Ευρετηρίαση, Αποθήκευση και Οργάνωση Αρχείων (Indexing, Storage and File Organization)

ΜΕΡΟΣ Ι

Κεφάλαιο 8

Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών
Άνοιξη 2009

Δομές Ευρετηρίου: Διάρθρωση Διάλεξης

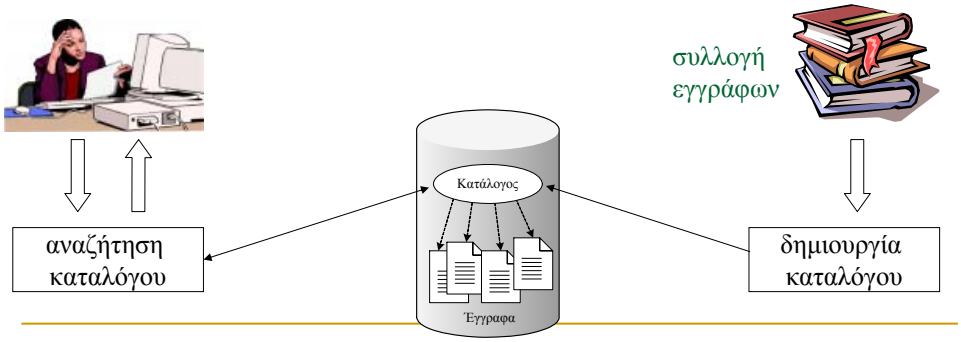
- Εισαγωγή – κίνητρο
- Ανεστραμμένα Αρχεία (Inverted files)
- Αρχεία Υπογραφών (Signature files)
- Δένδρα Καταλήξεων (Suffix trees)

Ευρετηριασμός Κειμένου: Εισαγωγή

- Σκοπός
  - Σχεδιασμός δομών δεδομένων που επιτρέπουν την αποδοτική υλοποίηση της γλώσσας επερώτησης
- Απλοϊκή προσέγγιση: σειριακή αναζήτηση (online sequential search)
  - Ικανοποιητική μόνο αν η συλλογή των κειμένων είναι **μικρή**
  - Είναι η **μόνη** επιλογή αν η συλλογή κειμένων είναι **ευμετάβλητη**
  - Σχεδιασμός δομών δεδομένων, που ονομάζονται ευρετήρια (called *indices*), για επιτάχυνση της αναζήτησης

Χρήση Καταλόγων/Ευρετηρίων

Τα συστήματα ανάκτησης σπάνια αναζητούν την πληροφορία απευθείας στη συλλογή εγγράφων. Συνήθως, χρησιμοποιούνται κατάλογοι οι οποίοι επιταχύνουν τη διαδικασία αναζήτησης.



Ανάγκες Γλωσσών Επερώτησης (και μοντέλων ανάκτησης γενικότερα)

- Απλές
  - βρες έγγραφα που **περιέχουν** μια λέξη t
  - βρες **πόσες φορές** εμφανίζεται η λέξη t σε ένα έγγραφο
  - βρες τις **θέσεις** των εμφανίσεων της λέξης t στο έγγραφο
- Πιο σύνθετες
  - λογικές (Boolean) επερωτήσεις
  - επερωτήσεις εγγύτητας (phrase/proximity queries)
  - ταιριάσματος προτύπου (pattern matching)
  - κανονικές εκφράσεις (regular expressions)
  - δομικές επερωτήσεις (structure-based queries)
  - ...

