ΕΠΛ660

Ανάκτηση Πληροφοριών και Μηχανές Αναζήτησης

Μάριος Δ. Δικαιάκος και Γιώργος Πάλλης

Στο προηγούμενο μάθημα…

ΕΠΛ660

- Overview of course topics
- Introduction to Information Retrieval
- Basic inverted indexes:
  - Dictionary and Postings
- Boolean query processing

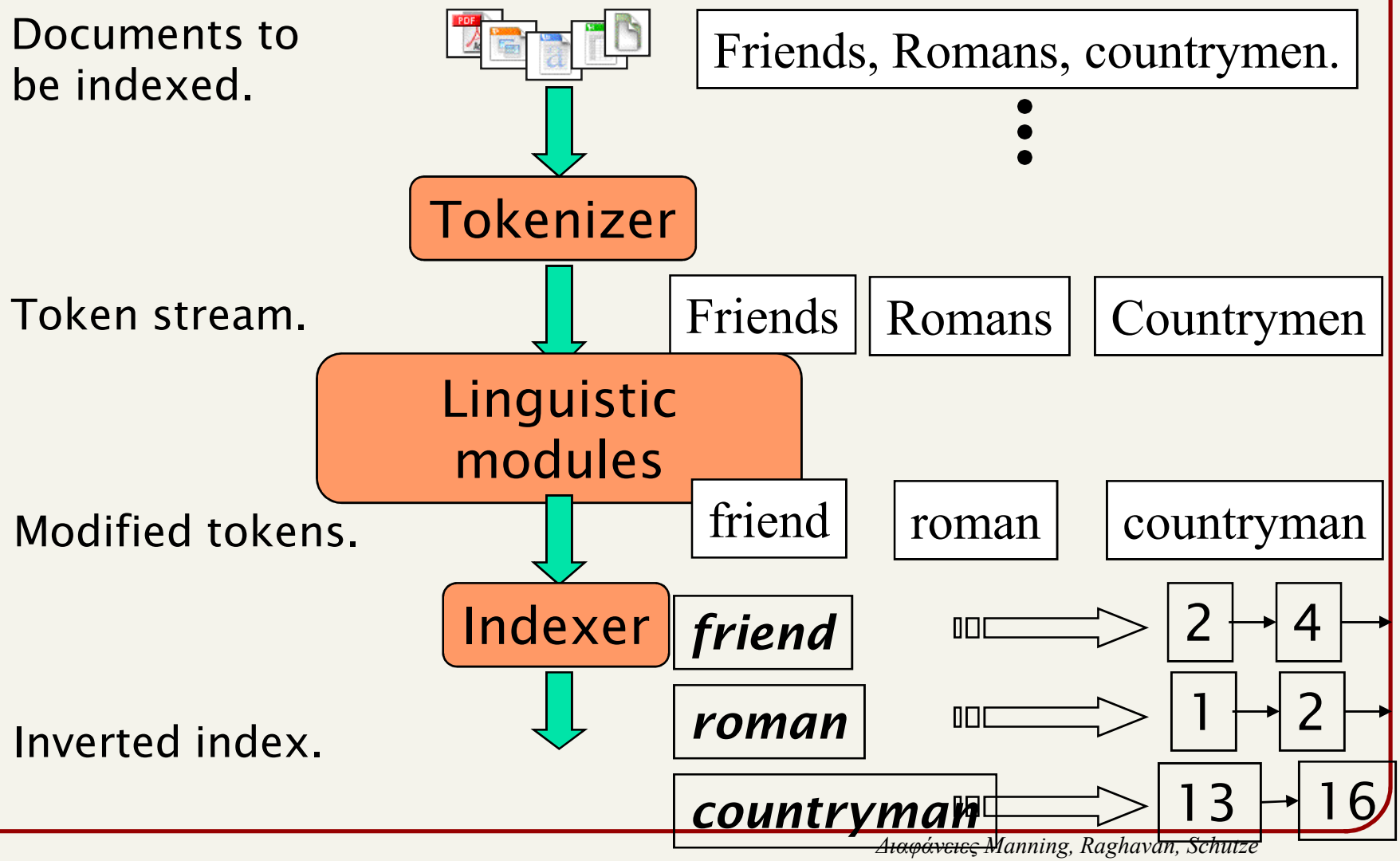
Plan for this lecture

ΕΠΛ660

Finish basic indexing

- Determining the vocabulary of terms
  - Tokenization
  - Normalization
  - Stop words
  - Stemming and lemmatization
- Postings
  - Skip pointers
  - Biword indexes/phrase queries
  - Positional indexes

Basic indexing pipeline



Parsing a document

ΕΠΛ660

The first step is to convert the byte sequence of a file into a linear sequence of characters

We need to determine the correct encoding

- What format is it in?
  - pdf/word/excel/html?
- What language is it in?
- What character set is in use?

Each of these is a classification problem, which we will study later in the course.

But these tasks are often done heuristically …

Complications: Format/language

ΕΠΛ660

- Documents being indexed can include docs from many different languages
  - A single index may have to contain terms of several languages.
- Sometimes a document or its components can contain multiple languages/formats
  - French email with a German pdf attachment.

Complications: Format/language

ΕΠΛ660

- The next step is to determine the **document unit** for indexing.

- What is a unit document?
  - A file?
  - An email? (Perhaps one of many in an mbox.)
  - An email with 5 attachments?
  - A group of files (PPT or LaTeX in HTML)

- Indexing granularity

Some Definitions

ΕΠΛ660

- **Word** – A string of characters as it appears in the text.
- **Term** – A "normalized" word (case, morphology, spelling etc); an equivalence class of words.
- **Token** – An instance of a word or term occurring in a document.
- **Type** – A class of all tokens containing the same character sequence.

Determining the vocabulary of terms

ΕΠΛ660

- If the document to be indexed is:
  - To sleep perchance to dream

- 5 tokens
- 4 types
- 3 terms

What are the correct tokens to emit?

Tokenization

ΕΠΛ660

Tokenization is the task of chopping a character sequence up into pieces, called tokens

- Input: "***Friends, Romans, Countrymen***"
- Output: Tokens
  - ***Friends***
  - ***Romans***
  - ***Countrymen***
- Each such token is now a candidate for an index entry, after further processing

Tokenization

ΕΠΛ660

- For O'Neill which of the following is the desired tokenization?
  - neill
  - Oneill
  - O'neill
  - O' neill
  - O neill

- How many cases would a query of o'neill AND capital match?

Tokenization

ΕΠΛ660

- Issues in tokenization:
  - ***Finland's capital*** →
    Finland? Finlands? Finland's?
  - ***Hewlett-Packard*** → ***Hewlett*** and ***Packard*** as two tokens?
    - ***State-of-the-art***: break up hyphenated sequence.
    - ***co-education*** ?
    - ***the hold-him-back-and-drag-him-away-maneuver*** ?
    - It's effective to get the user to put in possible hyphens
  - ***San Francisco***: one token or two? How do you decide it is one token?

Numbers

- *3/12/91* *Mar. 12, 1991*
- *55 B.C.*
- *B-52*
- *My PGP key is 324a3df234cb23e*
- *100.2.86.144*
  - Often, don't index as text.
    - But often very useful: think about things like looking up error codes/stacktraces on the web
    - (One answer is using n-grams)
  - Will often index "metadata" separately
    - Creation date, format, etc.

