

Lecturer: Prof. Paola Velardi

Lab Lecturer and project coordinator: Dr. Giovanni Stilo

WEB AND SOCIAL INFORMATION EXTRACTION

About this course

- <http://twiki.di.uniroma1.it/twiki/view/Estrinfo/WebHome>
- (Slides and course material)
- **PLEASE SIGN TO GOOGLE GROUP** (to receive self assessments and other course-related info)
- Course is organized as follows:
 - 2/3 “standard” lectures
 - 1/3 Lab:
 - design of an IR system with Lucene,
 - Using Twitter API
 - Implementi crawlers
 - Presentation of 2018 Project

Lectures

- Part I: web information retrieval
 - Architecture of an information retrieval system
 - Text processing, indexing
 - Ranking: vector space model, latent semantic indexing
 - Web information retrieval: browsing, scraping
 - Web information retrieval: link analysis (HITS, PageRank)
- Part II: social network analysis
 - Modeling a social network: local and global measures
 - Community detection
 - Mining social networks:
 - opinion mining,
 - temporal mining,
 - user profiling and Recommenders

PART I

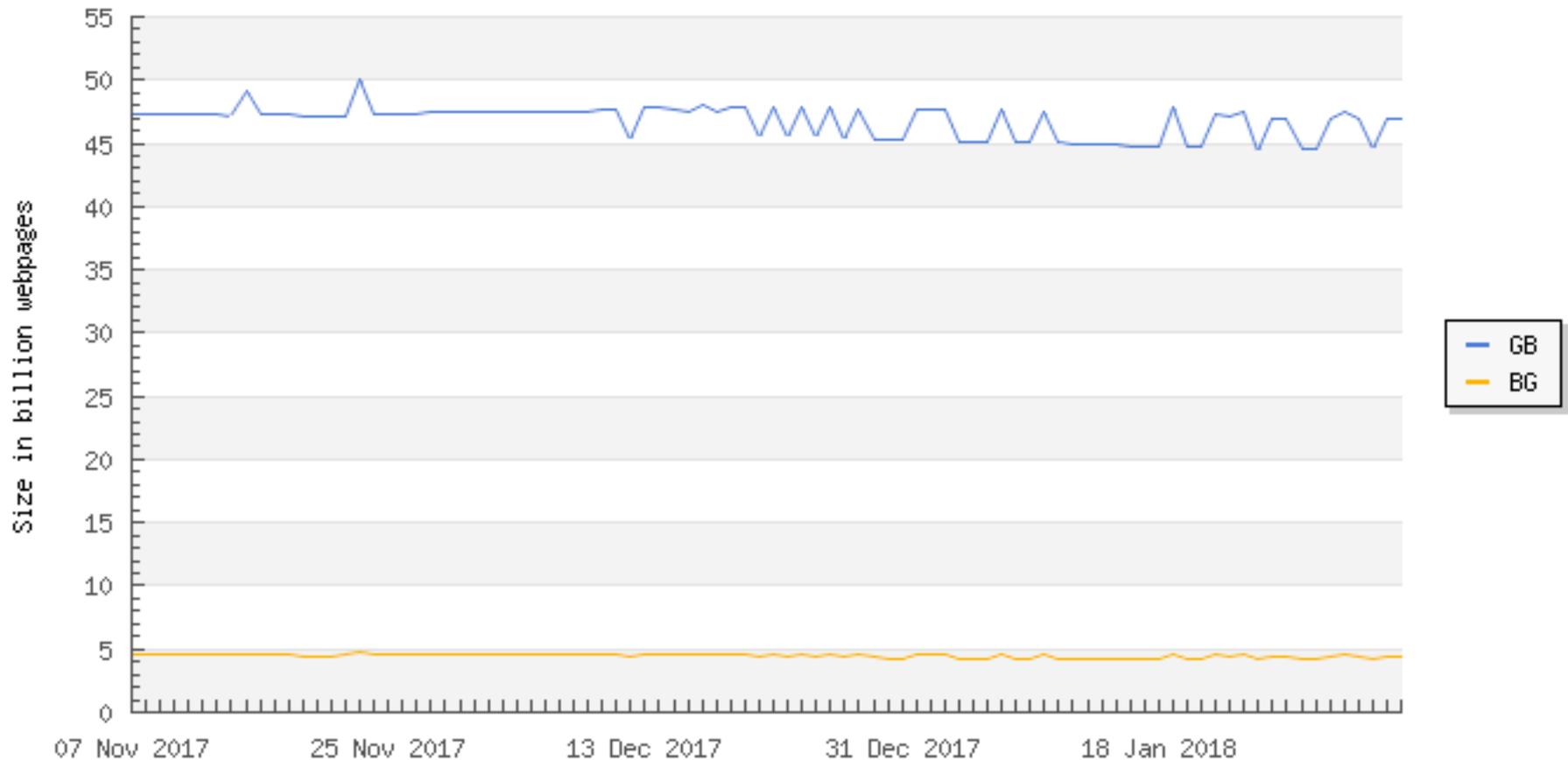
**INFORMATION RETRIEVAL:
DEFINITION AND ARCHITECTURE**

Information Retrieval is:

- Information Retrieval (IR) is **finding material** (usually documents) of an **unstructured** nature (usually text) that satisfies an **information need** from **large collections** (usually stored on computers).
- “*Usually*” text, **but** more and more: images, videos, data, services, audio..
- “*Usually*” unstructured (= no pre-defined model) **but**: Xml (and its dialects e.g. Voicexml..), RDF, html are “more structured” than txt or pdf
- “*Large*” collections: how large?? The Web! (The Indexed Web contains **at least 50 billion pages**)

Indexed pages (Google): 50 billion webpages! Bing is only around 5.

The size of the indexed World Wide Web
(Number of webpages)



IR vs. databases:

Structured vs unstructured data

- Structured data tends to refer to information in “tables”

Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

Typically allows numerical range and **exact match** (for text) queries, e.g.,
Salary < 60000 AND Manager = Smith.

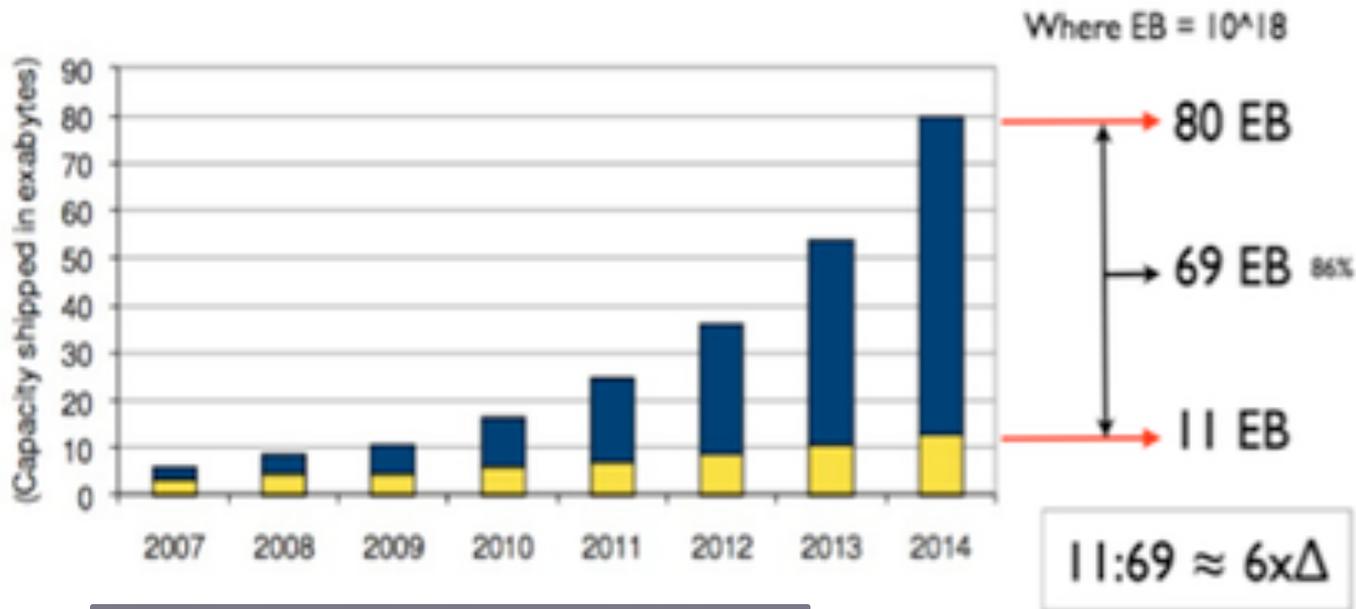
Unstructured data

- Typically refers to free-form text
- Allows:
 - Keyword queries including operators
 - (information \wedge (retrieval \vee extraction))
 - More sophisticated “concept” queries, e.g.,
 - find all web pages dealing with *drug abuse*

Semi-structured data

- In fact almost no data is “unstructured”
- E.g., this slide has distinctly identified zones such as the *Title* and *Bullets*
- This structure allows for “semi-structured” search such as
 - *Title* contains “data” AND *Bullets* contain “search”
 - Only **plain txt format** is truly unstructured (though even natural language does have a structure: a title, paragraphs, punctuation..)

Unstructured (text) vs. structured (database) data from 2007 to 2014 (exabyte)



Blue= unstructured
Yellow=structured

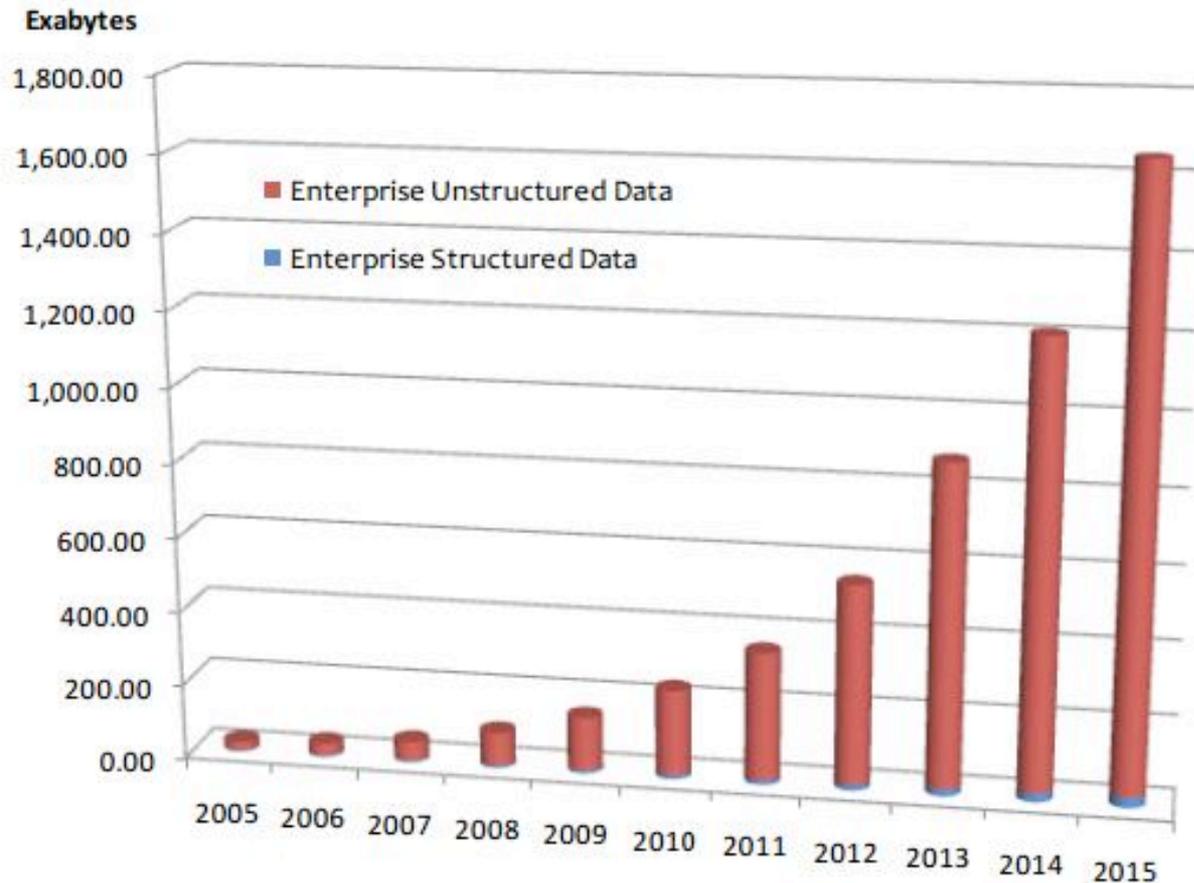
Google™

YAHOO!®

bing

Ask™
.com

Total (Unstructured) Enterprise Data Growth 2005-2015



The business is now unstructured data!

Not only text retrieval: other IR tasks

- **Clustering**: Given a set of docs, group them into clusters based on their contents.
- **Classification**: Given a set of topics, plus a new doc d , decide which topic(s) d belongs to (eg spam-nospam).
- **Information Extraction**: Find all sentence “snippets” dealing with a given topic (e.g. *company merges*)
- **Question Answering**: deal with a wide range of question types including: facts, lists, definitions, How, Why, hypothetical, semantically constrained, and cross-lingual questions
- **Opinion Mining**: Analyse/summarize sentiment in a text (e.g. TripAdvisor) (Hot Topic!!)
- All the above, applied to **images, video, audio, social networks**

Terminology

Searching: Seeking for specific information within a body of information. The result of a search is a set of **hits** (e.g. the list of web pages matching a query).

Browsing: Unstructured exploration of a body of information (e.g. a web browser is a software to traverse and retrieve info on the WWW).

Crawling: Moving from one item to another following links, such as citations, references, etc.

Scraping: pulling specific content from web pages

Terminology (2)

- **Query:** A string of text, describing the information that the user is seeking. Each word of the query is called a **search term** or **keyword**.
- A query can be a single search term, a string of terms, a phrase in natural language, or a stylized expression using special symbols.
- **Full text searching:** Methods that compare the query with **every word in the text**, without distinguishing the function (meaning, part-of-speech, position) of the various words.
- **Fielded searching:** Methods that search on specific bibliographic or **structural fields**, such as author or heading.

Examples of Search Systems

Find file on a computer system (e.g., *Spotlight* for Mac).

Library catalog for searching bibliographic records about books and other objects (e.g., *Library of Congress catalog*).

Abstracting and indexing system to find research information about specific topics (e.g., *Medline* for medical information).

Web search service to find web pages (e.g., *Google*).

Find file

The image shows a Mac OS X desktop with a Spotlight search window open. The search term is "information retrieval". The Spotlight results are displayed in a list view, showing the folder "information retrieval" as the top result. A tooltip is visible over the folder icon, providing details: "information retrieval", "Cartella", "44,4 MB, 54 elementi", and "Ultima modifica 08/gen/2014 16:14:30". The Spotlight window also shows a list of files and folders under the heading "Il migliore". The files listed include "Web Information Retrieval program engli", "Web sites for Information retrieval.docx", "x il primo scritto estrinfo 2010", "obarchi.doc", "esercizio bayes", and "D1". The folders listed include "information retrieval". The presentation files listed include "1.intro.ppt", "Probabilistic_Information_Retrieval_Notes", "13cap-patricia.ppt — partia", "lec8-VSPM,LSlecc.ppt", "2IntroIR.ppt — WebInfoext", "2.IntroIR.ppt", "2IntroIR.ppt — Lucidi PC 2006-7", and "13cap-patricia.ppt — partia Folder". The PDF files listed include "chp%3A10.1007%2F978-3-642-40285-" and "Multimedia Information Retrieval -Slides".

Spotlight information retrieval

Mostra tutto nel Finder

Il migliore **information retrieval**

Documenti

- Web Information Retrieval program engli
- Web Information Retrieval program engli
- Web sites for Information retrieval.docx
- x il primo scritto estrinfo 2010
- obarchi.doc
- esercizio bayes
- D1

Cartelle

- information retrieval

Presentazioni

- 1.intro.ppt
- Probabilistic_Information_Retrieval_Notes
- 13cap-patricia.ppt — partia
- lec8-VSPM,LSlecc.ppt
- 2IntroIR.ppt — WebInfoext
- 2.IntroIR.ppt
- 2IntroIR.ppt — Lucidi PC 2006-7
- 13cap-patricia.ppt — partia Folder

Documenti PDF

- chp%3A10.1007%2F978-3-642-40285-
- Multimedia Information Retrieval -Slides

Nome: **information retrieval**

Cartella

44,4 MB, 54 elementi

Ultima modifica 08/gen/2014 16:14:30

Library Catalogue

LIBRARY OF CONGRESS
ONLINE CATALOG

Print Subscribe Share/Save

LC Online Catalog Quick Search

Search

[Browse](#)

[Advanced Search](#)

[Keyword Search](#)



- [LC Online Catalog Home](#)
- [About the Catalog](#)
- [Frequently Asked Questions](#)
- [Search/Browse Help](#)
- [Print/Save/Email Help](#)

Search

- [Browse](#)
- [Advanced Search](#)
- [Keyword Search](#)

Your Account

- [Account Info](#)
- [Account Help](#)

Search/Browse Help - Browse

[Browse](#) | [Advanced Search](#) | [Keyword Search](#) | [Search Results \(Titles List\)](#) | [Search Results \(Headings Browse List\)](#) | [Record Display and Availability](#) | [Search Limits](#) | [Search History](#) | [Search Examples](#) | [Index Descriptions](#)

[About Browse](#) | [Browse Tips](#) | [Browse Results](#) | [Punctuation, Diacritics, Special Characters, Non-Roman Characters](#) | [Adding Limits](#)

About Browse

Browse provides a single search box for finding words and phrases for titles, authors/creators, subjects, call numbers, and standard numbers in an ordered list. *Shortcut URL:* <https://catalog.loc.gov/browse>

Browse search options fall into the following categories:

- Searches against controlled lists of authors/creators, subjects, names/titles, and uniform titles used by the Library of Congress -- along with cross-references. These controlled terms are referred to as "authorized headings." The LC Online Catalog supports two types of headings searches: you can either start with the left-most word of the heading (the "beginning with" options) or look for any word in the heading (the "containing" options).
- Searches for [LC classification number or shelving number](#), starting with the left-most part of the number.
- Searches for titles, starting with the left-most word of the title (removing initial articles).
- Searches for [LCCN](#) (Library of Congress Control Numbers), [ISSN](#), and [ISBN](#), starting at the beginning of the number.