Σχεδιάζουμε το ευρετήριο ανάλογα με το μοντέλο ανάκτησης και τη γλώσσα επερώτησης

Γενική (Λογική) μορφή ενός ευρετηρίου

Indexing Items (όροι ευρετηρίου)

Documents	k_1	k_2	...	k_j	...	k_t
d_1	$c_{1,1}$	$c_{2,1}$	...	$c_{i,1}$	...	$c_{t,1}$
d_2	$c_{1,2}$	$c_{2,2}$	...	$c_{i,2}$	...	$c_{t,2}$
...	...	...	...	...	...	...
d_i	$c_{1,j}$	$c_{2,j}$	...	$c_{i,j}$	...	$c_{t,j}$
...	...	...	...	...	...	...
d_N	$c_{1,N}$	$c_{2,N}$	...	$c_{i,N}$	...	$c_{t,N}$

c_{ij}: το κελί που αντιστοιχεί στο έγγραφο d_i και στον όρο k_j, το οποίο μπορεί να περιέχει:

- ένα w_{ij} που να δηλώνει την παρουσία ή απουσία του k_j στο d_i (ή τη σπουδαιότητα του k_j στο d_i)
- τις θέσεις στις οποίες ο όρος k_j εμφανίζεται στο d_i (αν πράγματι εμφανίζεται)

Ερωτήματα:

- Τι πρέπει να έχει το κάθε c_{ij}
- Πώς να υλοποιήσουμε αυτή τη λογική δομή ώστε να έχουμε καλή απόδοση;

Τεχνικές Ευρετηριασμού (Indexing Techniques)

- Ανεστραμμένα Αρχεία (Inverted files)
  - η πιο διαδεδομένη τεχνική
- Δένδρα και Πίνακες Καταλήξεων (Suffix trees and arrays)
  - γρήγορες για “phrase queries” αλλά η κατασκευή και η συντήρησή τους είναι δυσκολότερη και ακριβότερη
- Αρχεία Υπογραφών (Signature files)
  - Χρησιμοποιήθηκαν πολύ τη δεκαετία του 80. Σπανιότερα σήμερα – αλλά σε κατανεμημένα διάφορες παραλλαγές τους.

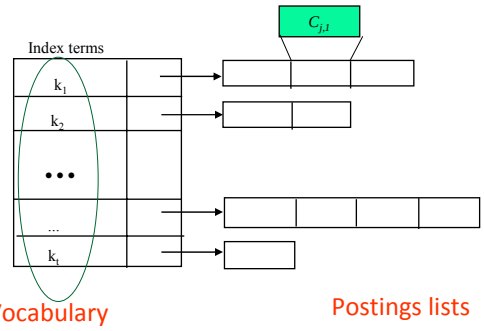
Ανεστραμμένα Αρχεία (Inverted Files)

Ανεστραμμένο Αρχείο

Λογική Μορφή Ευρετηρίου

	k_1	k_2 ...	k_t
d_1	$c_{1,1}$	$c_{2,1}$	$c_{t,1}$
d_2	$c_{1,2}$	$c_{2,2}$	$c_{t,2}$
...		...	
d_i	$c_{1,j}$	$c_{2,j}$	$c_{t,j}$
...		...	
d_N	$c_{1,N}$	$c_{2,N}$	$c_{t,N}$

Μορφή Ανεστραμμένου Ευρετηρίου



Άρα δεν δεσμεύουμε χώρο για **τα «μηδενικά κελιά» της λογικής μορφής του ευρετηρίου**

Inverted Files (Ανεστραμμένα αρχεία)

Inverted file = a word-oriented mechanism for indexing a text collection in order to speed up the searching task.

An inverted file consists of:

- Vocabulary: is the set of all distinct words in the text
- Occurrences: lists containing all information necessary for each word of the vocabulary (documents where the word appears, frequency, text position, etc.)
  - **Τι είδους πληροφορία κρατάμε στις posting lists εξαρτάται από το λογικό μοντέλο και το μοντέλο ερωτήσεων**

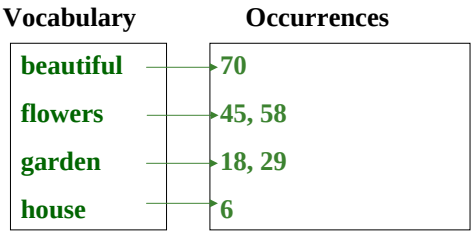
Ανεστραμμένο αρχείο για ένα μόνο έγγραφο και αποθήκευση θέσεων εμφάνισης κάθε λέξης