Metadata

ΕΠΛ660

- On Metadata
  - Often included in Web pages
  - Hidden from the browser, but useful for indexing
- Information about a document that may not be a part of the document itself (data about data).
- *Descriptive* metadata is external to the meaning of the document:
  - Author
  - Title
  - Source (book, magazine, newspaper, journal)
  - Date
  - ISBN
  - Publisher
  - Length

Web Metadata

ΕΠΛ660

- META tag in HTML
  - <META NAME="keywords" CONTENT="pets, cats, dogs">
- META "HTTP-EQUIV" attribute allows server or browser to access information:
  - <META HTTP-EQUIV="content-type" CONTENT="text/tml; charset=EUC-2">
  - <META HTTP-EQUIV="expires" CONTENT="Tue, 01 Jan 02">
  - <META HTTP-EQUIV="creation-date" CONTENT="23-Sep-01">

Tokenization: Language issues

ΕΠΛ660

- ***L'ensemble*** → one token or two?
  - ***L*** ? ***L'*** ? ***Le*** ?
  - Want ***l'ensemble*** to match with ***un ensemble***

- German noun compounds are not segmented
  - Lebensversicherungsgesellschaftsangestellter
  - 'life insurance company employee'

Tokenization: language issues

- Chinese and Japanese have no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - Not always guaranteed a unique tokenization
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana Hiragana Kanji Romaji

End-user can express query entirely in hiragana!

Tokenization: language issues

- Arabic (or Hebrew) is basically written right to left, but with certain items like numbers written left to right
- Words are separated, but letter forms within a word form complex ligatures
- استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.
- ← → ← → ← start
- 'Algeria achieved its independence in 1962 after 132 years of French occupation.'
- With Unicode, the surface presentation is complex, but the stored form is straightforward

Normalization

ΕΠΛ660

- Token normalization is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens
- Need to "normalize" terms in indexed text as well as query terms into the same form
  - We want to match ***U.S.A.*** and ***USA***

Normalization

ΕΠΛ660

- We most commonly implicitly define equivalence classes of terms
  - e.g., anti-discriminatory and antidiscriminatory
- Maintain relations between unnormalized tokens
  - Handle synonyms and homonyms
    - Hand-constructed equivalence classes
      - e.g., ***car*** = ***automobile***
      - ***color*** = ***colour***
  - Index such equivalences
    - When the document contains ***automobile***, index it under ***car*** as well (usually, also vice-versa)
  - Or expand query?
    - When the query contains ***automobile***, look under ***car*** as well

Normalization

ΕΠΛ660

- Alternative is to do asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows***
  - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient

Normalization: Case Folding

ΕΠΛ660

- Reduce all letters to lower case
  - Often best to lower case everything, since users will use lowercase regardless of 'correct' capitalization…
  - exception: upper case (in mid-sentence?)
    - e.g., ***General Motors***
    - ***Bush***
    - ***C.A.T.*** vs. ***cat***
  - A machine learning sequence model which uses features to make the decision of when to case-fold → truecasing

Normalization: Other Languages

ΕΠΛ660

- 60% of Web pages are in English
- 2/3 of blogs are not in English
- Accents: ***résumé*** vs. ***resume***.
- Most important criterion:
  - How are your users like to write their queries for these words?
- Even in languages that standardly have accents, users often may not type them
- German: Tuebingen vs. Tübingen
  - Should be equivalent
- Equate all words to a form without diacritics

Normalization: Other Languages

ΕΠΛ660

- Document collections include documents from many different languages
- Character-level alphabet detection
  - Language-particular tokenization and normalization rules are applied.
  - Sometimes ambiguous:

Morgen will ich in MIT …

Is this German "mit"?

Language Identification Classifier

ensemble.french
時間.japanese
MIT.english
mit.german
guaranteed.english
entries.english
sometimes.english
tokenization.english

These may be grouped by language (or not…).

Soundex

ΕΠΛ660

- Traditional class of heuristics to expand a query into phonetic equivalents
  - Language specific – mainly for names
  - E.g., ***chebyshev*** → ***tchebycheff***

Tokenizing HTML

ΕΠΛ660

- Should text in HTML commands not typically seen by the user be included as tokens?
  - Words appearing in URLs.
  - Words appearing in “meta text” of images.
- Simplest approach is to exclude all HTML tag information (between “<“ and “>”) from tokenization.

Google spam

ΕΠΛ660

Google Contact Us

Home

About Google

Contact Us

Help us maintain the quality of Google search results.

We work hard to return the most relevant results for every search we conduct. To that end, we encourage site managers to make their content straightforward and easily understood by users and search engines alike. Unfortunately, not all websites have users' best interests at heart. Trying to deceive (spam) our web crawler by means of hidden text, deceptive cloaking or doorway pages compromises the quality of our results and degrades the search experience for everyone.

We think that's a bad thing, and so we request that, if your Google search returns a result that you suspect is spam, you please let us know by using this form. We thoroughly investigate every report of deceptive practices and take appropriate action when we uncover genuine abuse. In especially egregious cases, we will remove spammers from our index immediately, so they don't show up in search results at all. At a minimum we'll use the data from each spam report to improve our site ranking and filtering algorithms, which, over time, should increase the quality of our results.

We appreciate your taking the time to help us improve our service for your fellow users around the world. By helping us eliminate spam, you're saving millions of people time, effort and energy.

Report a Spam Result

Exact query that shows a problem (copy this from the Google search box):

Resulting Google page that shows problem (copy the Google URL):

The specific web page or site that is misbehaving:

Type(s) of problem (check all that apply):

- ☐ Hidden text or links
- ☐ Misleading or repeated words
- ☐ Page does not match Google's description
- ☐ Cloaked page
- ☐ Deceptive redirects
- ☐ Doorway pages
- ☐ Duplicate site or pages
- ☐ Other (specify)

Additional details:

www.google.com/contact/spamreport.html

Stop words (λέξεις αποκλεισμού)

ΕΠΛ660

- With a stop list, you exclude from dictionary entirely the most frequent words. Intuition:
  - They have little semantic content: *the, a, and, to, be*
  - They take a lot of space: ~30% of postings for top 30
- But the trend is away from doing this:
  - Good compression techniques means the space for including stopwords in a system is very small
  - Good query optimization techniques mean you pay little at query time for including stop words.
  - You need them for:
    - Phrase queries: "King of Denmark"
    - Various song titles, etc.: "Let it be", "To be or not to be"
    - "Relational" queries: "flights to London"

Stop words in Greek [source: CELEX]