Browse

Titles, Authors/Creators, Subjects, Call Numbers, Standard Numbers

Browse

TITLES beginning with (omit initial article)
TITLES beginning with (omit initial article)
AUTHORS/CREATORS beginning with (enter last name first)
AUTHORS/CREATORS containing
SUBJECTS beginning with
SUBJECTS containing
CALL NUMBERS (LC Class No.)
CALL NUMBERS (Other Shelving No.)
STANDARD NUMBERS (LCCN-ISBN-ISSN)
NAMES/TITLES beginning with
NAMES/TITLES containing
SERIES/UNIFORM TITLES containing

Clear

Search

Abstracting & Indexing

NCBI Resources How to Sign in to NCBI

PubMed Clinical Queries

Results of searches on this page are limited to specific clinical research areas. For comprehensive searches, use [PubMed](#) directly.

Clinical Study Categories

This column displays citations filtered to a specific clinical study category and scope. These search filters were developed by [Haynes RB et al.](#) See more [filter information](#).

Systematic Reviews

This column displays citations for systematic reviews, meta-analyses, reviews of clinical trials, evidence-based medicine, consensus development conferences, and guidelines. See [filter information](#) or additional [related sources](#).

Medical Genetics

This column displays citations pertaining to topics in medical genetics. See more [filter information](#).

You are here: [NCBI](#) > [Literature](#) > [PubMed](#) Write to the Help Desk

GETTING STARTED	RESOURCES	POPULAR	FEATURED	NCBI INFORMATION
NCBI Education	Chemicals & Bioassays	PubMed	Genetic Testing Registry	About NCBI
NCBI Help Manual	Data & Software	Bookshelf	PubMed Health	Research at NCBI
NCBI Handbook	DNA & RNA	PubMed Central	GenBank	NCBI News
Training & Tutorials	Domains & Structures	PubMed Health	Reference Sequences	NCBI FTP Site
Submit Data	Genes & Expression	BLAST	Gene Expression Omnibus	NCBI on Facebook
	Genetics & Medicine	Nucleotide	Map Viewer	NCBI on Twitter
	Genomes & Maps	Genome	Human Genome	NCBI on YouTube
	Homology	SNP	Mouse Genome	
	Literature	Gene	Influenza Virus	
	Proteins	Protein	Primer-BLAST	
	Sequence Analysis	PubChem	Sequence Read Archive	
	Taxonomy			
	Variation			

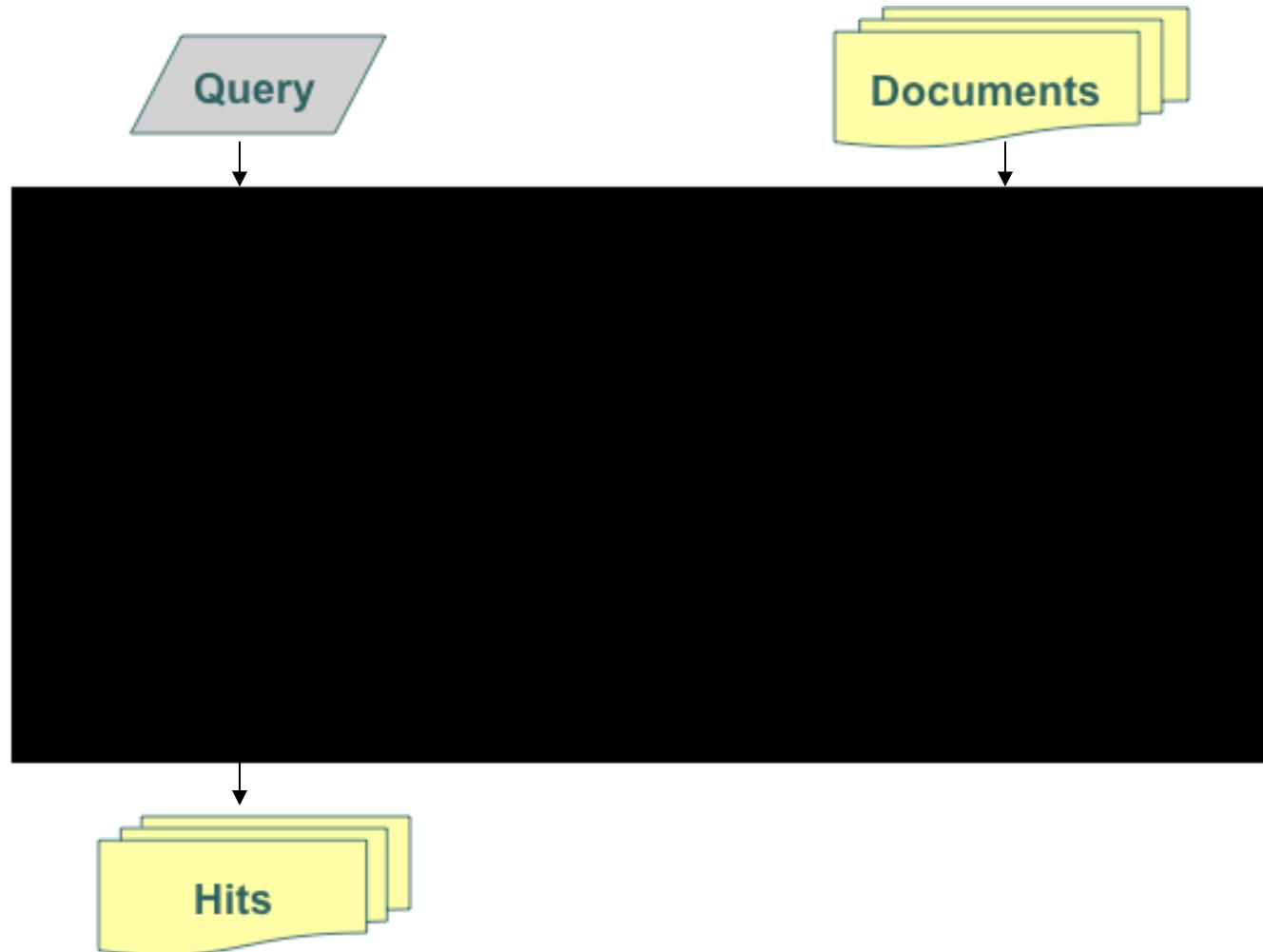
National Center for Biotechnology Information, U.S. National Library of Medicine
8600 Rockville Pike, Bethesda MD, 20894 USA
[Policies and Guidelines](#) | [Contact](#)

Web Search

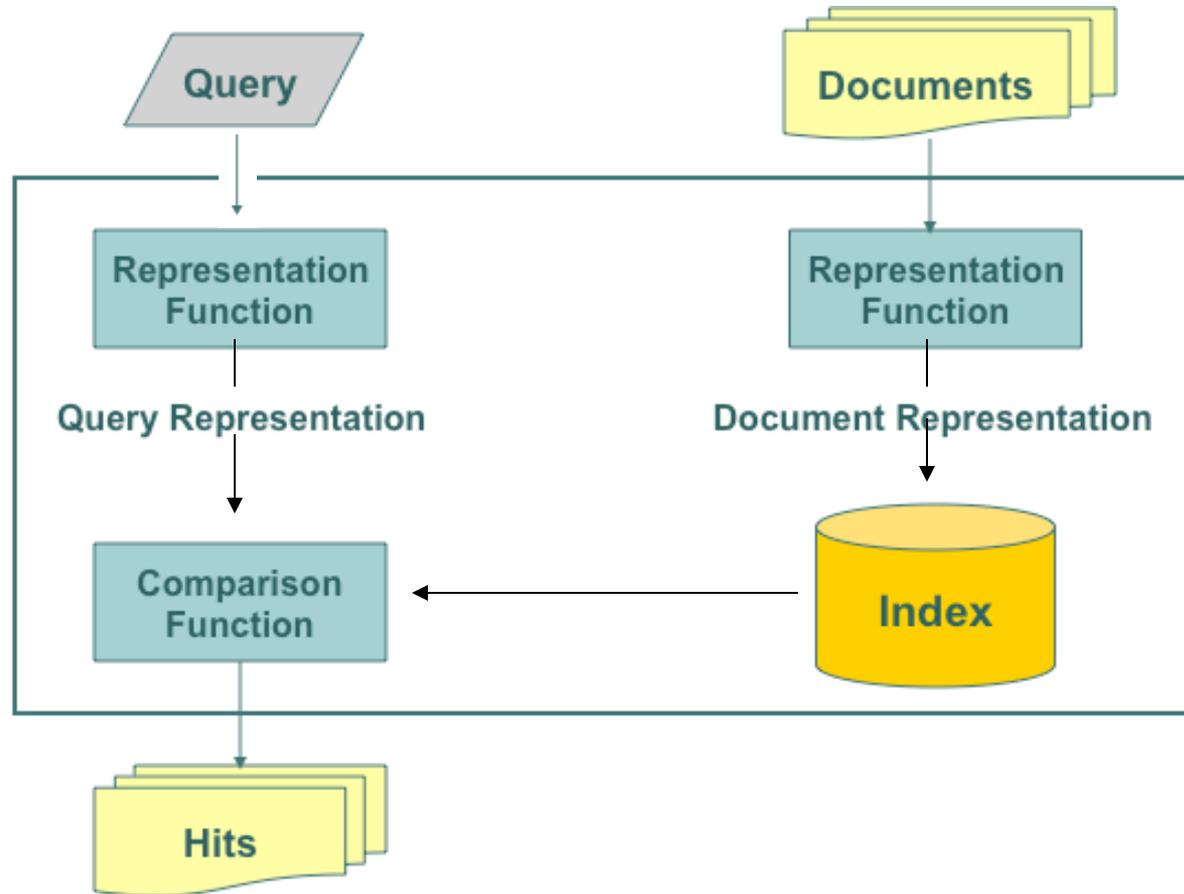


ARCHITECTURE OF AN IR SYSTEM

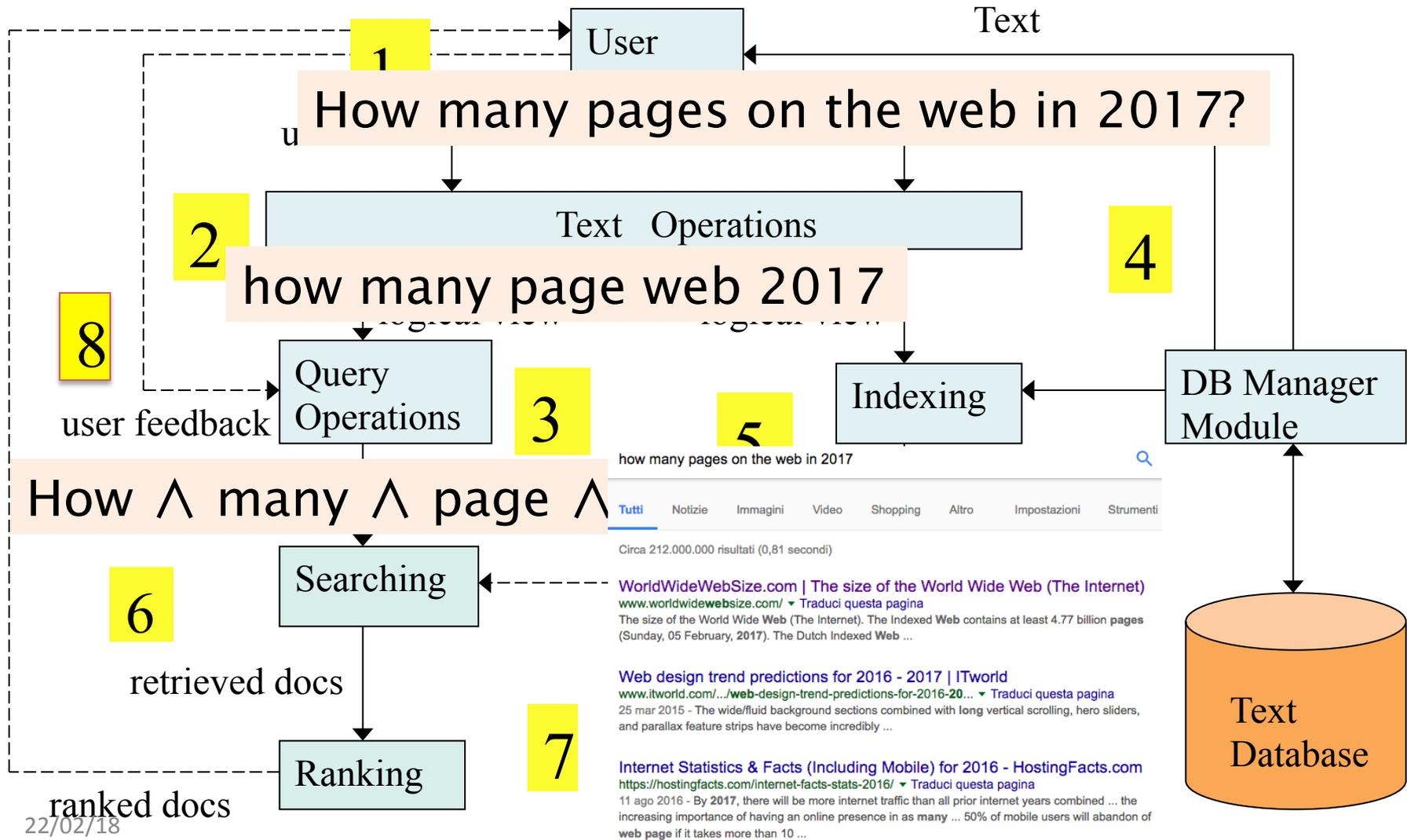
The IR Black Box



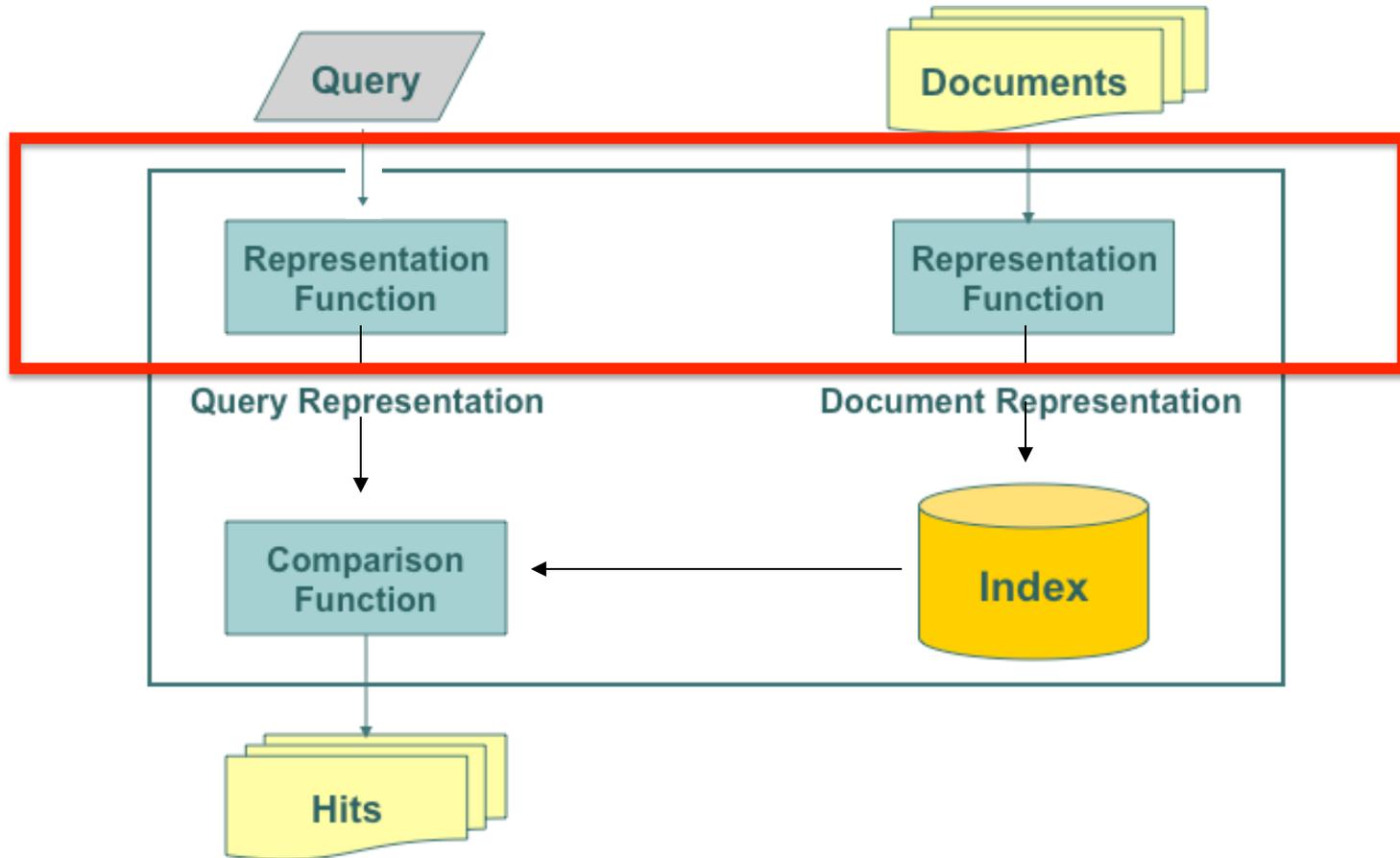
Inside The IR Black Box



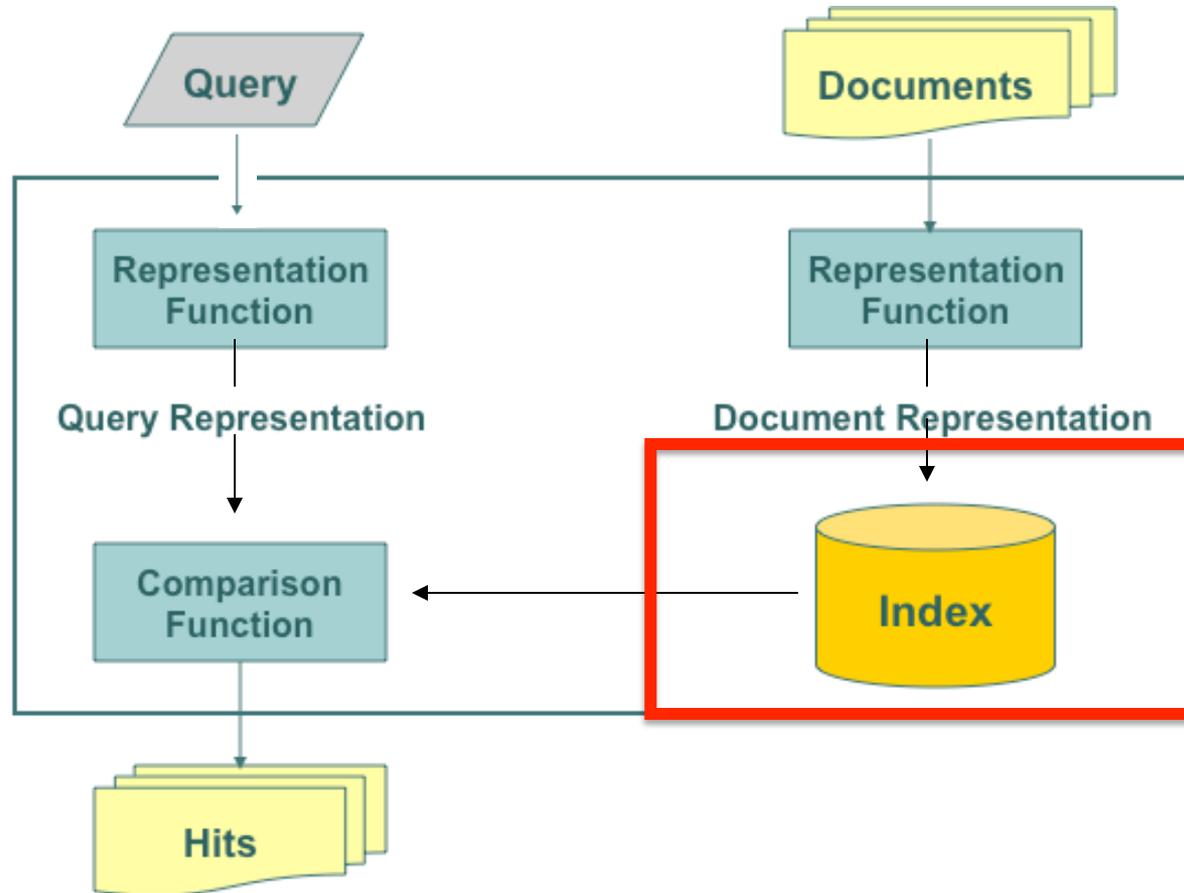
More in detail (representation, indexing, comparison, ranking)