Κείμενο

That house has a garden. The garden has many flowers. The flowers are beautiful

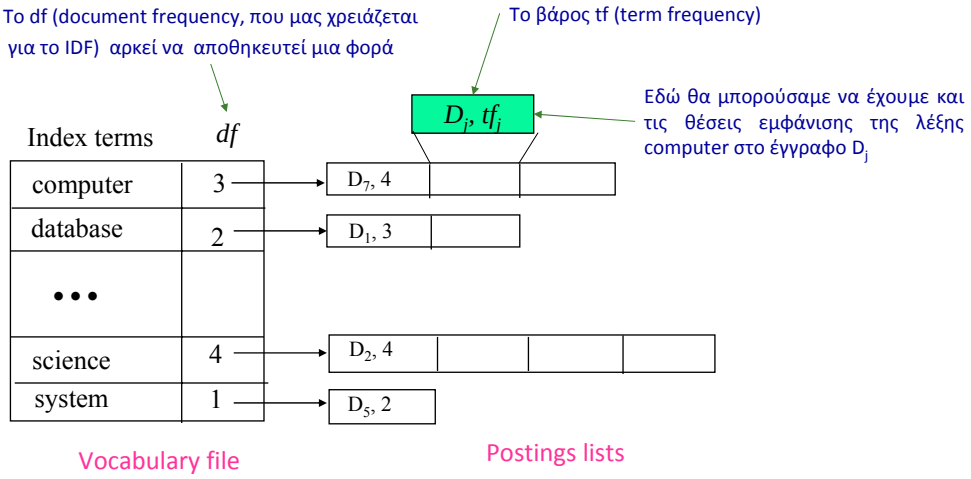
1 6 12 16 18 25 29 36 40 45 54 58 66 70

Inverted File:



Τι άλλο θα κάνατε (κρατούσατε) αν είχαμε πολλά έγραφα και θέλαμε να υλοποιήσουμε το Διανυσματικό Μοντέλο;

Ανεστραμμένο αρχείο για πολλά έγγραφα, και βάρυνση tf-idf



Παράδειγμα ανεστραμμένου αρχείου όπου για κάθε λέξη i και έγγραφο j κρατάμε μόνο το $freq_{ij}$

Document Corpus

Doc	Text
1	Pease porridge hot
2	Pease porridge cold
3	Pease porridge in the pot
4	Pease porridge hot, pease porridge not cold
5	Pease porridge cold, pease porridge not hot
6	Pease porridge hot in the pot

Inverted File

Vocabulary	Inverted Lists
cold	<2,1> <4,1> <5,1>
hot	<1,1> <4,1> <5,1> <6,1>
in	<3,1> <6,1>
not	<4,1> <5,1>
pease	<1,1> <2,1> <3,1> <4,2> <5,2> <6,1>
porridge	<1,1> <2,1> <3,1> <4,2> <5,2> <6,1>
pot	<3,1> <6,1>
the	<3,1> <6,1>

Another example

	term	df	document ids
1	Algorithms	3	: 3 5 7
2	Application	2	: 3 17
3	Delay	2	: 11 12
4	Differential	8	: 4 8 10 11 12 13 14 15
5	Equations	10	: 1 2 4 8 10 11 12 13 14 15
6	Implementation	2	: 3 7
7	Integral	2	: 16 17
8	Introduction	2	: 5 6
9	Methods	2	:
10	Nonlinear	2	: 9 13
11	Ordinary	2	: 8 10
12	Oscillation	2	: 11 12
13	Partial	2	: 4 13
14	Problem	2	: 6 7
15	Systems	3	: 6 8 9
16	Theory	4	: 3 11 12 17

Block Addressing

Στην περίπτωση που θέλουμε να κρατήσουμε και τη θέση εμφάνισης κάθε όρου στο κείμενο

- The text is divided in blocks
- The occurrences point to the blocks where the word appears

Block Addressing: Example

That house has a garden. The garden has many flowers. The flowers are beautiful

1 6 12 16 18 25 29 36 40 45 54 58 66 70

Vocabulary	Occurrences
beautiful	70
flowers	45, 58
garden	18, 29
house	6

Block 1	Block 2	Block 3	Block 4
That house has a	garden. The garden has	many flowers. The flowers	are beautiful