ΕΠΛ660

- ΑΔΙΑΚΟΠΑ, ΑΙ, ΑΚΟΜΑ, ΑΚΟΜΗ, ΑΚΡΙΒΩΣ, ΑΛΗΘΕΙΑ, ΑΛΗΘΙΝΑ, ΑΛΛΑ, ΑΛΛΑΧΟΥ, ΑΛΛΕΣ, ΑΛΛΗ, ΑΛΛΗΝ, ΑΛΛΗΣ, ΑΛΛΙΩΣ, ΑΛΛΙΩΤΙΚΑ, ΑΛΛΟ, ΑΛΛΟΙ, ΑΛΛΟΙΩΣ, ΑΛΛΟΙΩΤΙΚΑ, ΑΛΛΟΝ, ΑΛΛΟΣ, ΑΛΛΟΤΕ, ΑΛΛΟΥ, ΑΛΛΟΥΣ, ΑΛΛΩΝ, ΑΜΑ, ΑΜΕΣΑ, ΑΜΕΣΩΣ, ΑΝ, ΑΝΑ, ΑΝΑΜΕΣΑ, ΑΝΑΜΕΤΑΞΥ, ΑΝΕΥ, ΑΝΤΙ, ΑΝΤΙΠΕΡΑ, ΑΝΤΙΣ, ΑΝΩ, ΑΝΩΤΕΡΩ, ΑΞΑΦΝΑ, ΑΠ, ΑΠΕΝΑΝΤΙ, ΑΠΟ, ΑΠΟΨΕ, ΑΡΑ, ΑΡΑΓΕ, ΑΡΓΑ, ΑΡΓΟΤΕΡΟ, ΑΡΙΣΤΕΡΑ, ΑΡΚΕΤΑ, ΑΡΧΙΚΑ, ΑΣ, ΑΥΡΙΟ, ΑΥΤΑ, ΑΥΤΕΣ, ΑΥΤΗ, ΑΥΤΗΝ, ΑΥΤΗΣ, ΑΥΤΟ, ΑΥΤΟΙ, ΑΥΤΟΝ, ΑΥΤΟΣ, ΑΥΤΟΥ, ΑΥΤΟΥΣ, ΑΥΤΩΝ, ΑΦΟΤΟΥ, ΑΦΟΥ, ΒΕΒΑΙΑ, ΒΕΒΑΙΟΤΑΤΑ, ΓΙ, ΓΙΑ, ΓΡΗΓΟΡΑ, ΓΥΡΩ, ΔΑ, ΔΕ, ΔΕΙΝΑ, ΔΕΝ, ΔΕΞΙΑ, ΔΗΘΕΝ, ΔΗΛΑΔΗ, ΔΙ, ΔΙΑ, ΔΙΑΡΚΩΣ, ΔΙΚΑ, ΔΙΚΟ, ΔΙΚΟΙ, ΔΙΚΟΣ, ΔΙΚΟΥ, ΔΙΚΟΥΣ, ΔΙΟΛΟΥ, ΔΙΠΛΑ, ΔΙΧΩΣ, ΕΑΝ, ΕΑΥΤΟ, ΕΑΥΤΟΝ, ΕΑΥΤΟΥ, ΕΑΥΤΟΥΣ, ΕΑΥΤΩΝ, ΕΓΚΑΙΡΑ, ΕΓΚΑΙΡΩΣ, ΕΓΩ, ΕΔΩ, ΕΙΔΕΜΗ, ΕΙΘΕ, ΕΙΜΑΙ, ΕΙΜΑΣΤΕ, ΕΙΝΑΙ, ΕΙΣ, ΕΙΣΑΙ, ΕΙΣΑΣΤΕ, ΕΙΣΤΕ, ΕΙΤΕ, ΕΙΧΑ, ΕΙΧΑΜΕ, ΕΙΧΑΝ, ΕΙΧΑΤΕ, ΕΙΧΕ, ΕΙΧΕΣ, ΕΚΑΣΤΑ, ΕΚΑΣΤΕΣ, ΕΚΑΣΤΗ, ΕΚΑΣΤΗΝ, ΕΚΑΣΤΗΣ, ΕΚΑΣΤΟ, ΕΚΑΣΤΟΙ, ΕΚΑΣΤΟΝ, ΕΚΑΣΤΟΣ, ΕΚΑΣΤΟΥ, ΕΚΑΣΤΟΥΣ, ΕΚΑΣΤΩΝ, ΕΚΕΙ, ΕΚΕΙΝΑ, ΕΚΕΙΝΕΣ, ΕΚΕΙΝΗ, ΕΚΕΙΝΗΝ, ΕΚΕΙΝΗΣ, ΕΚΕΙΝΟ, ΕΚΕΙΝΟΙ, ΕΚΕΙΝΟΝ, ΕΚΕΙΝΟΣ, ΕΚΕΙΝΟΥ, ΕΚΕΙΝΟΥΣ, ΕΚΕΙΝΩΝ, ΕΚΤΟΣ, ΕΜΑΣ, ΕΜΕΙΣ, ΕΜΕΝΑ, ΕΜΠΡΟΣ, ΕΝ, ΕΝΑ, ΕΝΑΝ, ΕΝΑΣ, ΕΝΟΣ, ΕΝΤΕΛΩΣ, ΕΝΤΟΣ, ΕΝΤΩΜΕΤΑΞΥ, ΕΝΩ, ΕΞ, ΕΞΑΦΝΑ, ΕΞΗΣ, ΕΞΙΣΟΥ, ΕΞΩ, ΕΠΑΝΩ, ΕΠΕΙΔΗ, ΕΠΕΙΤΑ, ΕΠΙ, ΕΠΙΣΗΣ, ΕΠΟΜΕΝΩΣ, ΕΣΑΣ, ΕΣΕΙΣ, ΕΣΕΝΑ, ΕΣΤΩ, ΕΣΥ, ΕΤΕΡΑ, ΕΤΕΡΑΙ, ΕΤΕΡΑΣ, ΕΤΕΡΕΣ, ΕΤΕΡΗ, ΕΤΕΡΗΣ, ΕΤΕΡΟ, ΕΤΕΡΟΙ, ΕΤΕΡΟΝ, ΕΤΕΡΟΣ, ΕΤΕΡΟΥ, ΕΤΕΡΟΥΣ, ΕΤΕΡΩΝ, ΕΤΟΥΤΑ, ΕΤΟΥΤΕΣ, ΕΤΟΥΤΗ, ΕΤΟΥΤΗΝ, ΕΤΟΥΤΗΣ, ΕΤΟΥΤΟ, ΕΤΟΥΤΟΙ, ΕΤΟΥΤΟΝ, ΕΤΟΥΤΟΣ, ΕΤΟΥΤΟΥ, ΕΤΟΥΤΟΥΣ, ΕΤΟΥΤΩΝ, ΕΤΣΙ, ΕΥΓΕ, ΕΥΘΥΣ, ΕΥΤΥΧΩΣ, ΕΦΕΞΗΣ, ΕΧΕΙ, ΕΧΕΙΣ, ΕΧΕΤΕ, ΕΧΘΕΣ, ΕΧΟΜΕ, ΕΧΟΥΜΕ, ΕΧΟΥΝ, ΕΧΤΕΣ, ΕΧΩ, ΕΩΣ, Η, ΗΔΗ, ΗΜΑΣΤΑΝ, ΗΜΑΣΤΕ, ΗΜΟΥΝ, ΗΣΑΣΤΑΝ, ΗΣΑΣΤΕ, ΗΣΟΥΝ, ΗΤΑΝ, ΗΤΑΝΕ, ΗΤΟΙ, ΗΤΤΟΝ, ΘΑ, Ι, ΙΔΙΑ, ΙΔΙΑΝ, ΙΔΙΑΣ, ΙΔΙΕΣ, ΙΔΙΟ, ΙΔΙΟΙ, ΙΔΙΟΝ, ΙΔΙΟΣ, ΙΔΙΟΥ, ΙΔΙΟΥΣ, ΙΔΙΩΝ, ΙΔΙΩΣ, ΙΙ, ΙΙΙ, ΙΣΑΜΕ, ΙΣΙΑ, ΙΣΩΣ, ΚΑΘΕ, ΚΑΘΕΜΙΑ, ΚΑΘΕΜΙΑΣ, ΚΑΘΕΝΑ, ΚΑΘΕΝΑΣ, ΚΑΘΕΝΟΣ, ΚΑΘΕΤΙ, ΚΑΘΟΛΟΥ, ΚΑΘΩΣ, ΚΑΙ, ΚΑΚΑ, ΚΑΚΩΣ, ΚΑΛΑ, ΚΑΛΩΣ, ΚΑΜΙΑ, ΚΑΜΙΑΝ, ΚΑΜΙΑΣ, ΚΑΜΠΟΣΑ, ΚΑΜΠΟΣΕΣ, ΚΑΜΠΟΣΗ, ΚΑΜΠΟΣΗΝ, ΚΑΜΠΟΣΗΣ, ΚΑΜΠΟΣΟ, ΚΑΜΠΟΣΟΙ, ΚΑΜΠΟΣΟΝ, ΚΑΜΠΟΣΟΣ, ΚΑΜΠΟΣΟΥ, ΚΑΜΠΟΣΟΥΣ, ΚΑΜΠΟΣΩΝ, ΚΑΝΕΙΣ, ΚΑΝΕΝ, ΚΑΝΕΝΑ, ΚΑΝΕΝΑΝ, ΚΑΝΕΝΑΣ, ΚΑΝΕΝΟΣ, ΚΑΠΟΙΑ, ΚΑΠΟΙΑΝ, ΚΑΠΟΙΑΣ, ΚΑΠΟΙΕΣ, ΚΑΠΟΙΟ, ΚΑΠΟΙΟΙ, ΚΑΠΟΙΟΝ, ΚΑΠΟΙΟΣ, ΚΑΠΟΙΟΥ, ΚΑΠΟΙΟΥΣ, ΚΑΠΟΙΩΝ, ΚΑΠΟΤΕ, ΚΑΠΟΥ, ΚΑΠΩΣ, ΚΑΤ, ΚΑΤΑ, ΚΑΤΙ, ΚΑΤΙΤΙ, ΚΑΤΟΠΙΝ, ΚΑΤΩ, ΚΙΟΛΑΣ, ΚΛΠ, ΚΟΝΤΑ, ΚΤΛ, ΚΥΡΙΩΣ, ΛΙΓΑΚΙ, ΛΙΓΟ, ΛΙΓΩΤΕΡΟ, ΛΟΓΩ, ΛΟΙΠΑ, ΛΟΙΠΟΝ, ΜΑ, ΜΑΖΙ, ΜΑΚΑΡΙ, ΜΑΚΡΥΑ, ΜΑΛΙΣΤΑ, ΜΑΛΛΟΝ, ΜΑΣ, ΜΕ, ΜΕΘΑΥΡΙΟ, ΜΕΙΟΝ, ΜΕΛΕΙ, ΜΕΛΛΕΤΑΙ, ΜΕΜΙΑΣ, ΜΕΝ, ΜΕΡΙΚΑ, ΜΕΡΙΚΕΣ, ΜΕΡΙΚΟΙ, ΜΕΡΙΚΟΥΣ, ΜΕΡΙΚΩΝ, ΜΕΣΑ, ΜΕΤ, ΜΕΤΑ, ΜΕΤΑΞΥ, ΜΕΧΡΙ, ΜΗ, ΜΗΔΕ, ΜΗΝ, ΜΗΠΩΣ, ΜΗΤΕ, ΜΙΑ, ΜΙΑΝ, ΜΙΑΣ, ΜΟΛΙΣ, ΜΟΛΟΝΟΤΙ, ΜΟΝΑΧΑ, ΜΟΝΕΣ, ΜΟΝΗ, ΜΟΝΗΝ, ΜΟΝΗΣ, ΜΟΝΟ, ΜΟΝΟΙ, ΜΟΝΟΜΙΑΣ, ΜΟΝΟΣ, ΜΟΝΟΥ, ΜΟΝΟΥΣ, ΜΟΝΩΝ, ΜΟΥ, ΜΠΟΡΕΙ, ΜΠΟΡΟΥΝ, ΜΠΡΑΒΟ, ΜΠΡΟΣ, ΝΑ, ΝΑΙ, ΝΩΡΙΣ, ΞΑΝΑ, ΞΑΦΝΙΚΑ, Ο, ΟΙ, ΟΛΑ, ΟΛΕΣ, ΟΛΗ, ΟΛΗΝ, ΟΛΗΣ, ΟΛΟ, ΟΛΟΓΥΡΑ, ΟΛΟΙ, ΟΛΟΝ, ΟΛΟΝΕΝ, ΟΛΟΣ, ΟΛΟΤΕΛΑ, ΟΛΟΥ, ΟΛΟΥΣ, ΟΛΩΝ, ΟΛΩΣ, ΟΛΩΣΔΙΟΛΟΥ, ΟΜΩΣ, ΟΠΟΙΑ, ΟΠΟΙΑΔΗΠΟΤΕ, ΟΠΟΙΑΝ, ΟΠΟΙΑΝΔΗΠΟΤΕ, ΟΠΟΙΑΣ, ΟΠΟΙΑΣΔΗΠΟΤΕ, ΟΠΟΙΔΗΠΟΤΕ, ΟΠΟΙΕΣ, ΟΠΟΙΕΣΔΗΠΟΤΕ, ΟΠΟΙΟ, ΟΠΟΙΟΔΗΠΟΤΕ, ΟΠΟΙΟΙ, ΟΠΟΙΟΝ, ΟΠΟΙΟΝΔΗΠΟΤΕ, ΟΠΟΙΟΣ, ΟΠΟΙΟΣΔΗΠΟΤΕ, ΟΠΟΙΟΥ, ΟΠΟΙΟΥΔΗΠΟΤΕ, ΟΠΟΙΟΥΣ, ΟΠΟΙΟΥΣΔΗΠΟΤΕ, ΟΠΟΙΩΝ, ΟΠΟΙΩΝΔΗΠΟΤΕ, ΟΠΟΤΕ, ΟΠΟΤΕΔΗΠΟΤΕ, ΟΠΟΥ, ΟΠΟΥΔΗΠΟΤΕ, ΟΠΩΣ, ΟΡΙΣΜΕΝΑ, ΟΡΙΣΜΕΝΕΣ, ΟΡΙΣΜΕΝΩΝ, ΟΡΙΣΜΕΝΩΣ, ΟΣΑ, ΟΣΑΔΗΠΟΤΕ, ΟΣΕΣ, ΟΣΕΣΔΗΠΟΤΕ, ΟΣΗ, ΟΣΗΔΗΠΟΤΕ, ΟΣΗΝ, ΟΣΗΝΔΗΠΟΤΕ, ΟΣΗΣ, ΟΣΗΣΔΗΠΟΤΕ, ΟΣΟ, ΟΣΟΔΗΠΟΤΕ, ΟΣΟΙ, ΟΣΟΙΔΗΠΟΤΕ, ΟΣΟΝ, ΟΣΟΝΔΗΠΟΤΕ, ΟΣΟΣ, ΟΣΟΣΔΗΠΟΤΕ, ΟΣΟΥ, ΟΣΟΥΔΗΠΟΤΕ, ΟΣΟΥΣ, ΟΣΟΥΣΔΗΠΟΤΕ, ΟΣΩΝ, ΟΣΩΝΔΗΠΟΤΕ, ΟΤΑΝ, ΟΤΙ, ΟΤΙΔΗΠΟΤΕ, ΟΤΟΥ, ΟΥ, ΟΥΔΕ, ΟΥΤΕ, ΟΧΙ, ΠΑΛΙ, ΠΑΝΤΟΤΕ, ΠΑΝΤΟΥ, ΠΑΝΤΩΣ, ΠΑΡΑ, ΠΕΡΑ, ΠΕΡΙ, ΠΕΡΙΠΟΥ, ΠΕΡΙΣΣΟΤΕΡΟ, ΠΕΡΣΙ, ΠΕΡΥΣΙ, ΠΙΑ, ΠΙΘΑΝΟΝ, ΠΙΟ, ΠΙΣΩ, ΠΛΑΙ, ΠΛΕΟΝ, ΠΛΗΝ, ΠΟΙΑ, ΠΟΙΑΝ, ΠΟΙΑΣ, ΠΟΙΕΣ, ΠΟΙΟ, ΠΟΙΟΙ, ΠΟΙΟΝ, ΠΟΙΟΣ, ΠΟΙΟΥ, ΠΟΙΟΥΣ, ΠΟΙΩΝ, ΠΟΛΥ, ΠΟΣΕΣ, ΠΟΣΗ, ΠΟΣΗΝ, ΠΟΣΗΣ, ΠΟΣΟΙ, ΠΟΣΟΣ, ΠΟΣΟΥΣ, ΠΟΤΕ, ΠΟΥ, ΠΟΥΘΕ, ΠΟΥΘΕΝΑ, ΠΡΕΠΕΙ, ΠΡΙΝ, ΠΡΟ, ΠΡΟΚΕΙΜΕΝΟΥ, ΠΡΟΚΕΙΤΑΙ, ΠΡΟΠΕΡΣΙ, ΠΡΟΣ, ΠΡΟΤΟΥ, ΠΡΟΧΘΕΣ, ΠΡΟΧΤΕΣ, ΠΡΩΤΥΤΕΡΑ, ΠΩΣ, ΣΑΝ, ΣΑΣ, ΣΕ, ΣΕΙΣ, ΣΗΜΕΡΑ, ΣΙΓΑ, ΣΟΥ, ΣΤΑ, ΣΤΗ, ΣΤΗΝ, ΣΤΗΣ, ΣΤΙΣ, ΣΤΟ, ΣΤΟΝ, ΣΤΟΥ, ΣΤΟΥΣ, ΣΤΩΝ, ΣΥΓΧΡΟΝΩΣ, ΣΥΝ, ΣΥΝΑΜΑ, ΣΥΝΕΠΩΣ, ΣΥΝΗΘΩΣ, ΣΥΧΝΑ, ΣΥΧΝΑΣ, ΣΥΧΝΕΣ, ΣΥΧΝΗ, ΣΥΧΝΗΝ, ΣΥΧΝΗΣ, ΣΥΧΝΟ, ΣΥΧΝΟΙ, ΣΥΧΝΟΝ, ΣΥΧΝΟΣ, ΣΥΧΝΟΥ, ΣΥΧΝΟΥ, ΣΥΧΝΟΥΣ, ΣΥΧΝΩΝ, ΣΥΧΝΩΣ, ΣΧΕΔΟΝ, ΣΩΣΤΑ, ΤΑ, ΤΑΔΕ, ΤΑΥΤΑ, ΤΑΥΤΕΣ, ΤΑΥΤΗ, ΤΑΥΤΗΝ, ΤΑΥΤΗΣ, ΤΑΥΤΟ,ΤΑΥΤΟΝ, ΤΑΥΤΟΣ, ΤΑΥΤΟΥ, ΤΑΥΤΩΝ, ΤΑΧΑ, ΤΑΧΑΤΕ, ΤΕΛΙΚΑ, ΤΕΛΙΚΩΣ, ΤΕΣ, ΤΕΤΟΙΑ, ΤΕΤΟΙΑΝ, ΤΕΤΟΙΑΣ, ΤΕΤΟΙΕΣ, ΤΕΤΟΙΟ, ΤΕΤΟΙΟΙ, ΤΕΤΟΙΟΝ, ΤΕΤΟΙΟΣ, ΤΕΤΟΙΟΥ, ΤΕΤΟΙΟΥΣ, ΤΕΤΟΙΩΝ, ΤΗ, ΤΗΝ, ΤΗΣ, ΤΙ, ΤΙΠΟΤΑ, ΤΙΠΟΤΕ, ΤΙΣ, ΤΟ, ΤΟΙ, ΤΟΝ, ΤΟΣ, ΤΟΣΑ, ΤΟΣΕΣ, ΤΟΣΗ, ΤΟΣΗΝ, ΤΟΣΗΣ, ΤΟΣΟ, ΤΟΣΟΙ, ΤΟΣΟΝ, ΤΟΣΟΣ, ΤΟΣΟΥ, ΤΟΣΟΥΣ, ΤΟΣΩΝ, ΤΟΤΕ, ΤΟΥ, ΤΟΥΛΑΧΙΣΤΟ, ΤΟΥΛΑΧΙΣΤΟΝ, ΤΟΥΣ, ΤΟΥΤΑ, ΤΟΥΤΕΣ, ΤΟΥΤΗ, ΤΟΥΤΗΝ, ΤΟΥΤΗΣ, ΤΟΥΤΟ, ΤΟΥΤΟΙ, ΤΟΥΤΟΙΣ, ΤΟΥΤΟΝ, ΤΟΥΤΟΣ, ΤΟΥΤΟΥ, ΤΟΥΤΟΥΣ, ΤΟΥΤΩΝ, ΤΥΧΟΝ, ΤΩΝ, ΤΩΡΑ, ΥΠ, ΥΠΕΡ, ΥΠΟ, ΥΠΟΨΗ, ΥΠΟΨΙΝ, ΥΣΤΕΡΑ, ΦΕΤΟΣ, ΧΑΜΗΛΑ, ΧΘΕΣ, ΧΤΕΣ, ΧΩΡΙΣ, ΧΩΡΙΣΤΑ, ΨΗΛΑ, Ω, ΩΡΑΙΑ, ΩΣ, ΩΣΑΝ, ΩΣΟΤΟΥ, ΩΣΠΟΥ, ΩΣΤΕ, ΩΣΤΟΣΟ, ΩΧ