Inside The IR Black Box



Inside The IR Black Box



Indexing:

a data structure that improves the speed of word retrieval

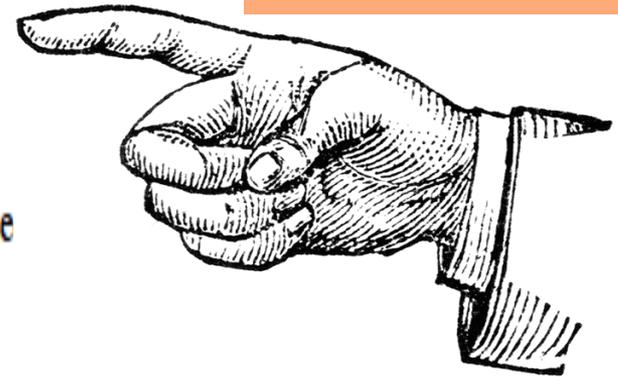
The corner of my mouth turned up in a wistful half-smile. “I used to think of you that way, you know. Like the sun. My personal sun. You balanced out the clouds nicely for me.”

He sighed. “The clouds I can handle. But I can’t fight with an eclipse.”

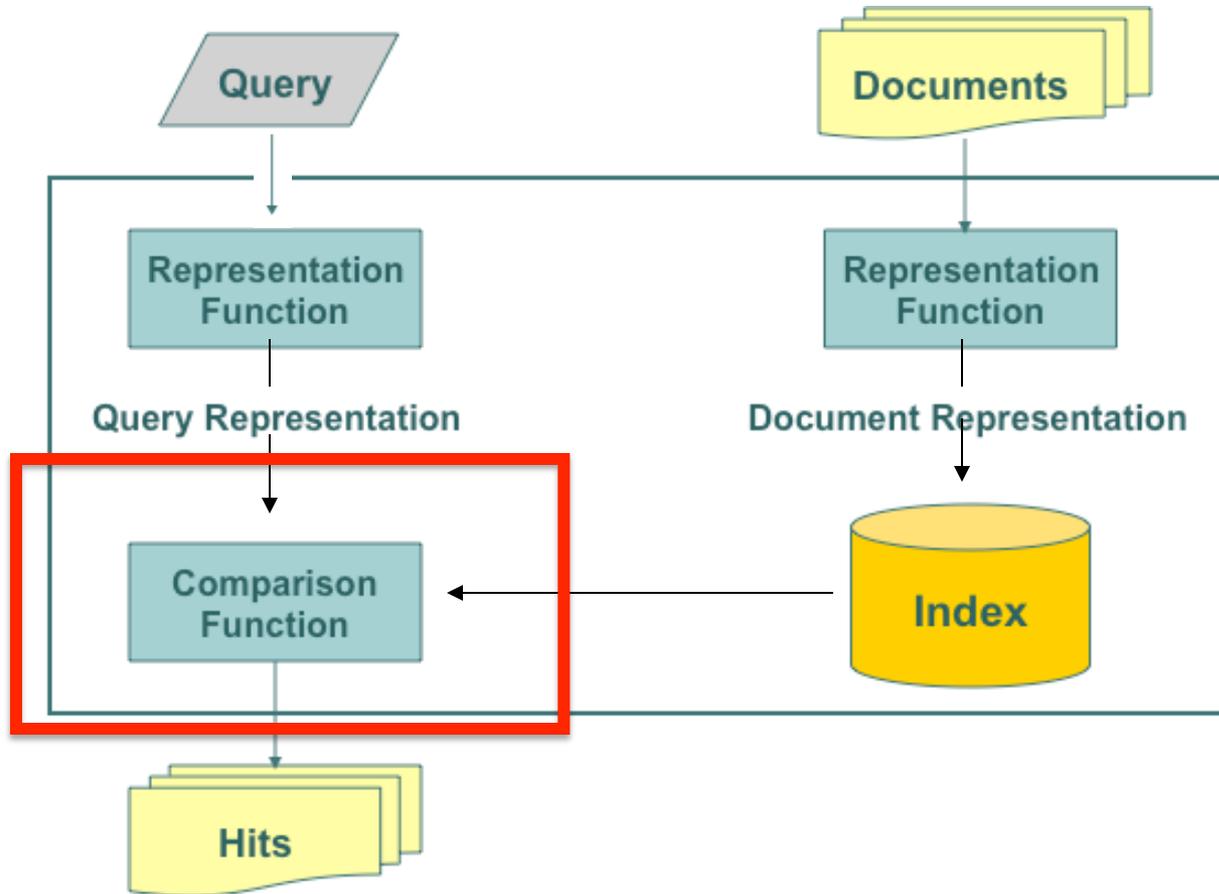
I touched his face, laying my hand against his cheek. He exhaled at my touch and close very quiet. For a minute I could hear the beating of his heart, slow and even.

“Tell me the worst part for you,” he whispered.

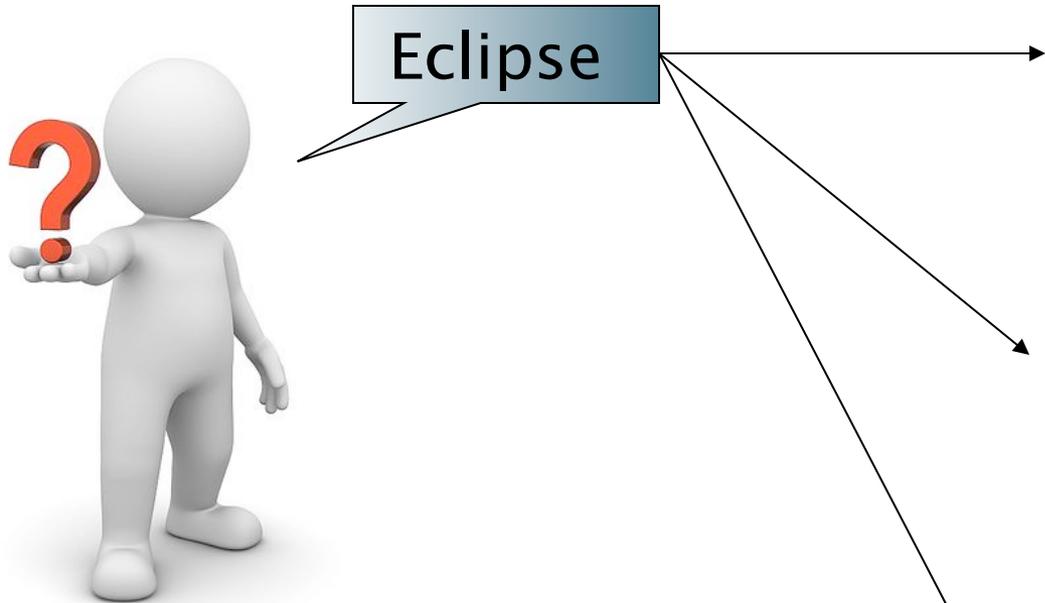
Points at words in texts



Inside The IR Black Box



Sorting & Ranking: how well a retrieved document matches the user's needs?



Solar and Lunar Eclipses of 2010			
Annular Solar Eclipse of 2010 Jan 15	Partial Lunar Eclipse of 2010 Jun 26	Total Solar Eclipse of 2010 Jul 11	Total Lunar Eclipse of 2010 Dec 21
Annular Saros 141 2010 Jan 15 07:08 TD Alt. = 66° Gam. = 0.4002 Dur. = 11m05s	Partial Saros 120 2010 Jun 26 11:40 TD P.Mag. = 160m U.Mag. = 0.5369 Gam. = -0.7931 P.Mag. = 1.9773	Total Saros 146 2010 Jul 11 19:35 TD Gam. = -0.6797 Alt. = 47° Dur. = 05m20s	Total Saros 125 2010 Dec 21 08:18 TD Tot. = 72m P.Mag. = 209m U.Mag. = 1.2561 Gam. = -0.2814 P.Mag. = 2.2857

Sorting & ranking

When a **user** submits a **query** to a **search system**, the system returns a set of **hits**. With a large collection of documents, the set of hits maybe very large.

The value to the user depends on the order in which the hits are presented.

Three main methods:

- **Sorting** the hits, e.g., by date (more recent are better.. Is this true?)
- **Ranking** the hits by **similarity** between query and document
- **Ranking** the hits by the **importance** of the documents

More details on

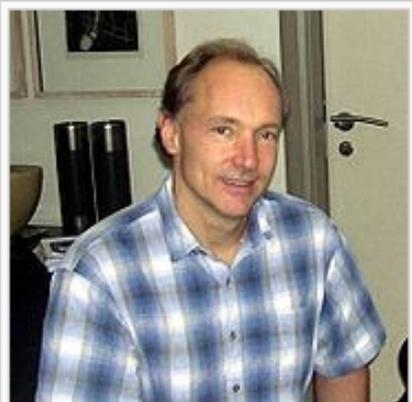
- Representation
- Indexing
- Ranking

(next 4-6 lessons)

1. Document Representation

- Objective: given a document in whatever format (txt, html, pdf..) provide a formal, structured representation of the document (e.g. a vector whose attributes are words, or a graph, or..)
- **Several steps from document downloading to the final selected representation**
- The most common representation model is “bag of words”

The bag-of-words model



Tim Berners-Lee on November 18, 2005.

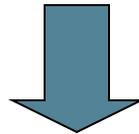
Background and early career

[edit]

Sir Tim Berners-Lee's parents, both [mathematicians](#), were employed together on the team that built the [Ferranti Mark 1](#), one of the earliest computers. They taught their son to use mathematics everywhere, even at the dinner table. Berners-Lee attended Sheen Mount Primary School, before moving on to study his O-Levels and A-Levels at [Emanuel School](#) in [Battersea, London](#) where a computer centre is dedicated in his name.

He is an [alumnus](#) of [The Queen's College, Oxford](#). While at Queen's, Berners-Lee built a computer with a [soldering iron](#), [TTL gates](#), an [M6800](#) processor and an old television. During his time at university, he was caught hacking with a friend and was subsequently banned from using the university's computer. He graduated in 1976 with a [degree](#) in [physics](#).

He met his first wife Jane while at [Oxford University](#) and they married soon after they started work in [Poole, Dorset](#). After graduation, Berners-Lee was employed at [Blessy Controls Limited](#) in Poole as a



$d_i = (\dots, \dots, \dots \text{after}, \dots \text{attend}, \dots \text{both}, \dots \text{build}, \dots \text{before}, \dots \text{center}, \dots \text{college}, \dots \text{computer}, \dots \text{dinner}, \dots \dots \dots \text{university}, \dots \text{work})$

WORD ORDER DOES NOT MATTER!!!

Bag of Words Model

- This is the most common way of representing documents in information retrieval
- Variants of this model include (more details later):
 - **How to weight a word** within a document (boolean, tf*idf, etc.)
 - Boolean: 1 is the word i is in doc j , 0 else
 - Tf*idf and others: the weight is a function of the word **frequency** in the document, and of the frequency of documents with that word
 - **What is a “word”**:
 - single, inflected word (“going”),
 - lemmatised word (going, go, gone → go)
 - Multi-word, proper nouns, numbers, dates (“board of directors”, “John Wyne”, “April, 2010”)
 - Meaning: (plan, project, design → PLAN#03)

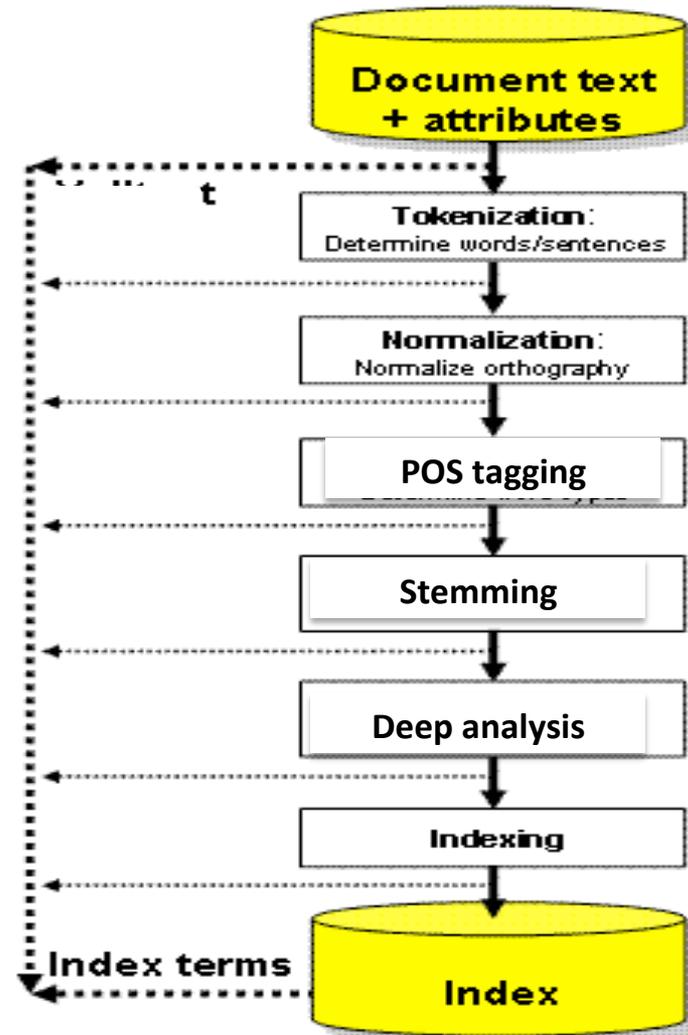
Bag of Words model is also used for images
("words" are now image features)



Phases in document processing (Several steps from document downloading to the final selected representation)

1. Document parsing
2. Tokenization
3. Stopwords/
Normalization
4. POS Tagging
5. Stemming
6. Deep NL Analysis
7. Indexing

Note that intermediate steps can be skipped



1. Document Parsing

Document parsing implies scanning a document and transforming it into a “bag of words” but: which words?

- We need to deal with format and language of each document.
- What format is it in?
- pdf/word/excel/html?
- What language is it in?
- What character set is in use (latin, greek, chinese..)?

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

But these tasks are often done heuristically ...

(Doc parsing) Complications: Format/ language

- 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
 - ex : French email with a German pdf attachment.
- What is a “unit” document?
 - A single file? A zipped group of files?
 - An email/message?
 - An email with 5 attachments?
 - A single web page of a full web site?

2. Tokenization

- Input: “*Friends, Romans and Countrymen*”
- Output: Tokens
 - *Friends*
 - *Romans*
 - *Countrymen*
- A **token** is an instance of a sequence of characters
- Each such token is now a candidate for an index entry, after further processing
 - Described later
- But which are valid tokens to emit?

2. Tokenization (cont'd)

- 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***
 - ***lowercase, lower-case, lower case ?***
 - ***San Francisco***: one token or two?
 - How do you decide it is one token?
 - ***cheap San Francisco-Los Angeles fares***

2. Tokenization : Numbers

- ***3/12/91***
- ***Mar. 12, 1991***
- ***12/3/91***
- ***55 B.C.***
- ***B-52***
- ***(800) 234-2333***
- ***1Z9999W99845399981 (package tracking numbers)***
 - Often have embedded spaces (ex. IBAN/SWIFT)
 - Older IR systems may not index numbers
 - Since their presence greatly expands the size of the vocabulary
 - IR systems **often index separately document “meta-data”**
 - Creation date, format, etc.