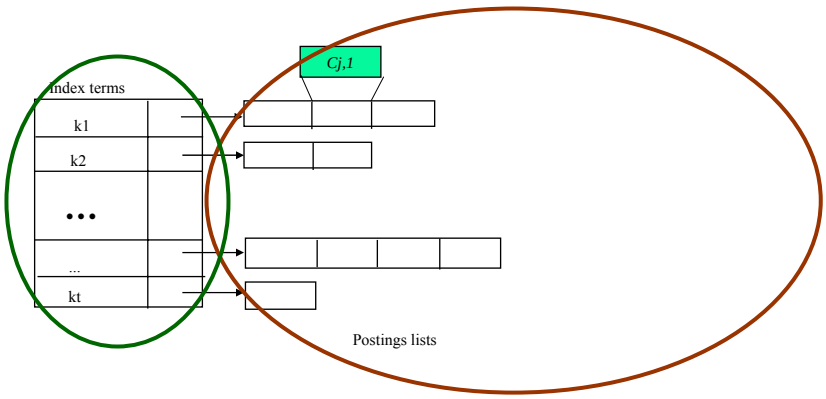
Vocabulary	Occurrences
beautiful	4
flowers	3
garden	2
house	1

Block Addressing

- Advantages:
  - the number of pointers is smaller than positions
  - all the occurrences of a word inside a single block are collapsed to one reference
  - (indices of only 5% overhead over the text size can be obtained with this technique. Of course this depends on the block size).
    - In many cases instead of defining the block size, we define the number of blocks (in this way we know how many bits we need per pointer)
- Disadvantages:
  - online sequential search over the qualifying blocks if exact positions are required
    - e.g. for finding the sentence where the word occurs
    - e.g. for evaluating a context (phrasal or proximity) query

Ανεστραμμένα Αρχεία: Απαιτήσεις Χώρου

μικρές μεγάλες



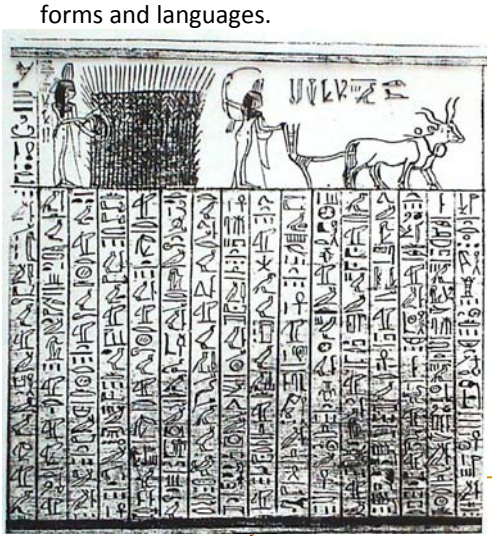
ΠΑΡΕΝΘΕΣΗ
Στατιστικά Κειμένου
Text Statistics

Διάρθρωση

- Συχνότητα Εμφάνισης Λέξεων
- Ο Νόμος του Zipf
- Ο Νόμος του Heaps

Γραπτός Λόγος - Κείμενο

Starting with hieroglyphs, the first written surfaces (stone, wood, animal skin, papyrus and rice paper), and paper, text has been created everywhere, in many forms and languages.



21

Στατιστικές Ιδιότητες Κειμένου

- How is the frequency of different words distributed?
- How fast does vocabulary size grow with the size of a corpus?

Such factors affect the performance of information retrieval and can be used to select appropriate term weights and other aspects of an IR system.