Lemmatization

ΕΠΛ660

- Lemmatization refers to doing things properly with the use of a vocabulary and morphological analysis of words.
- Reduce inflectional/variant forms to base form → lemma
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
  - *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization implies doing "proper" reduction to dictionary headword form

Stemming (στελέχωση κειμένου)

ΕΠΛ660

- Stemming refers to a heuristic process that reduces terms to their "roots" before indexing
- "Stemming" suggest crude affix chopping
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm

ΕΠΛ660

- Commonest algorithm for stemming English
  - Results suggest at least as good as other stemming options
- 5 phases of word reductions
  - Phases applied sequentially
  - Each phase consists of a set of commands
  - There are various conventions to select rules consists of a set of commands
    - Sample convention: S*electing the rule from each rule group that applies to the longest suffix.*

Typical rules in Porter

- *sses* → *ss*
- *ies* → *i*
- *ational* → *ate*
- *tional* → *tion*

Weight of word sensitive rules

- *(m>1) EMENT* →
  - *replacement* → *replac*
  - *cement* → *cement*

•Demo available at:
http://snowball.tartarus.org/demo.php
•Implementation (C, Java, …) available at:
http://www.tartarus.org/~martin/PorterStemmer/

Λάθη του Porter Stemmer

ΕΠΛ660

- Λάθος μετατροπή σε κοινή ρίζα:

  organization, organ → organ

  police, policy → polic

  arm, army → arm
- Μη-αναγνώριση της κοινής ρίζας:

  cylinder, cylindrical

  create, creation

  Europe, European

Other stemmers

ΕΠΛ660

- Other stemmers exist, e.g., Lovins stemmer
  http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
  - Single-pass, longest suffix removal (about 250 rules)
  - Motivated by linguistics as well as IR
- Rather than using a stemmer, you can use lemmatizer
  - Full morphological analysis – at most modest benefits for retrieval
- Do stemming and other normalizations help?
  - Often very mixed results: really help recall for some queries but harm precision on others

Does stemming improve effectiveness?

ΕΠΛ660

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.
- Examples: “operational AND research”, “operating AND system”, “operative AND dentistry”
- Porter Stemmer equivalence class (“oper”): operate operating operates operation operative operatives operational

Exercise

ΕΠΛ660

- Are the following statements true or false?

a. In a Boolean retrieval system, stemming never lowers precision.

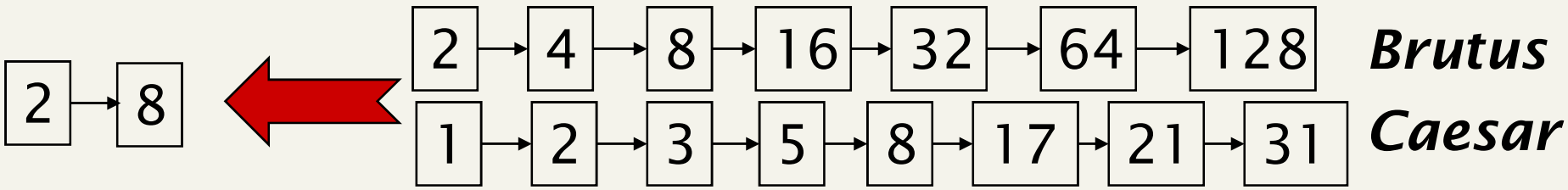
b. In a Boolean retrieval system, stemming never lowers recall.

c. Stemming increases the size of the lexicon.

d. Stemming should be invoked at indexing time but not while doing a query

Recall basic merge

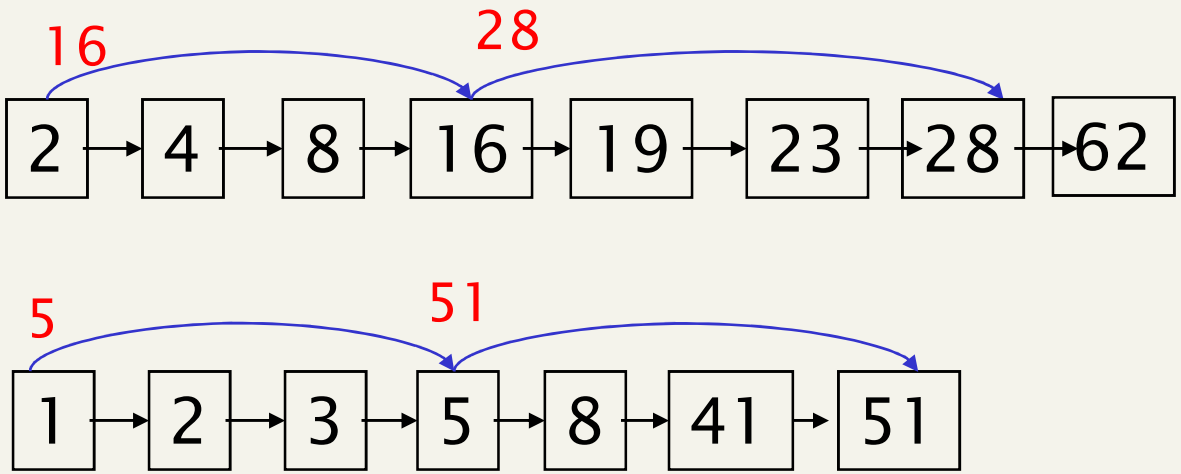
- Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are *m* and *n*, the merge takes O(*m*+*n*) operations.

Can we do better?
Yes, if index isn't changing too fast.

Augment postings with skip pointers (at indexing time)



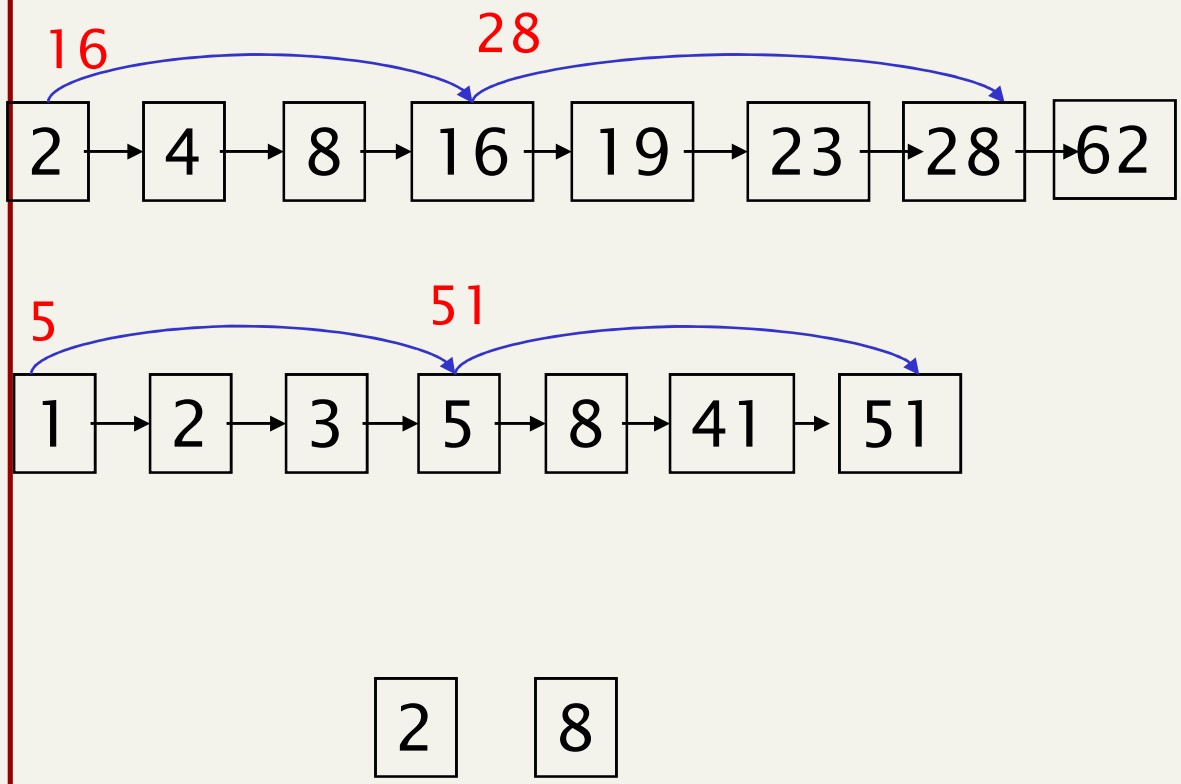
- Why?
- <u>To skip postings that will not figure in the search results.</u>
- How?
- Where do we place skip pointers?