2. Tokenization: language issues

- French & Italian apostrophes
 - *L'ensemble* → one token or two?
 - *L ? L' ? Le ?*
 - We may want *l'ensemble* to match with *un ensemble*
- German noun compounds are not segmented
 - *Lebensversicherungsgesellschaftsangestellter*
 - 'life insurance company employee'
 - German retrieval systems benefit greatly from a **compound splitter** module

2. 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

2. 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.'*
 - Bidirectionality is not a problem if text is coded in **Unicode**.

UNICODE standard

Unicode

From Wikipedia, the free encyclopedia

For the 1889 Universal Telegraphic Phrase-book, see [Commercial code \(communications\)](#).

Unicode is a [computing industry standard](#) for the consistent [encoding](#), representation, and handling of [text](#) expressed in most of the world's [writing systems](#). Developed in conjunction with the [Universal Character Set](#) standard and published as *The Unicode Standard*, the latest version of Unicode contains a repertoire of more than 110,000 [characters](#) covering 100 [scripts](#) and multiple symbol sets. The standard consists of a set of code charts for visual reference, an encoding method and set of standard [character encodings](#), a set of reference data [computer files](#), and a number of related items, such as character properties, rules for [normalization](#), decomposition, [collation](#), rendering, and [bidirectional](#) display order (for the correct display of text containing both right-to-left scripts, such as [Arabic](#) and [Hebrew](#), and left-to-right scripts).^[1] As of June 2014, the most recent version is *Unicode 7.0*. The standard is maintained by the [Unicode Consortium](#).

This article contains [special characters](#). Without proper [rendering support](#), you may see [question marks](#), [boxes](#), or other symbols.



3.1 Stop words

- With a stop list, you exclude from the dictionary entirely **the commonest words**. Intuition:
 - They have little semantic content: *the, a, and, to, be*
 - There are a lot of them: ~30% of postings for top 30 words
 - Stop word **elimination used to be standard in older IR systems**.
- But the trend is away from doing this:
 - Good compression techniques means the space for including stopwords in a system is very small- so removing them not a big deal
 - 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/books titles, etc.: “Let it be”, “To be or not to be”
 - “Relational” queries: “flights to London” vrs “flight from London”

Stop words on Google?

how many pages are there on the web in 2017



All News Images Shopping Videos More Settings Tools

About 356,000,000 results (0.92 seconds)

WorldWideWebSize.com | The size of the World Wide Web (The Internet)

www.worldwidewebsize.com/

The Indexed **Web** contains at least 4.56 billion **pages**. ... The Dutch Indexed **Web** contains at least 195.78 million **pages** (Monday, 06 February, 2017). ... The number of webpages found for these words are recorded; with their relative ...

Internet Statistics & Facts (Including Mobile) for 2016 - HostingFacts.com

<https://hostingfacts.com/internet-facts-stats-2016/>

Aug 11, 2016 - By 2017, there will be more internet traffic than all prior internet years combined ... **Web** Hosting/**Website** Statistics and Facts 2016 ... goes to show the increasing importance of having an online presence in as many places as possible. ... 50% of mobile users will abandon of **web page** if it takes more than 10 ...

The Best Search Engines of 2017 - Lifewire

<https://www.lifewire.com> > ... > Web & Search Tips & Strategies > Searching the Web

Jan 23, 2017 - There are so many search engines out there! ... Updated for 2017! ... relevant, and the largest single catalogue of **web pages** available today.

400 Facebook Statistics and Facts (February 2017)

expandedramblings.com/index.php/by-the-numbers-17-amazing-facebook-stats/

4 days ago - ... 2017. facebook statistics 2016 how many are on facebook Pingback: Five Tips to a Better Healthcare **Website** | MindStream Creative. Hi Craig, can you let me know how many **FB pages** are there for photography.

pages web i 2017

All Images News Videos Shopping More

About 304,000,000 results (0.70 seconds)

Create Your Own Web Page

Ad www.wix.com/create-own-web-page

Easily Create Your Own Professional Free & HTML, Design Yours Today!
Top industry hosting · Social media compatible · SEO wizard · Easy-to-add blog
Services: App Market, SEO Wizard, Mailboxes, Blog, Online Stores

Free Website Builder

Free Hosting

Customize Easily

Online Store Builder

Web design trends we can expect to see in 2017 - The Next Web

<https://thenextweb.com> > Design & Dev

Dec 22, 2016 - 2017 is sure to bring some amazing **website** designs, but if we look away from the entire point of a **web page**: the content.

Digital trends 2017: 106 pages of internet, mobile and ... - The Next Web

<https://thenextweb.com> > insights

Jan 24, 2017 - Teaming up with social media help tool Hootsuite, the agency has 238 countries to put together 106 **pages** of the latest stats and trends in the digital world, 2017 marks a major milestone in **web** usage as half of the world's population

18 web design trends for 2017 | Webflow Blog

<https://webflow.com/blog/18-web-design-trends-for-2017>

Dec 9, 2016 - Discover the **web** design trends that will define **website** and digital design for designers to stay on the same **page** — and that ...

3.2. Normalization to terms

- We need to “normalize” words in indexed text as well as query words into the same form
 - We want to match ***U.S.A.*** and ***USA***
- Result is terms: a **term** is a (normalized) word type, which is a **single entry** in our IR system dictionary
- We most commonly implicitly define **equivalence classes** of terms by, e.g.,
 - deleting periods to form a term
 - *U.S.A., USA* → *USA*
 - deleting hyphens to form a term
 - *anti-discriminatory, antidiscriminatory* → *antidiscriminatory*
 - Synonyms (this is rather more complex..)
 - car , automobile

3.2 Normalization: other languages

- Accents: e.g., French *résumé* vs. *resume*.
- Umlauts: e.g., German: *Tuebingen* vs. *Tübingen*
 - Should be equivalent
- 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**
 - Often best to normalize to a de-accented term
 - *Tuebingen, Tübingen, Tubingen* → *Tubingen*

3.2 Normalization: other languages

- Normalization of other strings like date forms
 - *7月30日 vs. 7/30*
 - *Japanese use of kana vs. Chinese characters*
- Tokenization and normalization may depend on the language and so is interweaved with language detection

Morgen will ich in MIT ...

Is this
German “mit”?

- Crucial: Need to “normalize” indexed text as well as query terms into the same form

Do we really want normalization?

- Normalization and language detection interact.
- *PETER WILL NICHT MIT.* → MIT = mit
- *He got his PhD from MIT.* → MIT ≠ mit

3.2 Case folding

- Reduce all letters to lower case
 - exception: upper case in mid-sentence
 - e.g., *General Motors*
 - *Fed* vs. *fed*
 - *MIT* vs. *mit*
 - Often **best to lower case everything**, since users will use lowercase regardless of ‘correct’ capitalization...
- This may cause different senses to be merged..
Often the most relevant is simply the most frequent on the WEB, rather than the most intuitive

Web

Immagini

Video

Maps

Notizie

Altro ▾

Strumenti di ricerca

Circa 2.080.000.000 risultati (0,30 secondi)

Cat | Prodotti e Servizi - Europe | Caterpillar

www.cat.com/it_IT.html ▾

Questo sito utilizza e installa dei "cookie" sul computer che contribuiscono a migliorarne la qualità. Per informazioni più dettagliate su questi cookie e ...

Caterpillar: Cat | global-selector

www.cat.com/ ▾ Traduci questa pagina

Manufacturer of construction and mining equipment, diesel and natural gas engines, industrial gas turbines, and a wide offering of related services.

Immagini relative a CAT

Segnala immagini non appropriate

[Altre immagini per CAT](#)

[Web](#)[Immagini](#)[Video](#)[Maps](#)[Notizie](#)[Altro ▾](#)[Strumenti di ricerca](#)

Circa 2.080.000.000 risultati (0,32 secondi)

Cat | Prodotti e Servizi - Europe | Caterpillar

www.cat.com/it_IT.html ▾

Questo sito utilizza e installa dei "cookie" sul computer che contribuiscono a migliorarne la qualità. Per informazioni più dettagliate su questi cookie e ...

Caterpillar: Cat | global-selector

www.cat.com/ ▾ [Traduci questa pagina](#)

Manufacturer of construction and mining equipment, diesel and natural gas engines, industrial gas turbines, and a wide offering of related services.

Immagini relative a C.A.T.

[Segnala immagini non appropriate](#)



[Altre immagini per C.A.T.](#)

Case folding on Google?

Web Immagini Video Maps Notizie Altro ▾ Strumenti di ricerca

Circa 2.080.000.000 risultati (0,30 secondi)

Cat | Prodotti e Servizi - Europe | Caterpillar

www.cat.com/it_IT.html ▾

Questo sito utilizza e installa dei "cookie" sul computer per migliorare la qualità. Per informazioni più dettagliate su come utilizzare i cookie, visitate la pagina [Cookie](#).

Caterpillar: Cat | global-selector

www.cat.com/ ▾ Traduci questa pagina

Manufacturer of construction and mining equipment, power generation equipment, industrial gas turbines, and a wide offering of related services.

Cat CAT C.A.T.
SAME RESULTS!!!

Immagini relative a CAT

Segnala immagini non appropriate



Altre immagini per CAT

3.2 Normalization: Synonyms

- Do we handle synonyms and homonyms?
 - E.g., by hand-constructed equivalence classes
 - *car = automobile* *color = colour*
 - We can rewrite to form **equivalence-class terms**
 - When the document contains *automobile*, index it under *car-automobile* (and vice-versa)
 - Or we can **expand a query**
 - When the query contains *automobile*, look under *car* as well
- What about spelling mistakes?
 - One approach is **Soundex**, a phonetic algorithm to encode *homophones* (=same sound) to the same representation so that they can be matched despite minor differences in spelling
 - Google → Googol

Synonyms on Google?

Finland's capital|

finland's capital
finland's capital **city**
finland capital **and currency**
finland's capital **crossword**

Finland / Capital



Helsinki

Plan a trip and points of interest

People also ask

- What country is Finland in?
- Is Helsinki in Europe?
- How many islands are there in Finland?
- Is Finland a European country?

Helsinki - Wikipedia

<https://en.wikipedia.org/wiki/Helsinki>
Helsinki (/helˈsɪŋki/; Finnish pronunciation: [ˈhelsiŋki] (listen); Swedish: **Helsingfo** largest city of Finland. It is in the region of Uusimaa, in southern Finland, on the sh
Finland.
Uusimaa · Suomenlinna · Helsinki Cathedral · Uspenski Cathedral

helsinki

helsinki
helsinki **weather**
helsinki **airport**
helsinki **time**

Learn more

Helsinki - Wikipedia

<https://en.wikipedia.org/wiki/Helsinki>
Helsinki is the capital and largest city of Finland. It is in the region of Uusimaa, in southern Finland, on the shore of the Gulf of Finland. Helsinki has a population ...
Uusimaa · Suomenlinna · Helsinki Cathedral · Uspenski Cathedral

Visit Helsinki : City of Helsinki's official website for tourism and travel ...

www.visithelsinki.fi/en
The Capital of Finland offers lots to see, do and experience for visitors of all ages. Here are just a few examples of the most popular attractions. Read more » ...

Top stories



Helsinki's mayoral race still missing one big name

Helsinki Times · 8 hours ago



Snøhetta to Design Helsinki Hotel Inspired by Shattered Ice

Interior Design · 1 day ago



Snøhetta's winning hotel design for Helsinki waterfront is inspired by broken ...

Inhabitat · 4 days ago

More for helsinki

Helsinki — VisitFinland.com

Spelling mistakes on Google?



Googel



Ricerca

Circa 14.980.000.000 risultati (0,34 secondi)

Tutto

Risultati visualizzati per [Google](#)

Cerca invece [Googel](#)

Immagini

Maps

[Google](#)

www.google.it/

Video

Versione italiana del popolare motore e directory. Utilizza inoltre i dati di ODP.

Notizie

Shopping

[Google Maps](#)

maps.google.it/

Più contenuti

Visualizza mappe e trova attività commerciali locali sul Web.

Hai visitato questa pagina 4 volte. Ultima visita: 26/02/12

4. Stemming/Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
 - *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 form (the **lemma**).
 - Relatively simple for English more complex for highly inflected languages (italian, german..)

4. Stemming

- Reduce 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 equal to compress

Porter's algorithm

- Commonest algorithm for stemming English
 - Results suggest it's at least as good as other stemming options
- Conventions + 5 phases of reductions
 - phases applied sequentially
 - each phase consists of a set of commands
 - sample convention: *out of multiple applying rules in a command, select the one that applies to the longest suffix.*
 - *E.g. caresses → caress rather than car*
-

Typical rules (commands) in Porter stemmer

- *sses* → *ss* → *caresses* → *caress*
- *ies* → *l* *ponies* → *poni*
- *SS* → *SS* *caress* → *caress*
- *es* → *cats* → *cat*

- Weight of word-sensitive rules:
- *(m>1) EMENT* →
 - *replacement* → *replac*
 - *cement* → *cement*
- *It means: root must be longer than 1 to apply the rule..*

Three stemmers: A comparison

Sample text: **Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation**

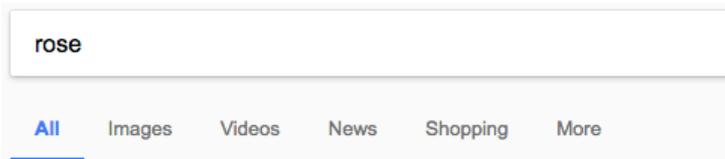
Porter's: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to pictur of express that is more biolog transpar and access to interpret

Lovins's: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Paice's : such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Stemming on Google?

■ rose



About 1,880,000,000 results (0.95 seconds)

Rose - Wikipedia

<https://en.wikipedia.org/wiki/Rose> ▼

A **rose** is a woody perennial flowering plant of the genus *Rosa*, in the family Rosaceae. There are over a hundred species and thousands ...

Rose - Free images on Pixabay

<https://pixabay.com/en/photos/rose/> ▼

Download free images about **Rose** from Pixabay's library of over 870000 public domain illustrations and vectors.

Home - The Rose

rosecafevenice.com/ ▼

The **Rose Cafe**-Restaurant is an iconic Venice Beach restaurant from chef Jasor. We serve Southern California Cuisine.

American Rose Society | Dedicated to America's favorite flower

www.rose.org/ ▼

Sep 30, 2016 - Official site of the ARS, dedicated to the enjoyment, enhancement and promotion of the rose. Features articles and links to help on **rose** ...

The Rose Art Museum | Brandeis University

www.brandeis.edu/rose/ ▼

The **Rose** is currently closed as we install our spring exhibitions. Please join us for our celebration, Thursday, February 16, 5 - 8 PM. > more.

■ roses

A screenshot of a Google search for the word "roses". The search bar contains the text "roses". Below the search bar, there are tabs for "All", "Images", "Videos", "Shopping", "Maps", and "More". The "All" tab is selected and underlined. Below the tabs, it says "About 615,000,000 results (0.87 seconds)". The first result is a video thumbnail for "The Chainsmokers - Roses (Audio) ft. ROZES - YouTube". The thumbnail shows a bouquet of pink roses with a play button in the center. Below the thumbnail, the text reads "The Chainsmokers - Roses (Audio) ft. ROZES - YouTube" and "https://www.youtube.com/watch?v=FyASdjZE0R0". Below the video information, there are lyrics, artist information, featured artist, album, release date, and genre. At the bottom, there is a "More about Roses" link with a downward arrow.

Guns N' Roses (@gunsnroses) · Twitter

<https://twitter.com/gunsnroses>

5. Deep Natural Language Analysis

- Has to do with more detailed Natural Language Processing algorithms
- E.g. semantic disambiguation, phrase indexing (board of directors), named entities (President Obama= Barak Obama) etc.
- Standard search engines increasingly use deeper techniques (e.g. Google's **Knowledge Graph** <http://www.google.com/insidesearch/features/search/knowledge.html> and **RankBrain**)
- More (on deep NLP techniques) in NLP course (not here)!



Tutti Notizie Immagini Video Maps Altro ▾ Strumenti di ricerca

Circa 178.000.000 risultati (0,65 secondi)

Risultati relativi a **Barack Obama**

Cerca invece [Barak Obama](#)

Barack Obama

<https://www.barackobama.com/> ▾ Traduci questa pagina

03 feb 2016 - OFA works to ensure the voices of ordinary Americans are heard in Washington, while training the next generation of grassroots organizers ...

Barack Obama - Wikipedia

https://it.wikipedia.org/wiki/Barack_Obama ▾

Barack Hussein Obama II (/bəˈrɑːk huːˈseɪn ouˈbɑːmə/, pronuncia; Honolulu, 4 agosto 1961) è un politico statunitense, 44° e attuale presidente degli ...

[Barack Obama, Sr. - Honolulu - Presidente degli Stati Uniti d ... - Mitt Romney](#)

Barack Obama - Wikipedia, the free encyclopedia

https://en.wikipedia.org/wiki/Barack_Obama

"Barack" and "Obama" redirect here. For his father, see **Barack Obama, Sr.** For other uses of "Barack", see [Barack \(disambiguation\)](#). For other uses of "Obama", ...

Nelle notizie



Corte Suprema ed esteri, così si rilancia Barack Obama

Il **Secolo XIX** - 5 ore fa

Da oggi c'è un protagonista in più, ed è l'attuale presidente Barack Obama. Dipenderà da ...

[San Valentino alla Casa Bianca: la poesia di Michelle per Barack - Video](#)

[Rai News](#) - 1 giorno fa

[Altre notizie su Barack Obama](#)

Knowledge map



Altre immagini

Barack Obama

44° presidente degli Stati Uniti

Barack Hussein Obama II è un politico statunitense, 44° e attuale presidente degli Stati Uniti d'America, primo afroamericano a ricoprire tale carica. [Wikipedia](#)

Data di nascita: 4 agosto 1961 (età 54), Honolulu, Hawaii, Stati Uniti

Altezza: 1,85 m

Coniuge: [Michelle Obama](#) (s. 1992)

Genitori: [Ann Dunham](#), [Barack Obama, Sr.](#)

Figli: [Malia Ann Obama](#), [Natasha Obama](#)

Nomine: [Nickelodeon Kids' Choice Award alla coppia più carina](#), altri

Ricerche correlate

Visualizza altri 15 elementi



Vladimir Putin



Michelle Obama
Moglie



Ann Dunham
Madre



Hillary Clinton



Donald Trump

1. Document Representation



2. Document Indexing

Why indexing

- The purpose of storing an index is to **optimize** speed and **performance** in finding relevant documents for a search query.
- Without an index, the search engine would scan every document in the document archive , which would require considerable time and computing power (especially if archive=the full web content).
- For example, while an index of 10,000 documents can be queried within milliseconds, a sequential scan of every word in 10,000 large documents could take hours.

