερωτήματα εγγύτητας (proximity searches)

Ερωτήματα εγγύτητας (proximity queries)

- LIMIT! /3 STATUTE /3 FEDERAL /2 TORT
  Εδώ, το /*k* σημαίνει "within *k* words of"
- Προφανώς, οι positional indexes μπορούν να χρησιμοποιηθούν για τέτοια ερωτήματα, ενώ οι biword indexes δεν μπορούν

- **Άσκηση**: Προσαρμόστε την γραμμική συγχώνευση των postings για να χειριστείτε proximity ερωτήματα
  - Μπορείτε να το κάνετε αυτό να δουλεύει για οποιαδήποτε τιμή του *k*;

Μέγεθος ενός positional index

- Μπορούμε να συμπιέσουμε τις position values/offsets (θα το αναπτύξουμε σε επόμενη διάλεξη)
- Παρόλ' αυτά, το μέγεθος των postings αυξάνει *ουσιαστικά*
- Όμως, χρησιμοποιείται ευρύτατα εξαιτίας τις ισχύως και της χρησιμότητας των phrase και των proximity ερωτημάτων

Μέγεθος ενός positional index

- Χρειαζόμαστε ένα entry για κάθε εμφάνιση του όρου σε κάποιο έγγραφο, όχι απλά ένα entry για κάθε έγγραφο
- Το μέγεθος του index εξαρτάται από το μέγεθος του "μέσου" εγγράφου

Γιατί;

  - Η "μέση" Web σελίδα έχει <1000 όρουςterms
  - SEC filings, βιβλία, ακόμη και κάποια επικά ποιήματα … εύκολα 100,000 όρους
- Θεωρήστε έναν όρο με συχνότητα 0.1%

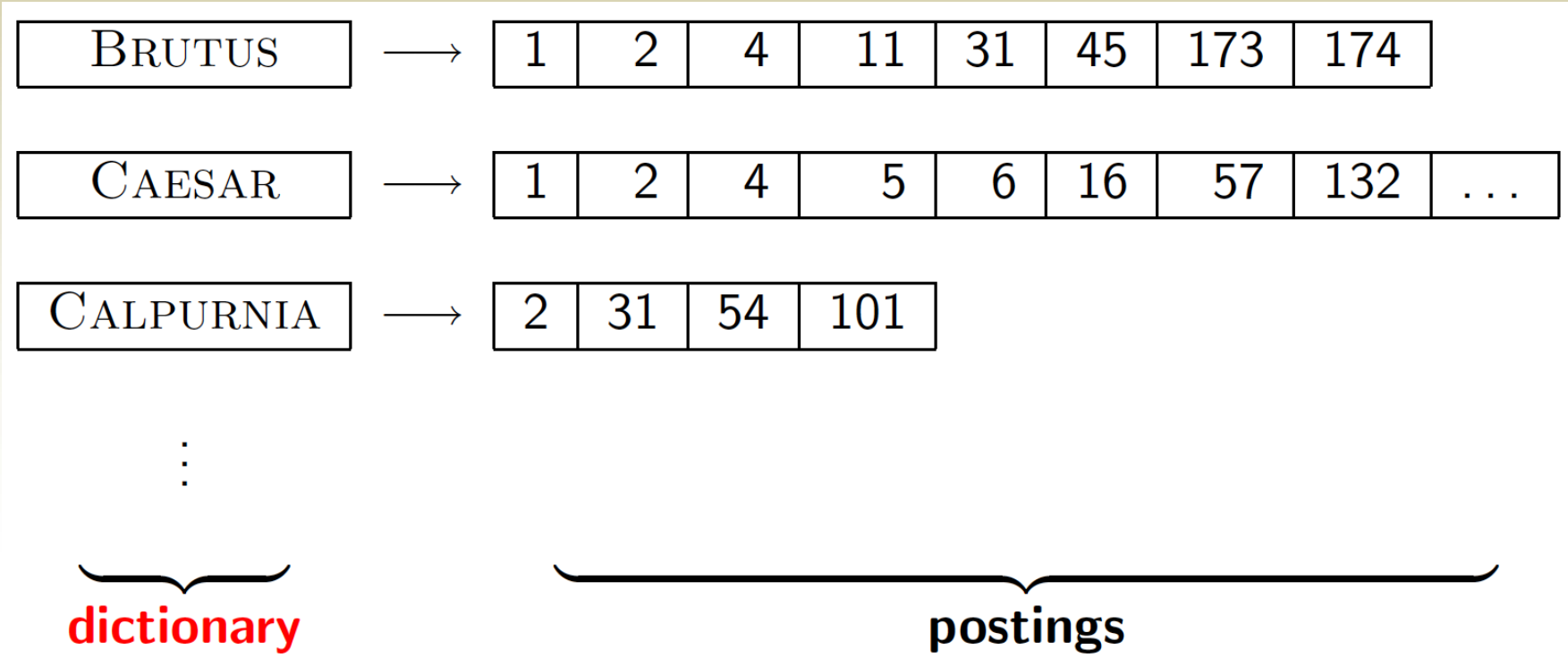
Document size	Postings	Positional postings
1000	1	1
100,000	1	100