Postings lists intersection with skip pointers

ΕΠΛ660

```
INTERSECTWITHSKIPS(p1, p2)
answer ← ⟨ ⟩
while p1 ≠ NIL and p2 ≠ NIL
do if docID(p1) = docID(p2)
      then ADD(answer, docID(p1))
           p1 ← next(p1)
           p2 ← next(p2)
      else if docID(p1) < docID(p2)
              then if hasSkip(p1) and (docID(skip(p1) ≤ docID(p2)))
                      then while hasSkip(p1) and (docID(skip(p1) ≤ docID(p2)))
                              do p1 ← skip(p1)
                      else p1 ← next(p1)
              else if hasSkip(p2) and (docID(skip(p2) ≤ docID(p1)))
                      then while hasSkip(p2) and (docID(skip(p2) ≤ docID(p1)))
                              do p2 ← skip(p2)
                      else p2 ← next(p2)
return answer
```

Query processing with skip pointers



docID(p1)=2, docID(p2)=1

2<1 False

hasSkip(p2) AND 5<2 False

p2← next(p2)

docID(p2)=2

2=2 TRUE

Answer=2

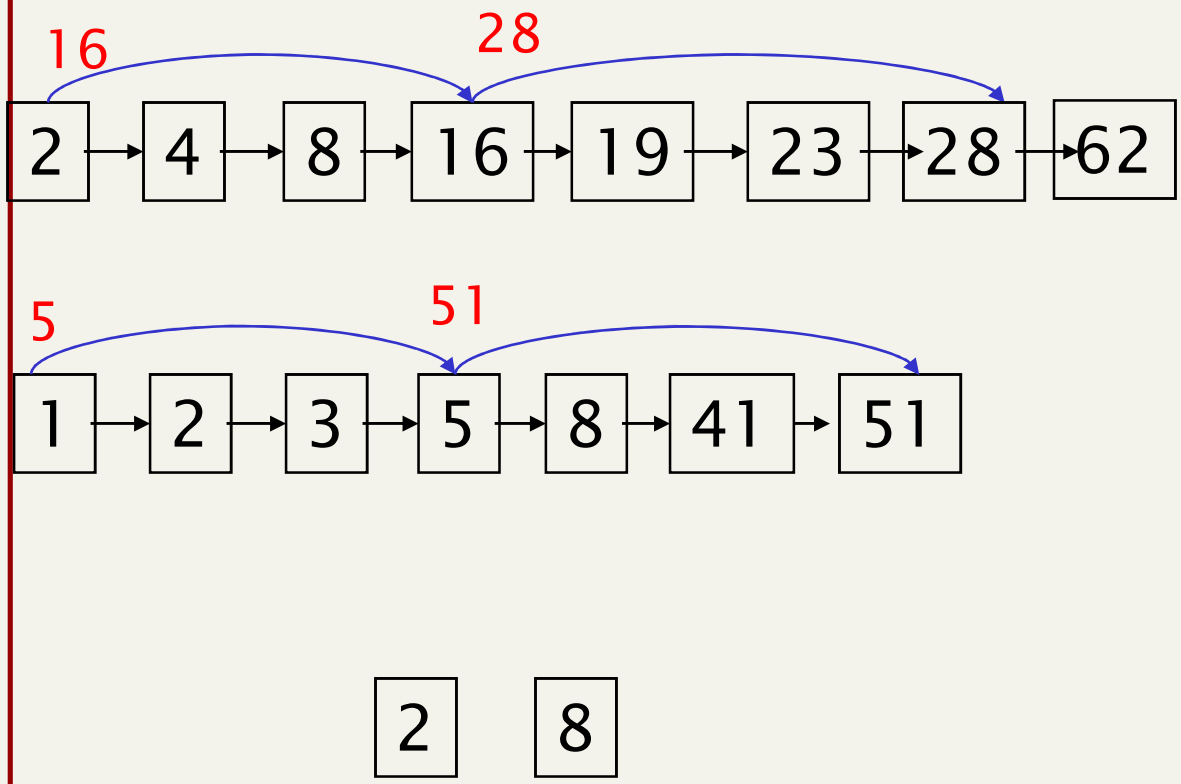
p1← next(p1) p2← next(p2)

docID(p1)=4, docID(p2)=3
4<3 False
hasSkip(p2) False
p2← next(p2), docID(p2)=5

4<5 TRUE

hasSkip(p1) False
p1← next(p1), docID(p1)=8

Query processing with skip pointers



8<5 False

hasSkip(p2) AND 51<8 False

p2← next(p2), docID(p2)=8

8=8 TRUE

Answer=2, 8

p1← next(p1) p2← next(p2)

docID(p1)=16, docID(p2)=41
16<41TRUE
hasSkip(p1) AND 28< 41 TRUE
p1← skip(p1) docID(p1)=28
hasSkip(p1) False

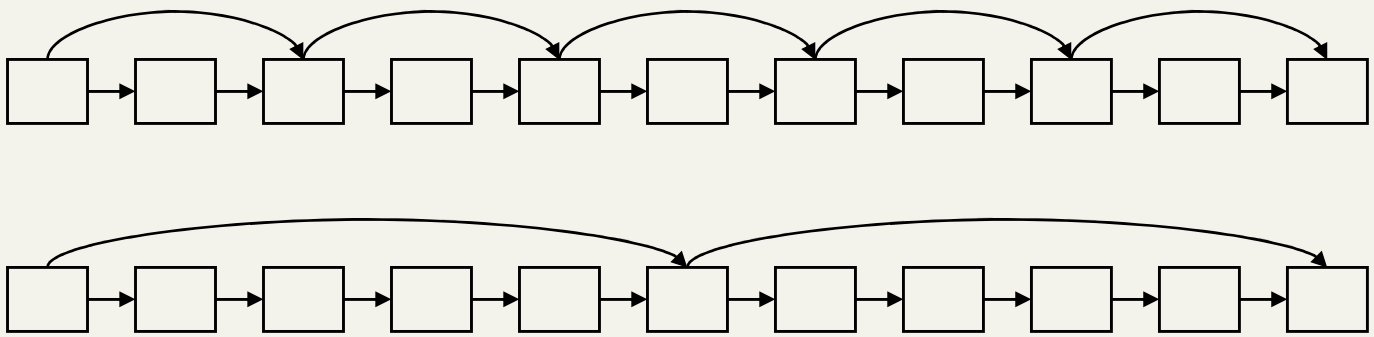
p1 ← next(p1), DocID(p1)=62

62<41 FALSE

hasSkip(p2) False
p2← next(p2), docID(p2)=51

Where do we place skips?