Inverted index construction

Documents to be indexed



Friends, Romans, Countrymen.

Tokenizer

Token stream

Friends Romans Countrymen

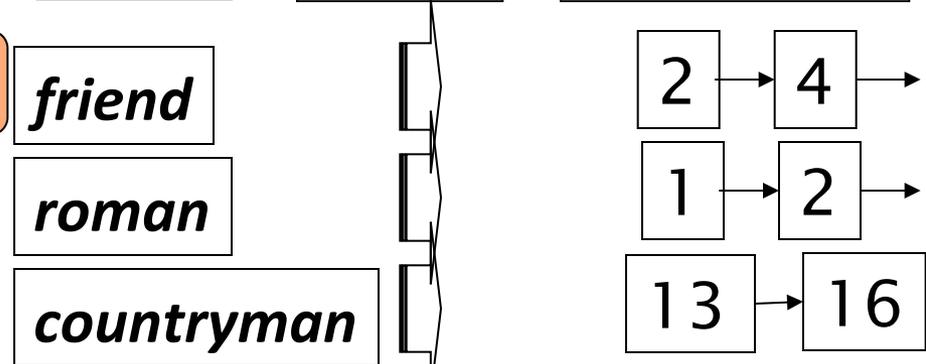
Linguistic modules

Modified tokens

friend roman countryman

Indexer

Inverted index



Indexer steps: Token sequence

- Sequence of (Modified token, Document ID) pairs.

Doc 1

Doc 2

I did enact Julius
Caesar I was killed
i' the Capitol;
Brutus killed me.

So let it be with
Caesar. The noble
Brutus hath told you
Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Indexer steps: Sort

- Sort by terms
 - And then “docID”



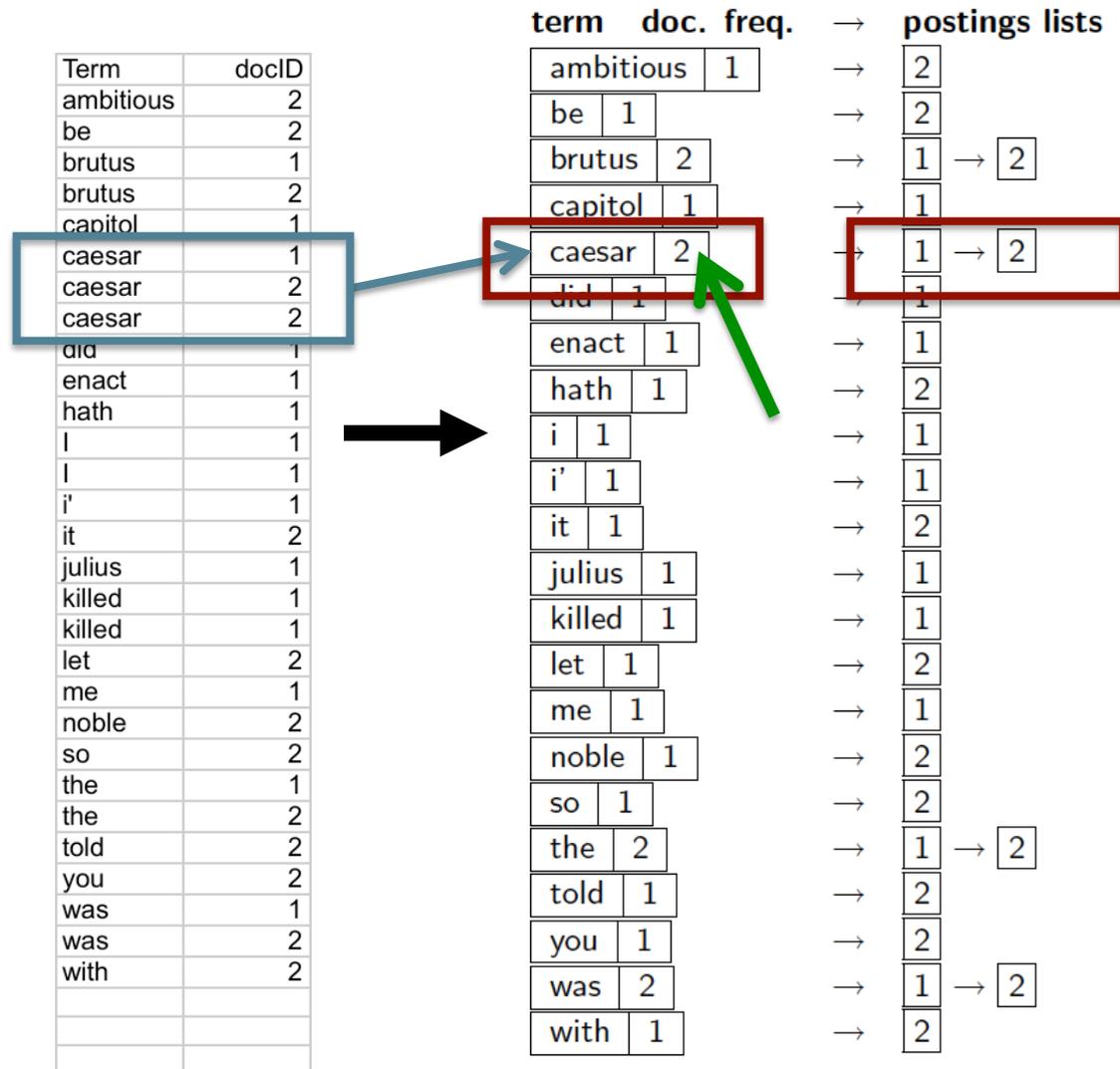
Core indexing step

Term	docID	Term	docID
I	1	ambitious	2
did	1	be	2
enact	1	brutus	1
julius	1	brutus	2
caesar	1	capitol	1
I	1	caesar	1
was	1	caesar	2
killed	1	caesar	2
i'	1	did	1
the	1	enact	1
capitol	1	hath	1
brutus	1	I	1
killed	1	I	1
me	1	i'	1
so	2	it	2
let	2	julius	1
it	2	killed	1
be	2	killed	1
with	2	let	2
caesar	2	me	1
the	2	noble	2
noble	2	so	2
brutus	2	the	1
hath	2	the	2
told	2	told	2
you	2	you	2
caesar	2	was	1
was	2	was	2
ambitious	2	with	2

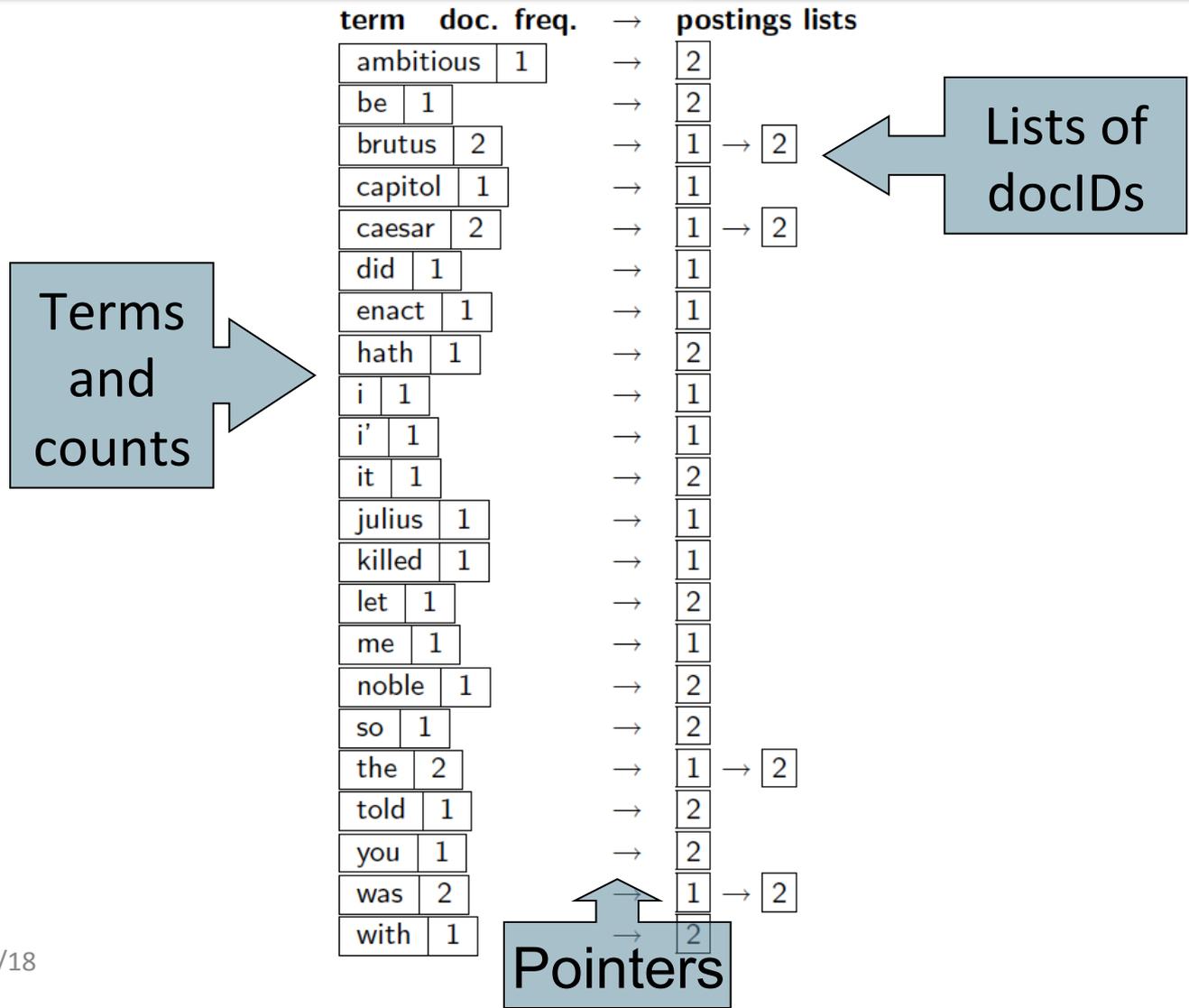
Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Doc. frequency information is added.**

Why frequency?
Will discuss later.

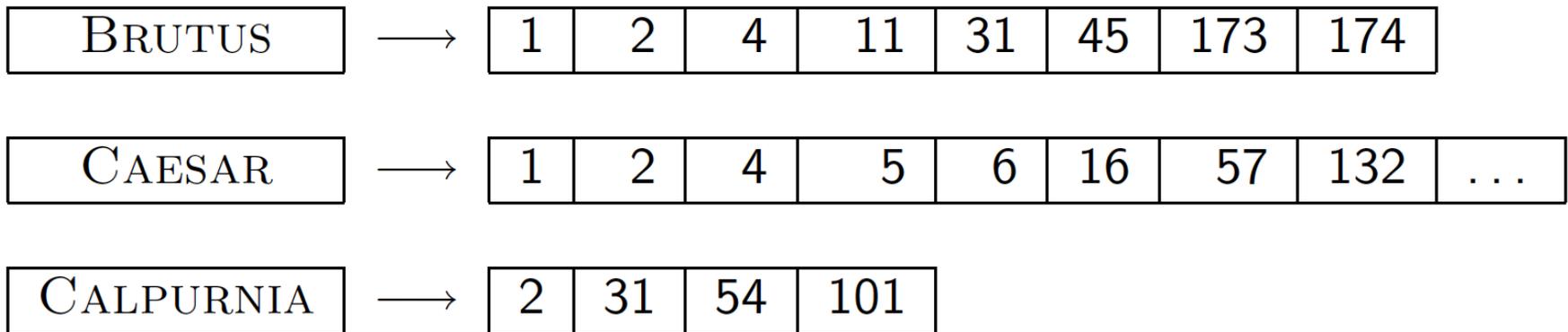


Where do we pay in storage?



Dictionary data structures for inverted indexes

- The dictionary data structure stores the term vocabulary, document frequency, pointers to each postings list ... **in what data structure?**



⋮

dictionary

postings

Array

- As we just have shown, the simplest structure is an **array**:

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

- Searching (lookup) an item is $O(\log(n))$; inserting an item $O(n)$
- Can we store a dictionary in memory more efficiently?

Alternative Dictionary data structures

- Two main choices:
 - Hashtables
 - Trees
- Some IR systems use hashtables, some trees

Hash tables (1)

1. Each vocabulary term is hashed to (mapped onto) an integer (We assume you have seen hashtables before)

- E.g. the index for a specific keyword will be equal to sum of ASCII values of characters multiplied by their respective order in the string after which it is modulo with 2069 (prime number).

String	Hash function	Index
abcdef	$(971 + 982 + 993 + 1004 + 1015 + 1026)\%2069$	38
bcdefa	$(981 + 992 + 1003 + 1014 + 1025 + 976)\%2069$	23
cdefab	$(991 + 1002 + 1013 + 1024 + 975 + 986)\%2069$	14
defabc	$(1001 + 1012 + 1023 + 974 + 985 + 996)\%2069$	11

Hash tables (2)

- 2. The keyword is stored in the hash table where it can be quickly retrieved using hashed key.

Index	
0	
1	
-	
-	
-	
11	defabc
12	
13	
14	cdefab
-	
-	
-	
-	
23	bcdefa
-	
-	
-	
38	abcdef
-	
-	

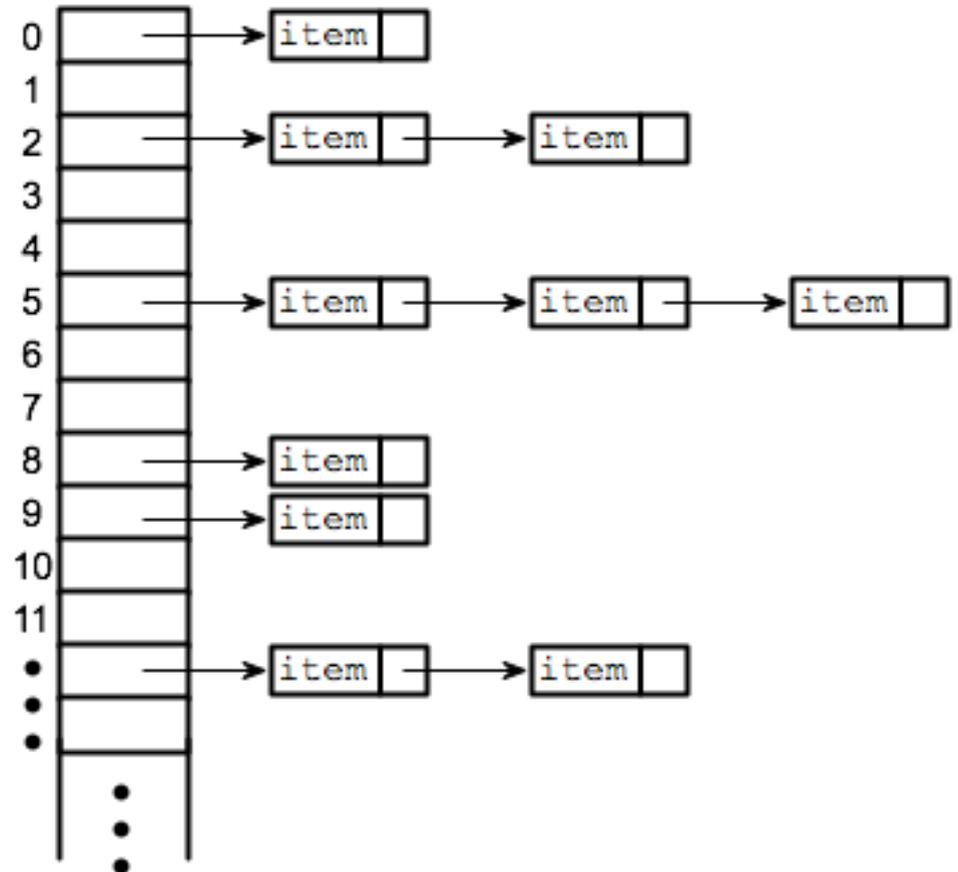
Hash tables (4)

- To achieve a good hashing mechanism, it is important to have a good hash function with the following basic requirements:
 - Easy to compute: It should be easy to compute and must not become an algorithm in itself.
 - Uniform distribution: It should provide a uniform distribution across the hash table and should not result in clustering.
 - Less collisions: Collisions occur when **pairs of elements are mapped to the same hash value**. These should be avoided.

Note: Irrespective of how good a hash function is, collisions occur. Therefore, to maintain the performance of a hash table, it is important to manage collisions through various **collision resolution** techniques.

Hash tables (5)

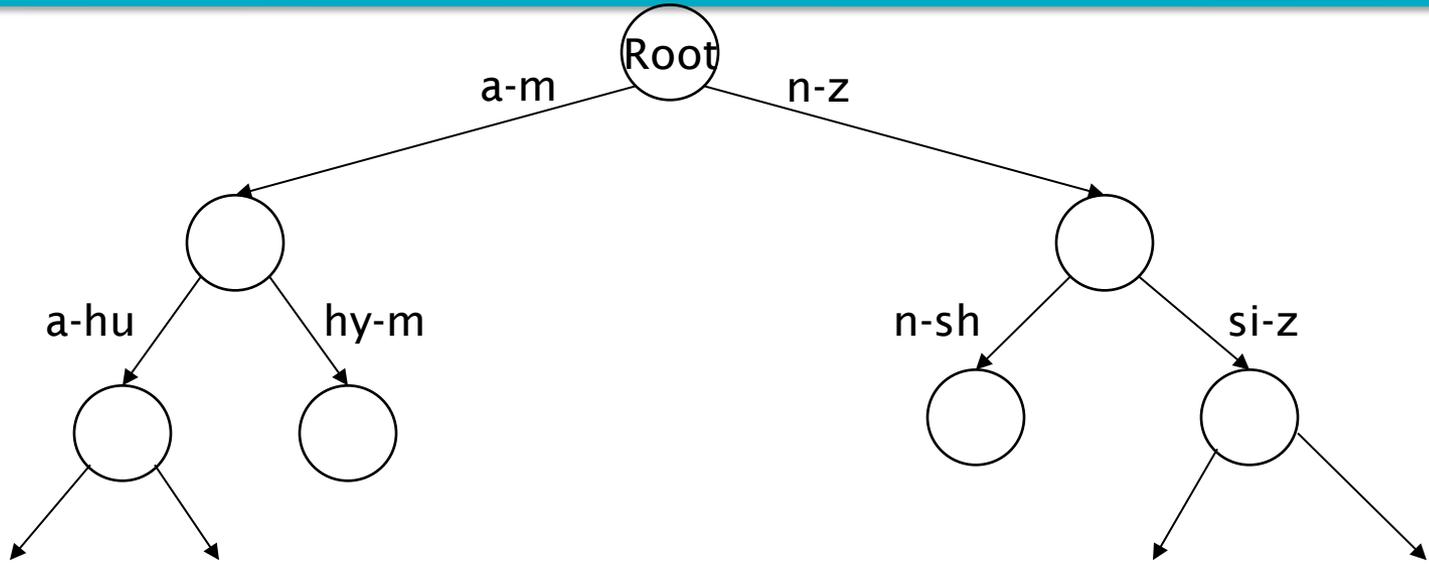
- Separate chaining is one of the most commonly used collision resolution techniques.
- It is usually implemented using linked lists.