Συχνότητα Λέξεων

- A few words are very common.
  - 2 most frequent words (e.g. “the”, “of”) can account for about 10% of word occurrences.
- Most words are very rare.
  - Half the words in a corpus appear *only once*, called *hapax legomena* (Greek for “read only once”)
- Called a “heavy tailed” distribution, since most of the probability mass is in the “tail”

Sample Word Frequency Data (from B. Croft, UMass)

Frequent Word	Number of Occurrences	Percentage of Total
the	7,398,934	5.9
of	3,893,790	3.1
to	3,364,653	2.7
and	3,320,687	2.6
in	2,311,785	1.8
is	1,559,147	1.2
for	1,313,561	1.0
The	1,144,860	0.9
that	1,066,503	0.8
said	1,027,713	0.8

Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words

Ο νόμος του Zipf

Rank *r of a word*: The numerical position of the word in a list sorted by decreasing frequency (*f*).

Zipf (1949) "discovered" that: the frequency of any word is inversely proportional to its rank

$$f \cdot r = k \quad (\text{for constant } k)$$

Πχ:

- *f1 * 1 = k*
- *f2 * 2 = k*
- *f3 * 3 = k*
- ...
- *fi * i = k*
- *= f1 * 1 = f1 ⇔ fi = f1 / i*

The most frequent word will appear twice as often as the second most frequent word, which occurs twice as often as the fourth, etc

- Η συχνότητα της i-th πιο συχνά εμφανιζόμενης λέξης είναι 1/i φορές η συχνότητα της πιο συχνής.
- *Πιο ακριβές:* ***1/i^θ*** *όπου θ μεταξύ 1.5 και 2*

Sample Word Frequency Data (again) (from B. Croft, UMass)

Frequent Word	Number of Occurrences	Percentage of Total	
the	7,398,934	5.9	1 * 5.9 = 5.9
of	3,893,790	3.1	2 * 3.1 = 6.2
to	3,364,653	2.7	3 * 2.7 = 8.1
and	3,320,687	2.6	4 * 2.6 = 10.4
in	2,311,785	1.8	5 * 1.8 = 9
is	1,559,147	1.2	6 * 1.2 = 7.2
for	1,313,561	1.0	7 * 1 =7
The	1,144,860	0.9	8 * 0.9 =7.2
that	1,066,503	0.8	9 * 0.8 =7.2
said	1,027,713	0.8	...

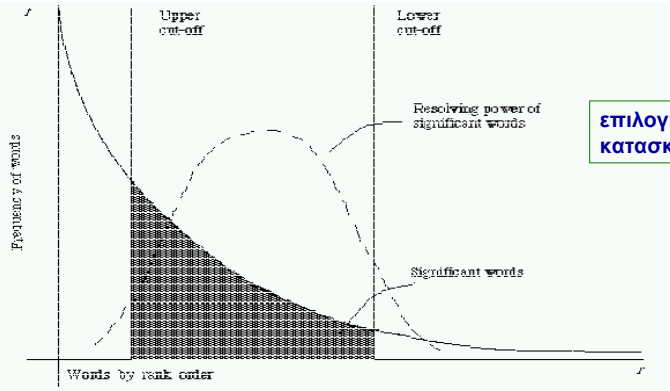
Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words

Zipf's Law Impact on IR

- Good News: Stopwords will account for a large fraction of text so eliminating them greatly reduces inverted-index storage costs.
- Bad News: For most words, gathering sufficient data for meaningful statistical analysis (e.g. for correlation analysis for query expansion) is difficult since they are extremely rare.

Zipf and Term Weighting

- Luhn (1958) suggested that both *extremely common* and *extremely uncommon* words were not very useful for indexing.



επιλογή όρων στην αυτόματη κατασκευή θησαυρών

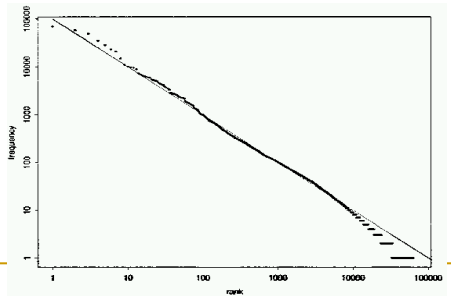
Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adapted from Schultz[68] page 122)

Does Real Data Fit Zipf's Law?