Εμπειρικοί κανόνες

- Ένας positional index είναι 2–4 φορές τόσο μεγάλος όσο μεγάλος είναι ένας non-positional index
- Το μέγεθος του positional index είναι 35–50% του μεγέθους της αρχικής συλλογής των εγγράφων
- Τουλάχιστον, αυτά ισχύουν για “αγγλογενείς” γλώσσες

Δομές δεδομένων για το dictionary ενός inverted index

- Η δομή δεδομένων για το dictionary αποθηκεύει το vocabulary των όρων, την document frequency, δείκτη στην αντίστοιχη postings list … ποια μπορεί να είναι η μορφή της δομής δεδομένων;



Ένα απλοϊκό dictionary

- Ως πίνακας από struct:

term	document frequency	pointer to postings list
a	656,265	→
aachen	65	→
. . .	. . .	. . .
zulu	221	→

char[20] int Postings *

20 bytes 4/8 bytes 4/8 bytes

- Πώς το αποθηκεύουμε αποδοτικά στην μνήμη;
- Πώς ψάχνουμε γρήγορα για έναν όρο όταν υποβληθεί κάποιο ερώτημα (at query time);

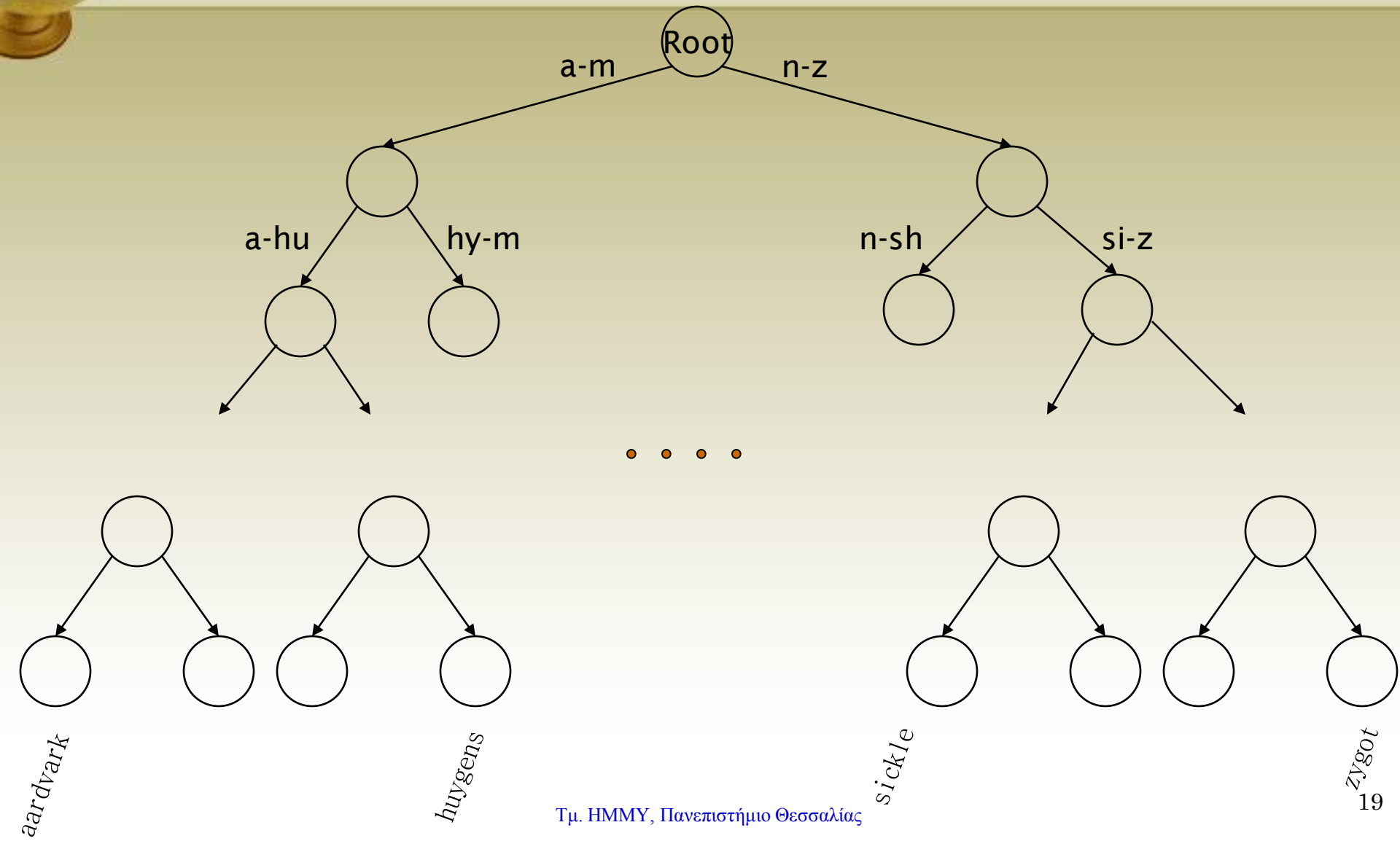
Δομές δεδομένων για το Dictionary

- Δυο κύριες επιλογές:
  - Πίνακας κατακερματισμού (hash table)
  - Δένδρο
- Μερικά συστήματα IR χρησιμοποιούν hashes, κάποια άλλα χρησιμοποιούν δένδρα

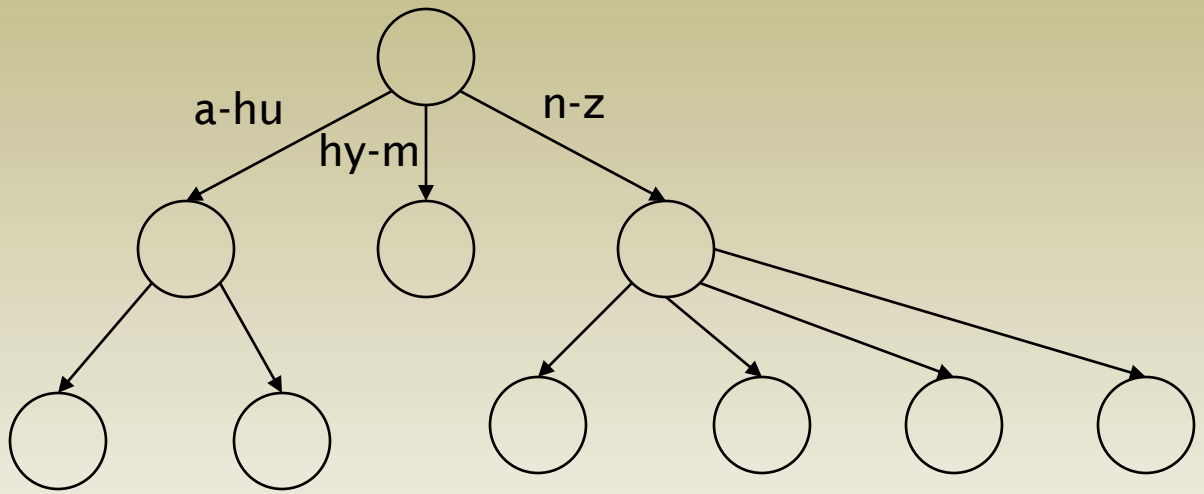
Hashes

- Κάθε όρος στο vocabulary γίνεται hashed σε κάποιον integer
- Πλεονεκτήματα:
  - Η λειτουργία αναζήτησης είναι ταχύτερη από την αντίστοιχη σ' ένα δένδρο: O(1)
- Μειονεκτήματα:
  - Δεν υπάρχει εύκολος τρόπος να βρεθούν όροι με ελαφρές παραλλαγές μεταξύ τους:
    - judgment/judgement
  - Δεν υποστηρίζει prefix search [tolerant retrieval]
  - Εάν το vocabulary μεγαλώνει συνεχώς, προκύπτει η ανάγκη να εκτελείται περιστασιακά η ακριβή λειτουργία του rehashing *όλων*

Δένδρο: Δυαδικό δένδρο (binary tree)



Δένδρο: B-tree



- Ορισμός: Κάθε εσωτερικός κόμβος έχει έναν αριθμό παιδιών με τιμές στο διάστημα [a,b], όπου a, b είναι κατάλληλοι φυσικοί αριθμοί, π.χ., [2,4].

Δένδρα

- Απλούστερο: binary tree
- Πιο σύνηθες: B-tree
- Τα δένδρα απαιτούν εγγενή διάταξη μεταξύ των χαρακτήρων
- Πλεονεκτήματα:
  - Λύνει το prefix problem
- Μειονεκτήματα:
  - Πιο αργά: O(log *M*) [για *balanced* δένδρο μόνο]
  - Η λειτουργία rebalancing σε binary trees είναι ακριβή
    - Τα B-trees αμβλύνουν το rebalancing πρόβλημα