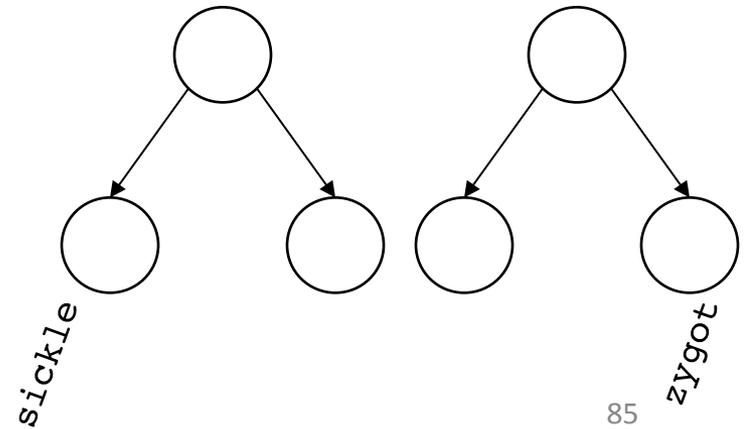
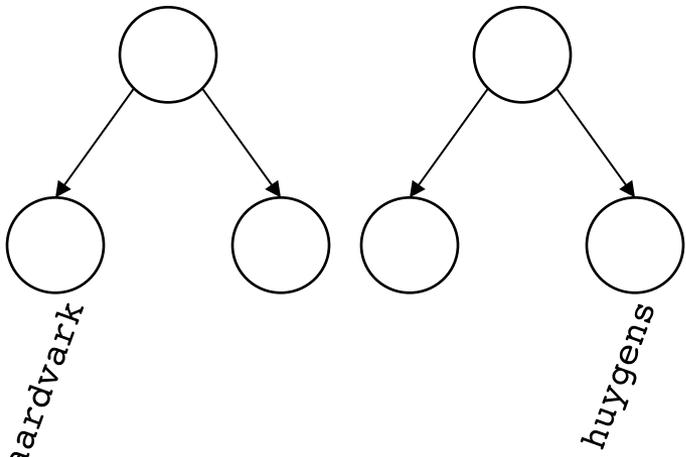
Hash Tables (6)

- Pros:
 - Lookup is faster: $O(1)$ (average) (depends on probability of collisions and collision-handling). Insertion is faster $O(1)$
 - Can reduce storage requirement if n (number of keywords) is much smaller than the universe of keys U
- Cons:
 - good, general purpose hash functions are very difficult to find
 - static table size requires costly resizing if indexed set is highly dynamic
 - search performance degrades considerably as the table nears its capacity (too many collisions)

Other structures: binary trees



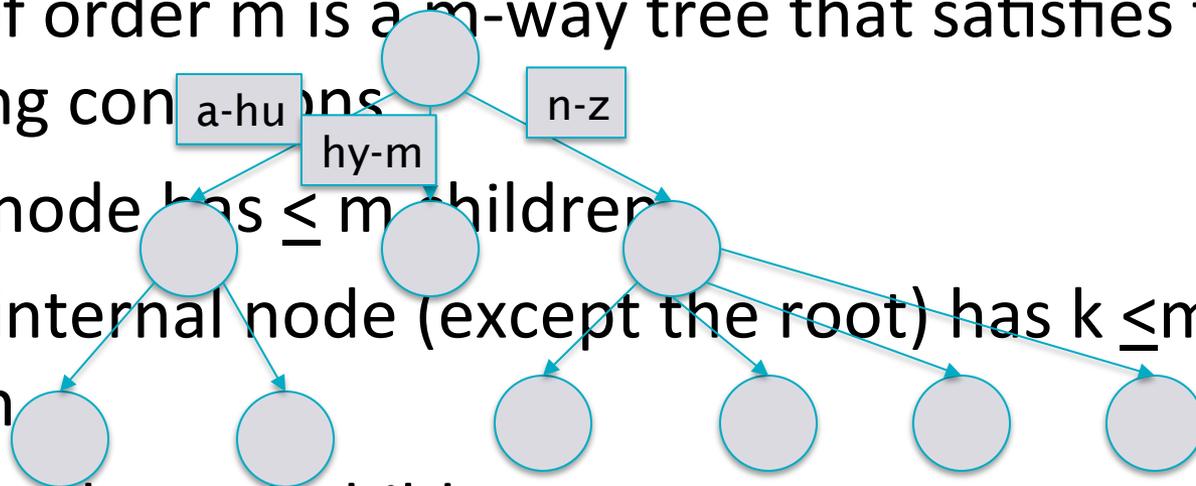
...



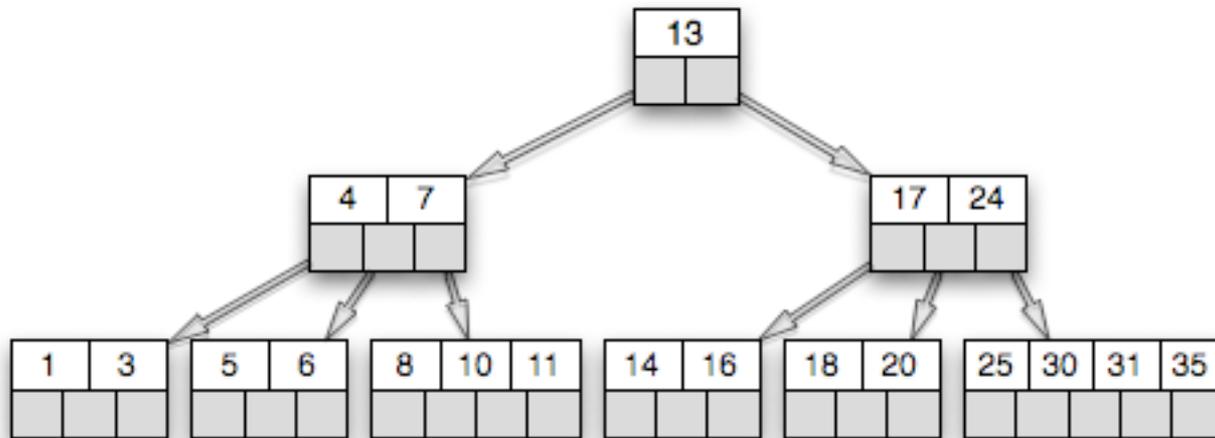
Tree: B-tree (balanced trees)

A B-tree of order m is a m -way tree that satisfies the following conditions

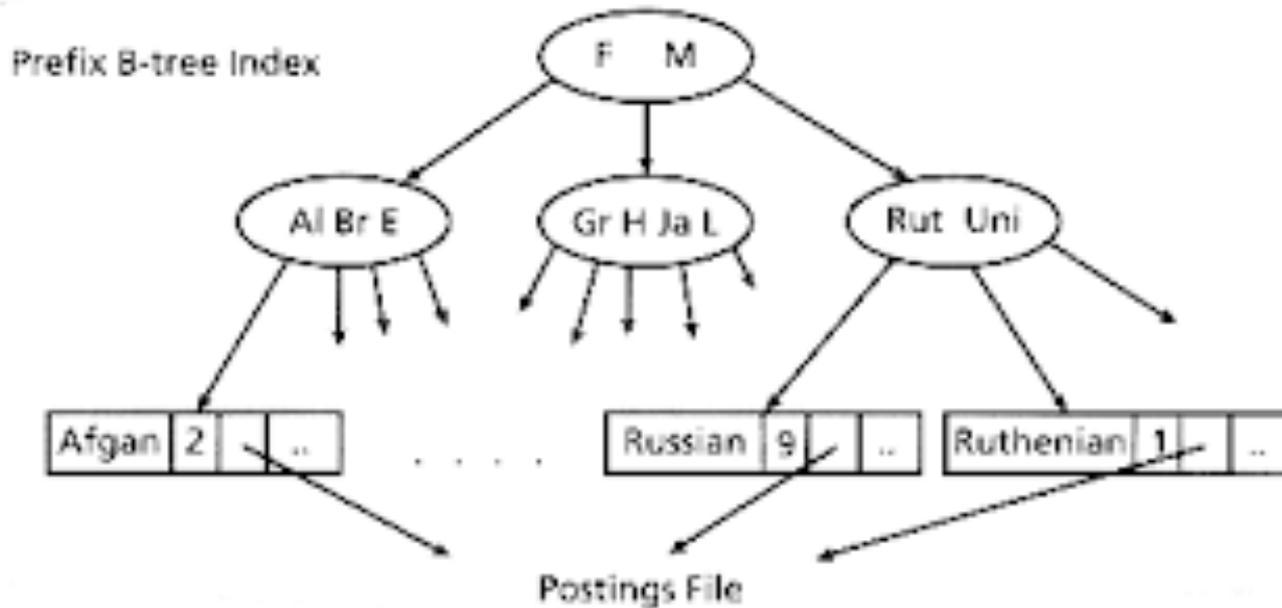
- Every node has $\leq m$ children
- Every internal node (except the root) has $k \leq m/2$ children
- The root has ≥ 2 children.
- An internal node with k children contains $(k-1)$ ordered keys. Its **leftmost** child contains keys less than or equal to the first key in the node. The second child contains keys **greater than the first keys but less than or equal to the second key**, and so on.



B-tree example (numeric)

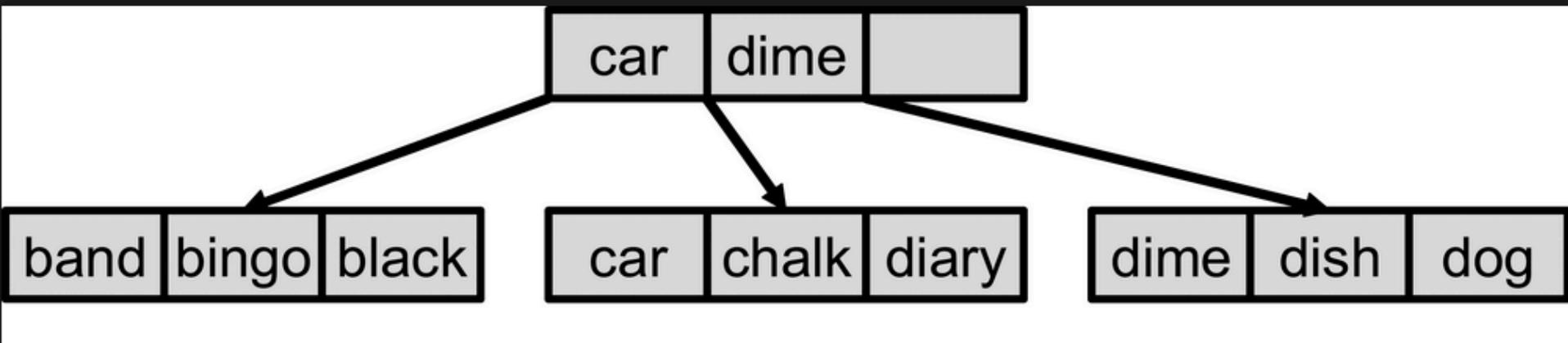


B-tree example (keywords)



Keywords are stored at the leaves of the tree, known as “buckets”

Another example (full words)



Inserting into a B-Tree

- To insert key value x into a B-Tree:
- Use the *B-Tree search* to determine on which node to make the insertion.
- Insert x into appropriate position on that leaf node.
- If resulting number of keys on that node $< k$, then simply output that node and return.
- Otherwise, **split** the node.

B-Trees pros and cons

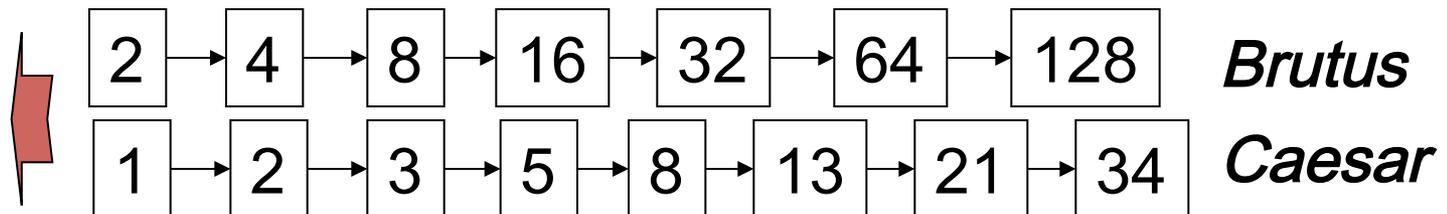
- Trees require a standard ordering of characters and hence strings ... but we typically have one
- Pros:
 - Size not limited as for hashing
- Cons:
 - Search is slower (wrt hash tables): $O(h)$ where $h = \log(n)$ is depth of B-tree [and this requires *balanced* tree]
 - B-trees mitigate the rebalancing problem (wrt standard binary trees)

Summary

- Once keywords are extracted, we create posting lists
- How do we search/insert a keyword?
Array, Hash Tables, Trees (B-trees, B+trees)
- Next problem is: How do we process a query (= searching documents with some combination of keywords)?

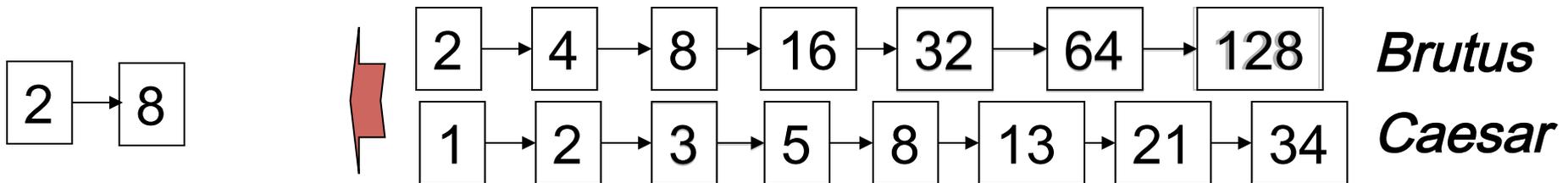
Query processing: AND

- Consider processing the query:
Brutus AND Caesar
 - Locate *Brutus* in the Dictionary;
 - Retrieve its postings (e.g. pointers to documents including Brutus).
 - Locate *Caesar* in the Dictionary;
 - Retrieve its postings.
 - “Merge” the two postings:



The “merge” operation

- Walk through the two postings simultaneously from right to left, in time linear in the total number of postings entries

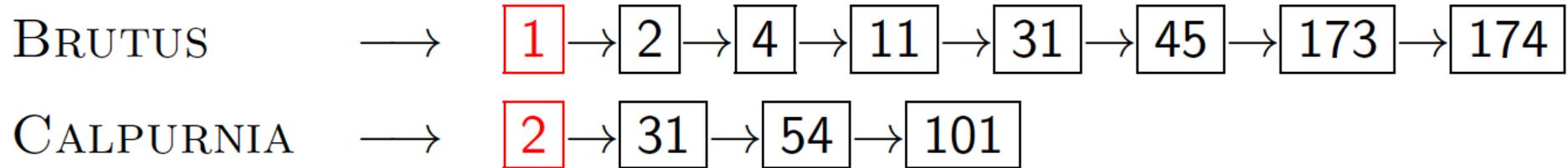


If list lengths are x and y , merge takes $O(x+y)$ operations.
Crucial: postings sorted by docID.