- Tradeoff:
  - More skips → shorter skip spans ⇒ more likely to skip. But lots of comparisons to skip pointers.
  - Fewer skips → few pointer comparison, but then long skip spans ⇒ few successful skips.



Placing skips

- Simple heuristic: for postings of length L, use $\sqrt{L}$ evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is relatively static; harder if L keeps changing because of updates.

- This definitely used to help; with modern hardware it may not
  - The cost of loading a bigger postings list outweighs the gain from quicker in memory merging

Phrase queries

ΕΠΛ660

- Want to answer queries such as "***stanford university***" – as a phrase
- Thus the sentence *"I went to university at Stanford"* is not a match.
  - The concept of phrase queries has proven easily understood by users; about 10% of Web queries are phrase queries
- No longer sufficient for postings lists

  <*term* : *docs*> entries

A first attempt: Biword indexes

ΕΠΛ660

- Index every consecutive pair of terms in the text as a phrase
- For example the text "Friends, Romans, Countrymen" would generate the biwords
  - ***friends romans***
  - ***romans countrymen***
- Each of these biwords is now a dictionary term
- Two-word phrase query-processing is now immediate.

Longer phrase queries

ΕΠΛ660

- Longer phrases are processed as follows:
- ***stanford university palo alto*** can be broken into the Boolean query on biwords:

stanford university *AND* ***university palo*** *AND* ***palo alto***

We cannot verify that the docs matching the above Boolean query do contain the phrase.

Can have false positives!

Extended biwords

ΕΠΛ660

- Parse the indexed text and perform part-of-speech-tagging (POST).
- Bucket the terms into (say) Nouns (N) and articles/prepositions (X).
- Now deem any string of terms of the form NX*N to be an <u>extended biword</u>.
  - Each such extended biword is now made a term in the dictionary.
- Example: ***catcher in the rye***

  N X X N
- Query processing: parse it into N's and X's
  - cost overruns on a power plant → "cost overruns" AND "overruns power" AND "power plant"

Issues for biword indexes

ΕΠΛ660

- False positives, as noted before
- Index blowup due to bigger dictionary

- The concept of a biword index can be extended to longer sequences of words → phrase index

Solution 2: Positional indexes

ΕΠΛ660

- Store, for each ***term***, entries of the form:

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

  doc1: position1, position2 … ;

  doc2: position1, position2 … ;

  etc.>

Positional index example

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

Which of docs 1,2,4,5 could contain "***to be or not to be***"?

Processing a phrase query

ΕΠΛ660

- Extract inverted index entries for each distinct term: ***to, be, or, not.***
- Merge their *doc:position* lists to enumerate all positions with "***to be or not to be***".
  - ***to***:
    - *2*:1,17,74,222,551; *4*:8,16,20,429,433; *7*:13,23,191; ...
  - ***be***:
    - *1*:17,19; *4*:17,21,291,430,434; *5*:14,19,101; ...

Proximity queries

ΕΠΛ660

- GATES /3 MICROSOFT

Here, /*k* means "within *k* words of".

- Clearly, positional indexes can be used for such queries; biword indexes cannot.

Positional index size

ΕΠΛ660

- Need an entry for each occurrence, not just once per document
- Index size depends on average document size
  - Average Web page has <1000 terms
  - SEC filings, books, even some epic poems … easily 100,000 terms
- Consider a term with frequency 0.1%

Why?

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

Rules of thumb

ΕΠΛ660

- A posting needs an entry for each occurrence of a term, not just once per document
- A positional index is 2–4 as large as a non-positional index
- Positional index size 35–50% of volume of original text
- Note: all of this holds for “English-like” languages

Combination schemes

ΕΠΛ660

- Biword indexes and positional indexes can be combined
  - For particular phrases (***"Michael Jackson", "Britney Spears"***) it is inefficient to keep on merging positional postings lists
    - Even more so for phrases like ***"The Who"***

Search engine features comparison

ΕΠΛ660

Search Engines	Boolean	Default	Proximity	Truncation	Fields	Limits	Stop	Sorting
Google Review	-, OR	and	Phrase	No (stems) word in phrase	intitle, inurl, link, site, more	Language, filetype, date, domain	Few, + searches	Relevance, site
Yahoo! Review	AND, OR, NOT, ( ), -	and	Phrase	No word in phrase	intitle, inurl, link, site, more	Language, file type, date, domain	No	Relevance, site
Ask Review	-, OR	and	Phrase	No	intitle, inurl, site	Language, site, date	Yes, + searches	Relevance, metasites
Live Search Review	AND, OR, NOT, ( ), -	and	Phrase	No	intitle, site, loc, url	Language, site	Varies, + searches	Relevance,site, sliders
Gigablast Review	AND, OR, AND NOT, ( ), +, -	and	Phrase	No	title, site, ip, more	Domain, type	Varies, + searches	Relevance
Exalead Review	AND, OR, NOT, ( ),-	and	Phrase, NEAR	Yes and stems	intitle, inurl, link, site	Language, file type, date, domain	Varies, + searches	Relevance, date

- Source: www.searchengineshowdown.com

Exercise

ΕΠΛ660

- The following pairs of words are stemmed to the same form by the Porter stemmer. Which pairs would you argue shouldn't be conflated. Give your reasoning.

a. abandon/abandonment

b. absorbency/absorbent

c. marketing/markets

d. university/universe

e. volume/volumes

Exercise

ΕΠΛ660

- Shown below is a portion of the positional index in the format term: doc1: position1, position2, . . . ; doc2: position1, position2, . . . ; etc.

angels: 2:36,174,252,651; 4:12,22,102,432; 7:17;

fools: 2:1,17,74,222; 4:8,78,108,458; 7:3,13,23,193;

fear: 2:87,704,722,901; 4:13,43,113,433; 7:18,328,528;

in: 2:3,37,76,444,851; 4:10,20,110,470,500; 7:5,15,25,195;

rush: 2:2,66,194,321,702; 4:9,69,149,429,569; 7:4,14,404;

to: 2:47,86,234,999; 4:14,24,774,944; 7:199,319,599,709;

tread: 2:57,94,333; 4:15,35,155; 7:20,320;

where: 2:67,124,393,1001; 4:11,41,101,421,431; 7:16,36,736;

Which document(s) if any meet each of the following queries, where each expression within quotes is a phrase query?

a. “fools rush in”

b. “fools rush in” AND “angels fear to tread”