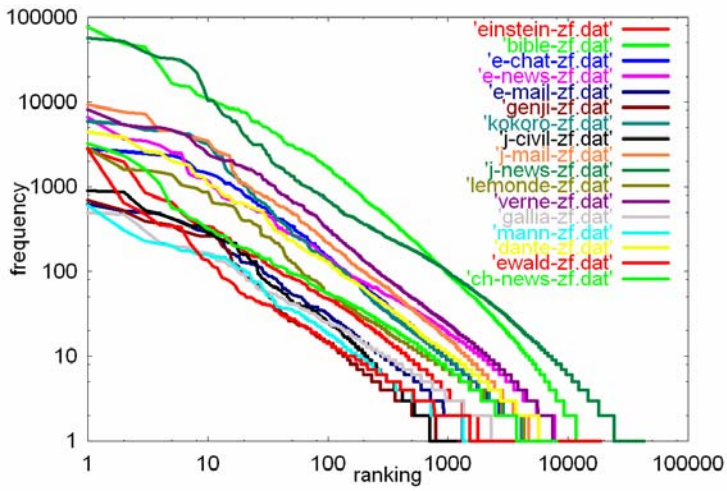
- A law of the form $y = kx^c$ is called a power law.
  - Zipf's law ($fi=f1/i$) is a power law with $c = -1$
- On a log-log plot, power laws give a straight line with slope c.

$$\log(y) = \log(kx^c) = \log k + c\log(x) = \log k - \log(x)$$

Zipf is quite accurate except for very high and low rank.



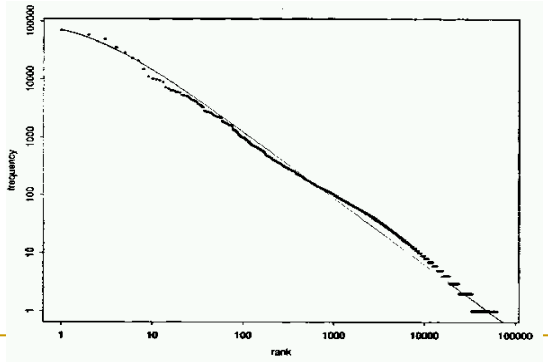
Does Real Data Fit Zipf's Law?



Σημείωση: Ο Χ και Υ έχουν λογαριθμική κλίμακα

Mandelbrot (1954) Correction

- *Ziph's Law:* $fi = f1/i^{\vartheta}$
- Mandelbrot correction: $fi = f1*k/(c+i)^{\vartheta}$
  - *c: parameter*
  - *k: so that all frequencies add to N*
  - This formula fits better with the read texts



Mandelbrot's function on Brown corpus

Explanations for Zipf's Law

- Zipf's explanation was his "principle of least effort." Balance between speaker's desire for a small vocabulary and hearer's desire for a large one.
  - Η επανάληψη λέξεων είναι ευκολότερη από την επινόηση/χρήση νέων
- Debate (1955-61) between Mandelbrot and H. Simon over explanation.
- Με επιφύλαξη:
  - Li (1992) shows that just random typing of letters including a space will generate "words" with a Zipfian distribution.

http://www.nslij-genetics.org/wli/zipf/

Vocabulary Growth

- How does the size of the overall vocabulary (number of unique words) grow with the size of the corpus?
- This determines how the size of the inverted index will scale with the size of the corpus.
- Vocabulary not really upper-bounded due to proper names, typos, etc.