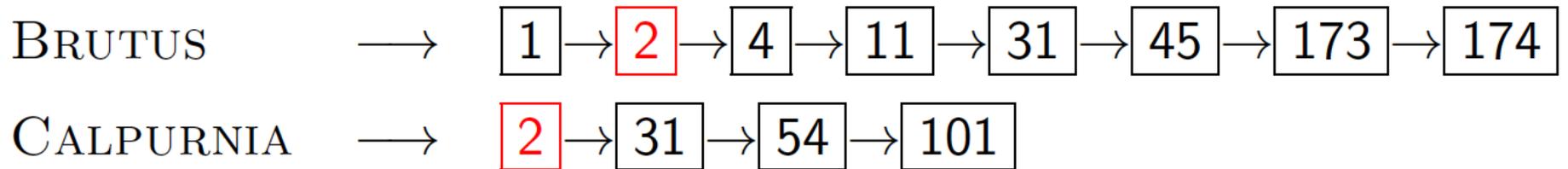
Intersecting two postings lists (a “merge” algorithm)

```
INTERSECT( $p_1, p_2$ )
1   $answer \leftarrow \langle \rangle$ 
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3  do if  $docID(p_1) = docID(p_2)$ 
4      then  $\text{ADD}(answer, docID(p_1))$ 
5           $p_1 \leftarrow next(p_1)$ 
6           $p_2 \leftarrow next(p_2)$ 
7      else if  $docID(p_1) < docID(p_2)$ 
8          then  $p_1 \leftarrow next(p_1)$ 
9          else  $p_2 \leftarrow next(p_2)$ 
10 return  $answer$ 
```

Step 1

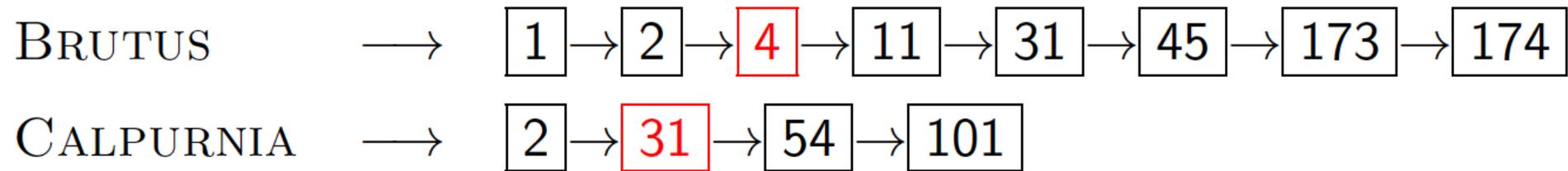


Step 2

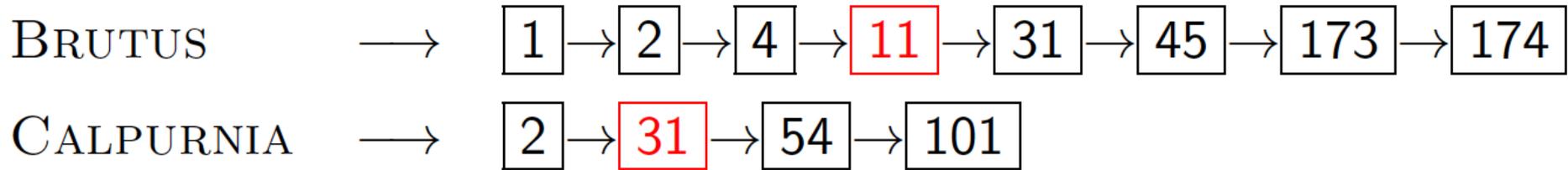


Intersection → **2**

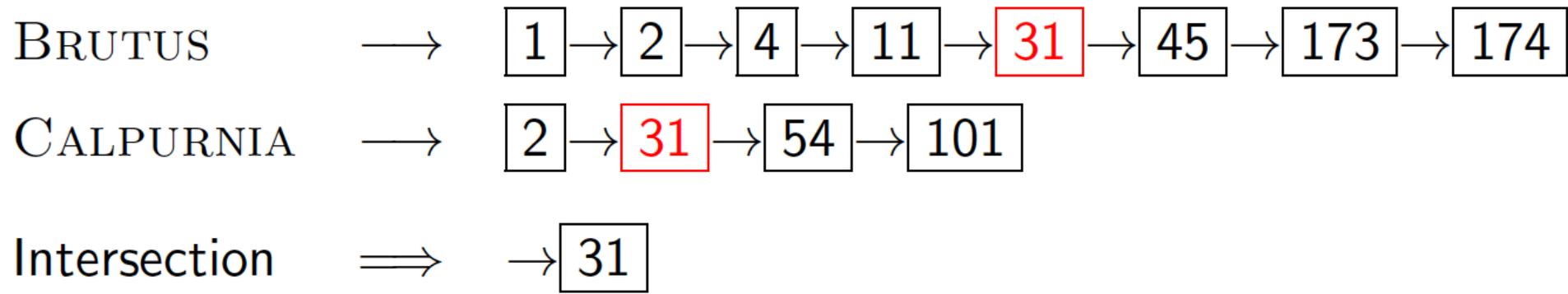
Step 3



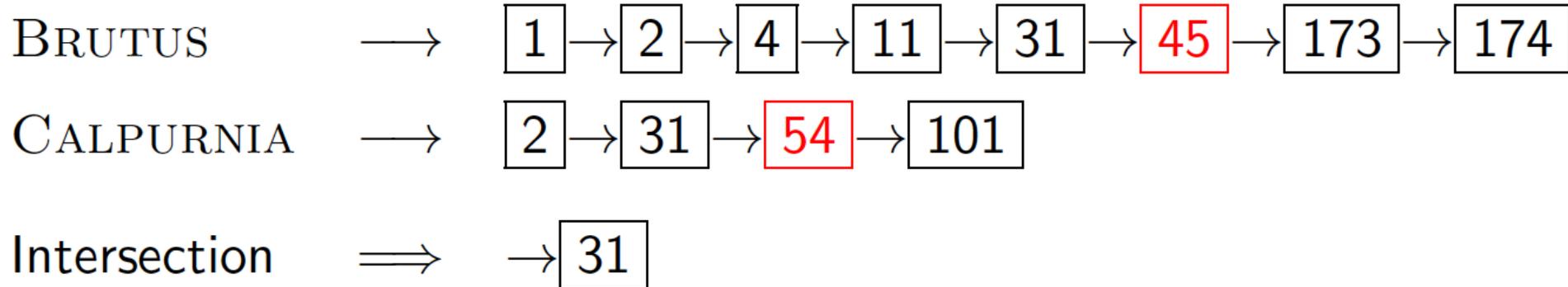
Step 4



Step 5

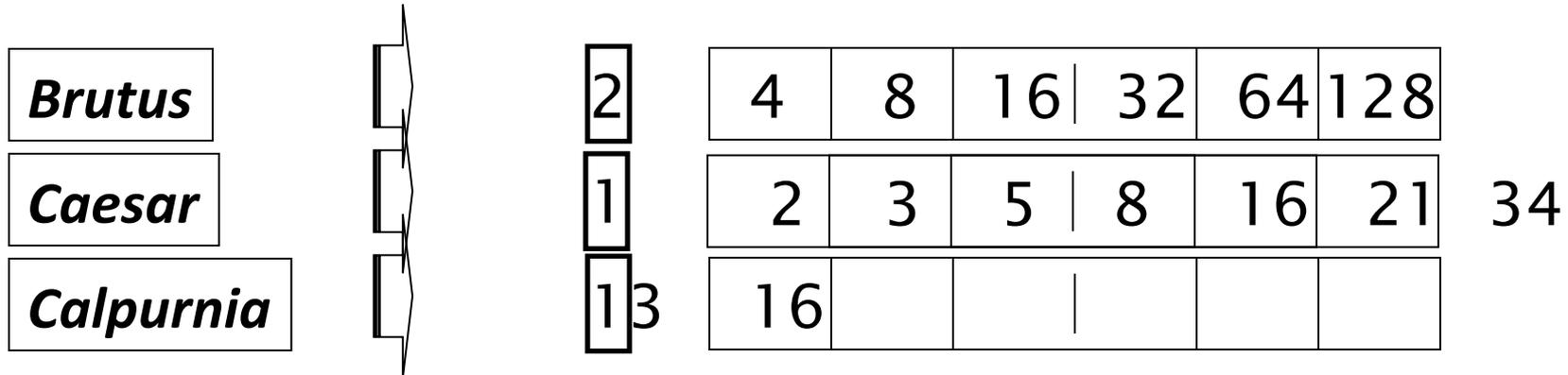


Step 6



Optimization of index search

- What is the best order of words for query processing?
- Consider a query that is an *AND* of n terms.
- For each of the n terms, get its postings, then *AND* them together.



Query: Brutus AND Calpurnia AND Caesar

Query optimization example

- Process words in order of increasing freq:
 - *start with smallest set (word with smallest list), then keep cutting further.*

This is why we kept document freq. in dictionary

Brutus	}	2	4	8	16	32	64	128	
Caesar		1	2	3	5	8	16	21	34
Calpurnia		3	16						

Execute the query as **(Calpurnia AND Brutus) AND Caesar.**

More general optimization

- e.g., (*madding OR crowd*) AND (*ignoble OR strife*)
- Get doc. freq.'s for all terms.
- Estimate the **size of each OR** by the **sum** of its doc. freq.'s (conservative).
- Process AND **in increasing order of OR** sizes.

Example

OR
size

- Recommend a query processing order for:

*(tangerine OR trees) AND
(marmalade OR skies) AND
(kaleidoscope OR eyes)*

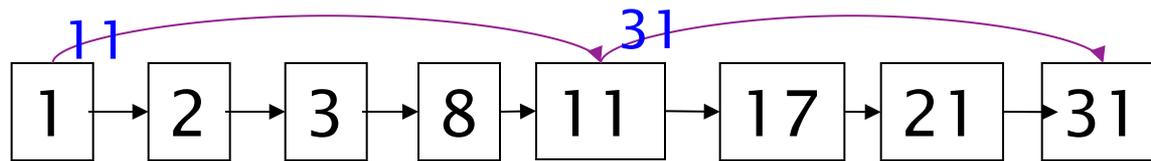
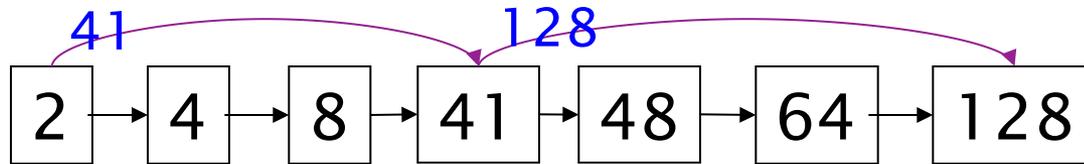
Term	Freq	
eyes	213312	300321
kaleidoscope	87009	
marmalade	107913	
skies	271658	379571
tangerine	46653	
trees	316812	363465

(kaleydoscopeOREyes)AND(tangerineORtrees)AND(marmaladeORskies)

Skip pointers

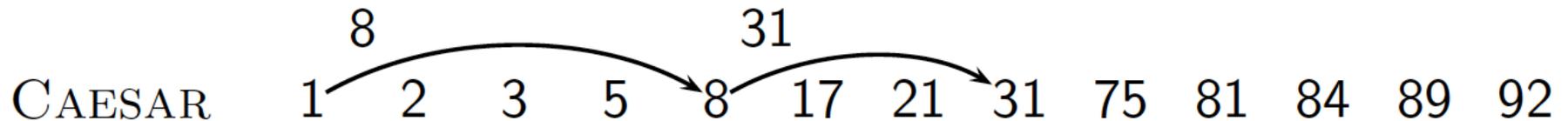
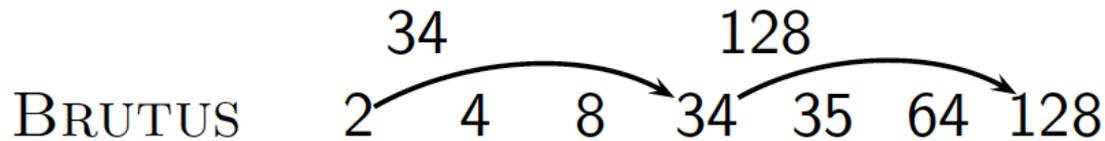
- Intersection is the most important operation when it comes to search engines.
- This is because **in web search, most queries are implicitly intersections**: *e.g. "car repairs", "britney spears songs" etc.* translates into – "**car AND repairs**", "**britney AND spears AND songs**", which means it will be intersecting 2 or more postings lists in order to return a result.
- Because intersection is so crucial, search engines try **to speed it up** in any possible way. One such way is to use **skip pointers**.

Augment postings with skip pointers (at indexing time)



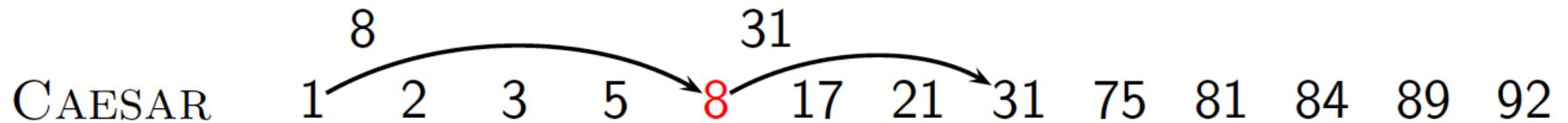
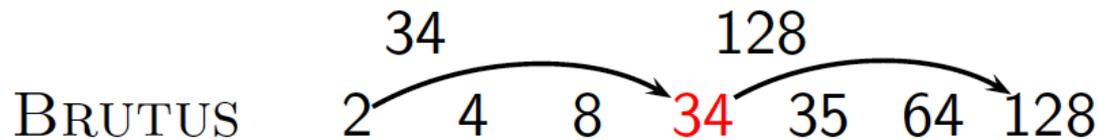
- Idea: move cursor on a posting list to the first posting where DocID is equal or larger than the compared one.
- Why? To avoid unnecessary comparisons
- Where do we place skip pointers?

Example: Step 1



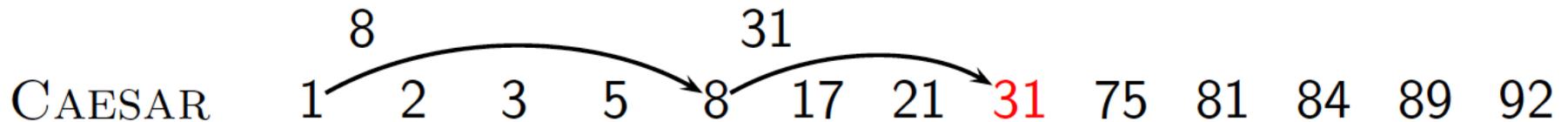
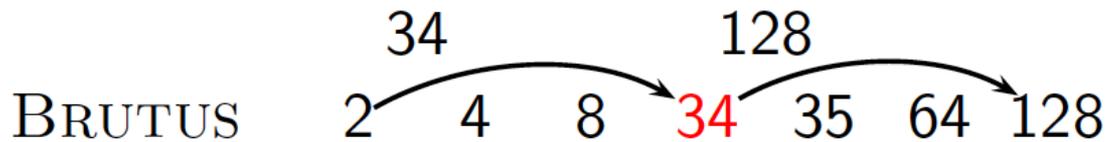
Some items have two pointers, one pointing to an adjacent item the other skipping a few items ahead

Step 2



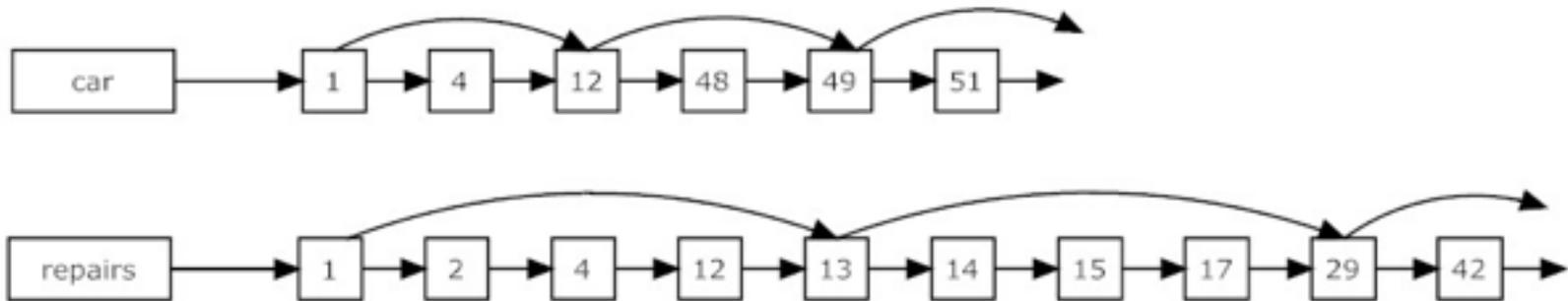
Say we are comparing $p_2=8$ and $p_1=34$: since $8 < 34$, we must move the pointer to the right. But 8 has a skip! So we first compare the id of the skip (31) with the id of p_1 (34).

Step 3



Since $31 < 34$, we skip from 8 to 31 avoiding useless comparisons with 17 and 21!

Another example



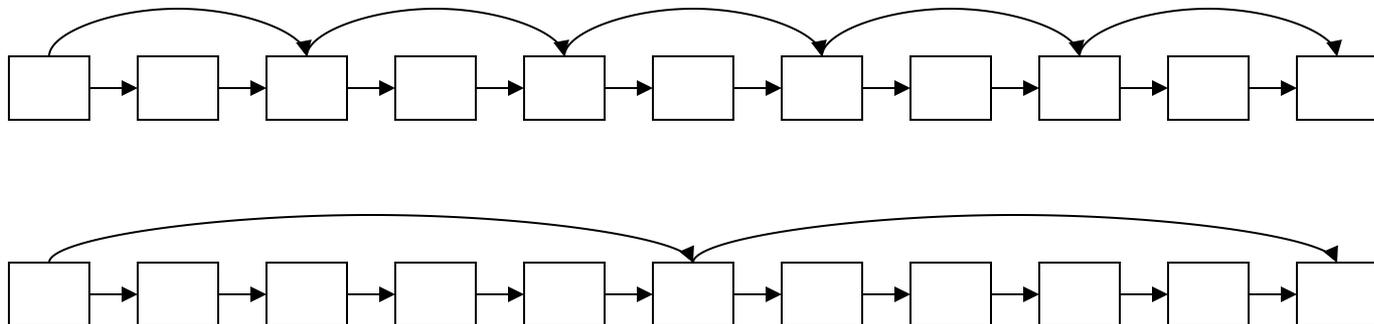
Example: Start using the normal intersection algorithm.

Continue until the match **12** and advance to the next item in each list (48, 13). At this point the "car" list is on **48** and the "repairs" list is on **13**, but 13 has a **skip pointer**.

Check the value the skip pointer is pointing at (i.e. **29**) and **if** this value is **less than the current value of the "car" list** (which it is **48** in our example), we follow our skip pointer and jump to this value in the list. It would be useless to compare all elements between the current and subsequent skip!!

Where do we place skips?

- Tradeoff:
 - More skips \rightarrow shorter skip spans \Rightarrow more likely to skip.
But **lots of comparisons** to skip pointers.
 - Fewer skips \rightarrow few pointer comparison, but then long skip spans \Rightarrow **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.
 - How much do skip pointers help?
 - Traditionally, CPUs were slow , they used to help a lot.
 - But today's CPUs are fast and disk is slow, so **reducing disk postings list size dominates.**

Algorithm INTERSECT with skip pointers

INTERSECTWITHSKIPS(p_1, p_2)

```
1  answer  $\leftarrow \langle \rangle$ 
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$ 
4      then  $\text{ADD}(\text{answer}, \text{docID}(p_1))$ 
5           $p_1 \leftarrow \text{next}(p_1)$ 
6           $p_2 \leftarrow \text{next}(p_2)$ 
7  else if  $\text{docID}(p_1) < \text{docID}(p_2)$ 
8      then if  $\text{hasSkip}(p_1)$  and  $(\text{docID}(\text{skip}(p_1)) \leq \text{docID}(p_2))$ 
9          then while  $\text{hasSkip}(p_1)$  and  $(\text{docID}(\text{skip}(p_1)) \leq \text{docID}(p_2))$ 
10             do  $p_1 \leftarrow \text{skip}(p_1)$ 
11             else  $p_1 \leftarrow \text{next}(p_1)$ 
12         else if  $\text{hasSkip}(p_2)$  and  $(\text{docID}(\text{skip}(p_2)) \leq \text{docID}(p_1))$ 
13             then while  $\text{hasSkip}(p_2)$  and  $(\text{docID}(\text{skip}(p_2)) \leq \text{docID}(p_1))$ 
14                 do  $p_2 \leftarrow \text{skip}(p_2)$ 
15                 else  $p_2 \leftarrow \text{next}(p_2)$ 
16 return answer
```

Phrase queries

- Want to be able to answer queries such as "**red brick house**" – as a phrase
- **red AND brick AND house** match phrases such as "*red house near the brick factory*" which is not what we are searching for
 - The concept of phrase queries has proven easily understood by users; **one of the few “advanced search” ideas that works**
 - **About 10% of web queries are phrase queries.**
- For this, it no longer suffices to store only *<term : docs>* entries

A first attempt: Bi-word indexes

- 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

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

stanford university AND university palo AND palo alto

Extended biwords

- Parse the indexed text and perform part-of-speech-tagging (POS Tagging).
- Identify Nouns (N) and articles/prepositions (X).
- Call any string of terms of the form NX*N (regex) an extended biword (noun followed by article/prep followed by anything followed by noun).
 - 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
 - Segment query into enhanced biwords
 - Look up in index: ***catcher rye (NN)***

Issues for biword indexes

- Index **blowup** due to bigger dictionary
 - Infeasible for more than biwords, big even for them
- Biword indexes are not the standard solution (for all biwords) but can **be part of a compound strategy**

Solution 2: Positional indexes

- **Positional indexes** are a more efficient alternative to biword indexes.
- **In a non-positional index each posting is a document ID**
- **In a positional index each posting is a docID and a list of positions**

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

doc1: position1, position2 ... ;

doc2: position1, position2 ... ;

etc.>

to, 993427:

⟨ 1, 6: ⟨7, 18, 33, 72, 86, 231⟩;
 2, 5: ⟨1, 17, 74, 222, 255⟩;
 4, 5: ⟨8, 16, 190, 429, 433⟩;
 5, 2: ⟨363, 367⟩;
 7, 3: ⟨13, 23, 191⟩; ... ⟩

be, 178239:

⟨ 1, 2: ⟨17, 25⟩;
 4, 5: ⟨17, 191, 291, 430, 434⟩;
 5, 3: ⟨14, 19, 101⟩; ... ⟩

Example: search for «to be»

Query: “*to₁ be₂ or₃ not₄ to₅ be₆*”

TO, 993427:

⟨ 1,6: ⟨7, 18, 33, 72, 86, 231⟩;

2,5: ⟨1, 17, 74, 222, 255⟩;

4,5: ⟨8, 16, 190, 429, 433⟩;

5,2: ⟨363, 367⟩;

7,3: ⟨13, 23, 191⟩; ... ⟩

Frequency
in doc

BE, 178239:

⟨ 1,2: ⟨17, 25⟩;

4,5: ⟨17, 191, 291, 430, 434⟩;

5,3: ⟨14, 19, 101⟩; ... ⟩

Step 1: “to” and “be” co-occur in DOC 1

Query: “*to*₁ *be*₂ *or*₃ *not*₄ *to*₅ *be*₆”

TO, 993427:

⟨ 1,6: ⟨7, 18, 33, 72, 86, 231⟩;

2,5: ⟨1, 17, 74, 222, 255⟩;

4,5: ⟨8, 16, 190, 429, 433⟩;

5,2: ⟨363, 367⟩;

7,3: ⟨13, 23, 191⟩; ... ⟩

BE, 178239:

⟨ 1,2: ⟨17, 25⟩;

4,5: ⟨17, 191, 291, 430, 434⟩;

5,3: ⟨14, 19, 101⟩; ... ⟩

Step 2

TO, 993427:

⟨ 1,6: ⟨7, 18, 33, 72, 86, 231⟩;

2,5: ⟨1, 17, 74, 222, 255⟩;

4,5: ⟨8, 16, 190, 429, 433⟩;

5,2: ⟨363, 367⟩;

7,3: ⟨13, 23, 191⟩; ... ⟩

Not consecutive!

BE, 178239:

⟨ 1,2: ⟨17, 25⟩;

4,5: ⟨17, 191, 291, 430, 434⟩;

5,3: ⟨14, 19, 101⟩; ... ⟩

Step 3 (move pointer of “to” in DOC 1)

TO, 993427:

⟨ 1,6: ⟨7, 18, 33, 72, 86, 231⟩;

2,5: ⟨1, 17, 74, 222, 255⟩;

4,5: ⟨8, 16, 190, 429, 433⟩;

5,2: ⟨363, 367⟩;

7,3: ⟨13, 23, 191⟩; ... ⟩

Not a match!
18 is after 17

BE, 178239:

⟨ 1,2: ⟨17, 25⟩;

4,5: ⟨17, 191, 291, 430, 434⟩;

5,3: ⟨14, 19, 101⟩; ... ⟩

Step 4 (move pointer of “BE” in doc 1)

TO, 993427:

⟨ 1,6: ⟨7, 18, 33, 72, 86, 231⟩;
2,5: ⟨1, 17, 74, 222, 255⟩;
4,5: ⟨8, 16, 190, 429, 433⟩;
5,2: ⟨363, 367⟩;
7,3: ⟨13, 23, 191⟩; ... ⟩

BE, 178239:

⟨ 1,2: ⟨17, 25⟩;
4,5: ⟨17, 191, 291, 430, 434⟩;
5,3: ⟨14, 19, 101⟩; ... ⟩

Step 5

Query: “ $to_1 be_2 or_3 not_4 to_5 be_6$ ”

TO, 993427:

⟨ 1,6: ⟨7, 18, 33, 72, 86, 231⟩;

2,5: ⟨1, 17, 74, 222, 255⟩;

4,5: ⟨8, 16, 190, 429, 433⟩;

5,2: ⟨363, 367⟩;

7,3: ⟨13, 23, 191⟩; ... ⟩

No matches in
DOC1 !!

BE, 178239:

⟨ 1,2: ⟨17, 25⟩;

4,5: ⟨17, 191, 291, 430, 434⟩;

5,3: ⟨14, 19, 101⟩; ... ⟩

Step 6: start looking in DOC 4

Query: “ $to_1 be_2 or_3 not_4 to_5 be_6$ ”

TO, 993427:

⟨ 1,6: ⟨7, 18, 33, 72, 86, 231⟩;

2,5: ⟨1, 17, 74, 222, 255⟩;

4,5: ⟨8, 16, 190, 429, 433⟩;

5,2: ⟨363, 367⟩;

7,3: ⟨13, 23, 191⟩; ... ⟩

BE, 178239:

⟨ 1,2: ⟨17, 25⟩;

4,5: ⟨17, 191, 291, 430, 434⟩;

5,3: ⟨14, 19, 101⟩; ... ⟩

After a number of steps..

Query: “ $to_1 be_2 or_3 not_4 to_5 be_6$ ”

TO, 993427:

⟨ 1,6: ⟨7, 18, 33, 72, 86, 231⟩;

2,5: ⟨1, 17, 74, 222, 255⟩;

4,5: ⟨8, 16, 190, 429, 433⟩;

5,2: ⟨363, 367⟩;

7,3: ⟨13, 23, 191⟩; ... ⟩

429->430

433->434!!

DOC4 has 2 matches

BE, 178239:

⟨ 1,2: ⟨17, 25⟩;

4,5: ⟨17, 191, 291, 430, 434⟩;

5,3: ⟨14, 19, 101⟩; ... ⟩

Proximity search

- We just saw how to use a positional index for phrase searches (phrase: sequence of consecutive words).
- We can also use it for proximity search.
- For example: employment /4 place: **Find all documents that contain EMPLOYMENT and PLACE within 4 words of each other.**
- *“Employment agencies that place healthcare workers are seeing growth”* **is a hit.**
- *“Employment agencies that have learned to adapt now place healthcare workers”* **is not a hit.**

Proximity search

- Use the positional index
- Simplest algorithm: look at cross-product of positions of (i) EMPLOYMENT in document and (ii) PLACE in document
- Very inefficient for frequent words, especially stop words
- Note that we want to return the actual matching positions, not just a list of documents.

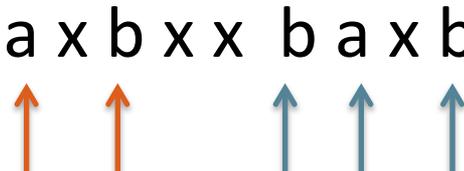
Proximity intersection

```
POSITIONALINTERSECT( $p_1, p_2, k$ )
1   $answer \leftarrow \langle \rangle$ 
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$ 
4      then  $l \leftarrow \langle \rangle$ 
5           $pp_1 \leftarrow \text{positions}(p_1)$ 
6           $pp_2 \leftarrow \text{positions}(p_2)$ 
7          while  $pp_1 \neq \text{NIL}$ 
8          do while  $pp_2 \neq \text{NIL}$ 
9              do if  $|\text{pos}(pp_1) - \text{pos}(pp_2)| \leq k$ 
10                 then  $\text{ADD}(l, \text{pos}(pp_2))$ 
11                 else if  $\text{pos}(pp_2) > \text{pos}(pp_1)$ 
12                     then break
13                  $pp_2 \leftarrow \text{next}(pp_2)$ 
14                 while  $l \neq \langle \rangle$  and  $|l[0] - \text{pos}(pp_1)| > k$ 
15                     do  $\text{DELETE}(l[0])$ 
16                 for each  $ps \in l$ 
17                     do  $\text{ADD}(answer, \langle \text{docID}(p_1), \text{pos}(pp_1), ps \rangle)$ 
18                  $pp_1 \leftarrow \text{next}(pp_1)$ 
19              $p_1 \leftarrow \text{next}(p_1)$ 
20              $p_2 \leftarrow \text{next}(p_2)$ 
21         else if  $\text{docID}(p_1) < \text{docID}(p_2)$ 
22             then  $p_1 \leftarrow \text{next}(p_1)$ 
23         else  $p_2 \leftarrow \text{next}(p_2)$ 
24 return  $answer$ 
```

An algorithm for proximity intersection of postings lists p_1 and p_2 .

The algorithm finds places where the two terms appear within k words of each other and returns a **list of triples** giving docID and the term position in p_1 and p_2 .

Example (search for a,b at max distance $k=2$)

- **1**: 1 2 3 4 5 6 7 8 9
- a x b x x b a x b

- $l = \langle 3 \rangle$ (pos(b))
- **$\langle 1, 1, 3 \rangle$** $\langle \text{DocID}, \text{pos}(a), \text{pos}(b) \rangle$
- $l = \langle 3, 6 \rangle$
- $l = \langle 6 \rangle$, **$\langle 1, 7, 6 \rangle$**
- **etc**

```

POSITIONALINTERSECT( $p_1, p_2, k$ )
1  answer  $\leftarrow \langle \rangle$ 
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$ 
4     then  $l \leftarrow \langle \rangle$ 
5          $pp_1 \leftarrow \text{positions}(p_1)$ 
6          $pp_2 \leftarrow \text{positions}(p_2)$ 
7         while  $pp_1 \neq \text{NIL}$ 
8         do while  $pp_2 \neq \text{NIL}$ 
9             do if  $|\text{pos}(pp_1) - \text{pos}(pp_2)| \leq k$ 
10                then ADD( $l, \text{pos}(pp_2)$ )
11                else if  $\text{pos}(pp_2) > \text{pos}(pp_1)$ 
12                   then break
13                 $pp_2 \leftarrow \text{next}(pp_2)$ 
14            while  $l \neq \langle \rangle$  and  $|l[0] - \text{pos}(pp_1)| > k$ 
15                do DELETE( $l[0]$ )
16            for each  $ps \in l$ 
17                do ADD( $\text{answer}, \langle \text{docID}(p_1), \text{pos}(pp_1), ps \rangle$ )
18             $pp_1 \leftarrow \text{next}(pp_1)$ 
19         $p_1 \leftarrow \text{next}(p_1)$ 
20         $p_2 \leftarrow \text{next}(p_2)$ 
21    else if  $\text{docID}(p_1) < \text{docID}(p_2)$ 
22        then  $p_1 \leftarrow \text{next}(p_1)$ 
23    else  $p_2 \leftarrow \text{next}(p_2)$ 
24    return answer
    
```

Positional index size

- 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%

Document size	Postings	Positional postings
1 000	1	1
1 00,000	1	1 00

Positional index size

- Positional index expands postings storage *substantially*
 - some rough rules of thumb are to expect a positional index to be **2 to 4 times** as large as a non-positional index
- Positional index is now **standardly used** because of the power and usefulness of phrase and proximity queries

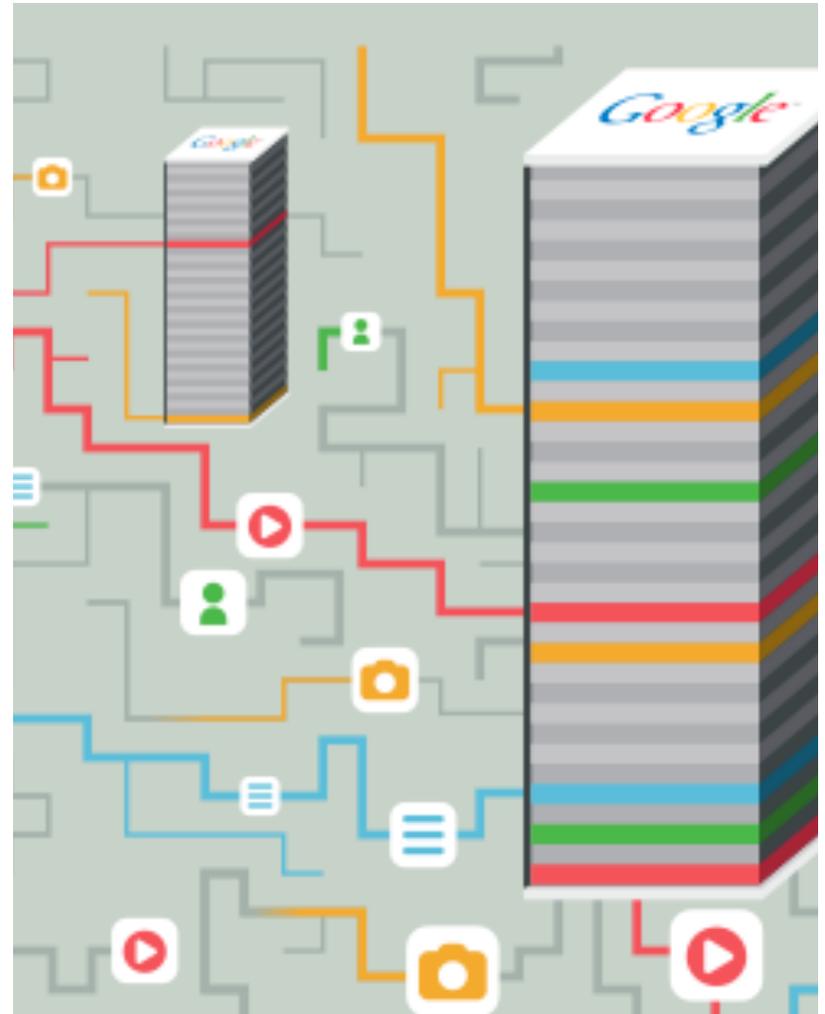
Combined scheme

- Biword indexes and positional indexes can be profitably combined.
- Many biwords are extremely frequent: Michael Jackson, Britney Spears etc
- For these biwords, increased speed compared to positional postings intersection is substantial.
- **Combination scheme:** Include frequent biwords as vocabulary terms in the index. Do all other phrases by positional intersection.

Google indexing system

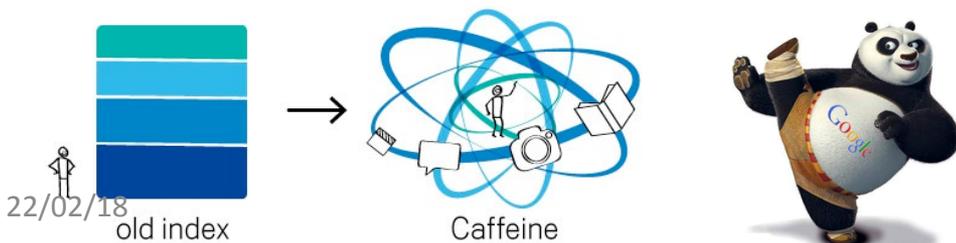
- Google is changing the way to handle its index continuously
- See an history on:

<http://moz.com/google-algorithm-change>



Caffeine+Panda, Google Index

- Major recent changes have been Caffeine & Panda
- **Caffeine:**
 - Old index had several layers, some of which were refreshed at a faster rate than others (they had different indexes); the main layer would update every couple of weeks (“**Google dance**”)
 - Caffeine analyzes the web in small portions and update search index **on a continuous basis, globally**. As new pages are found, or new information on existing pages, these are added straight to the index.
- **Panda:** aims to promote the high quality content site by dooming the rank of low quality content sites.
- Note: Caffeine is parte of the INDEXING strategy, not searching (later in this course)



Suggested Reading: <https://arxiv.org/pdf/1712.01208.pdf>

The Case for Learned Index Structures

David Kraska*
MIT
Cambridge, MA
kraska@mit.edu

Alex Beutel
Google, Inc.
Mountain View, CA
alexbeutel@google.com

Ed H. Chi
Google, Inc.
Mountain View, CA
edchi@google.com

Jeffrey Dean
Google, Inc.
Mountain View, CA
jeff@google.com

Neoklis Polyzotis
Google, Inc.
Mountain View, CA
npolyzotis@google.com