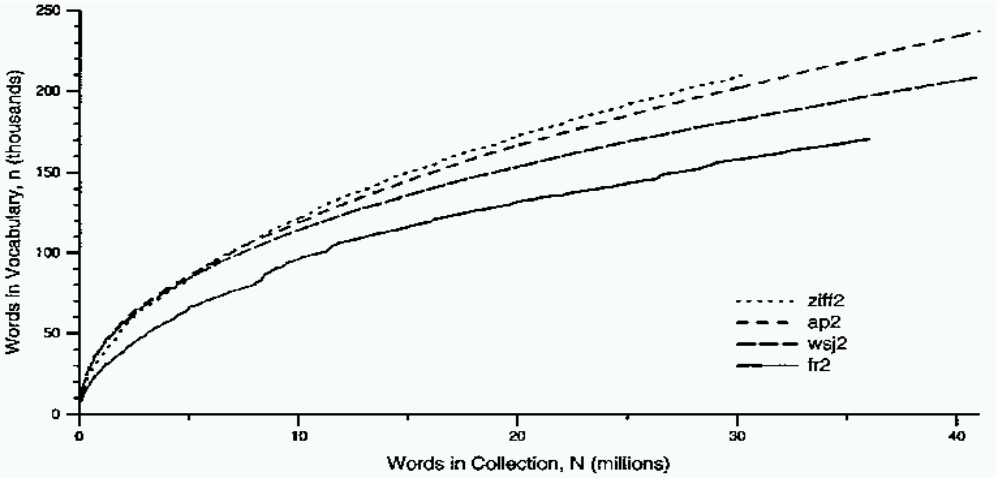
Heaps' Law

- If *V* is the size of the vocabulary (i.e. number of distinct words) and the *n* is the length of the corpus in words:

$$V = Kn^{\beta} \quad \text{with constants } K,\ 0 < \beta < 1$$

  - Typical constants:
    - $K \approx 10–100$
    - $\beta \approx 0.4–0.6$ (approx. square-root)

Heaps' Law Data



Heaps' Law

- Explanation for Heaps' Law
  - Can be derived from Zipf's law by assuming documents are generated by randomly sampling words from a Zipfian distribution

Heaps' law also means that:

as more instance text is gathered, there will be diminishing returns in terms of discovery of the full vocabulary from which the distinct terms are drawn.

Word Length

- Average Length of Words
  - Why? To estimate the storage space needed for the vocabulary.
  - Average word length in TREC-2 = 5 letters
  - If we remove stopwords then average word length: 6-7 letters

Επίσης

- Σνωμφύα με μια ένυερα του Κέμπριτζ η σιερά των γμμάαρωτν σε μια λξέη δεν έεχι σησίμαα. Ακρεί το πώτρο και το ταίυελετο γμαράμ να είανι στη σστωή σεριά.
- Σύμφωνα με μια έρευνα του Κέμπριτζ η σειρά των γραμμάτων σε μια λέξη δεν έχει σημασία. Αρκεί το πρώτο και το τελευταίο γράμμα να είναι στη σωστή σειρά.

Κλείσιμο ΠΑΡΕΝΘΕΣΗΣ

Ανεστραμμένα Αρχεία: Απαιτήσεις Χώρου

Notations
- *n*: the size of the text (of all documents in the collection)
- *V*: the size of the vocabulary

For the Vocabulary:
- Rather **small**.
- According to *Heaps'* law the vocabulary grows as $O(n^{\beta})$, where β is a constant between 0.4 and 0.6 in practice. So V ~ sqrt(n) // άρα ανάλογο της τετραγωνικής ρίζας του μεγέθους της συλλογής)

For Occurrences (posting lists):
- **Much more** space.
- Since each word appearing in the text is referenced once in that structure (i.e. we keep a pointer), the extra space is *O(n)*

(To reduce space requirements, a technique called *block addressing* is used)

Size of Inverted Files as percentage of the size of the whole collection

45% of all words are stopwods

Index	Small collection (1Mb)		Medium collection (200Mb)		Large collection (2Gb)	
Addressing words	**45%**	**73%**	**36%**	**64%**	**35%**	**63%**
Addressing 64K blocks	**27%**	**41%**	**18%**	**32%**	**5%**	**9%**
Addressing 256 blocks	**18%**	**25%**	**1.7%**	**2.4%**	**0.5%**	**0.7%**
	Without stopwords	All words	Without stopwords	All words	Without stopwords	All words

Addressing words: 4 bytes per pointer (2^32 ~ giga)

Addressing 64K blocks: 2 bytes per pointer

Addressing 256 blocks: 1 byte per pointer