Part 1

- Introduction to Information Retrieval (IR)
  - What is Information Retrieval?
  - What is IR about?
  - How is IR related to Information Science

Acknowledgment: Manning, C.D., Raghavan, P. and Schütze, H., *Introduction to Information Retrieval, Cambridge University Press. 2008.*

INFO 300

Information Retrieval Systems

Introduction

Weimao Ke

wk@drexel.edu

Information Retrieval

- Mooers (1951) coined ***Information Retrieval:***
  - The investigation of information description and specification for search and techniques for search operations. (see also Saracevic, 1999)
- Mooers' Law (1959) [NOT the Moore's law]
  - "An information retrieval system will tend NOT to be used whenever it is more painful and troublesome for a customer to have information than for him not to have it."

IR and Information Science

- Baeza-Yates and Ribeiro-Neto (2004):
  - Information Retrieval (IR) "deals with the representation, storage, organization of, and access to information items."
- Considered an important area of Information Science
  - Information Science is about "gathering, organizing, storing, retrieving, and dissemination of information." (Bates, 1999)
  - IR has a long tradition and root in Information & Library Science
  - Computer science has strong influence on system-centric IR

IR to Information Science

- Saracevic (1999):

> "Surely, information science is more than IR, but many of the problems raised by IR or derived from objects and phenomena involved in IR, are at its core."

Cranfield Tests and IR Tradition

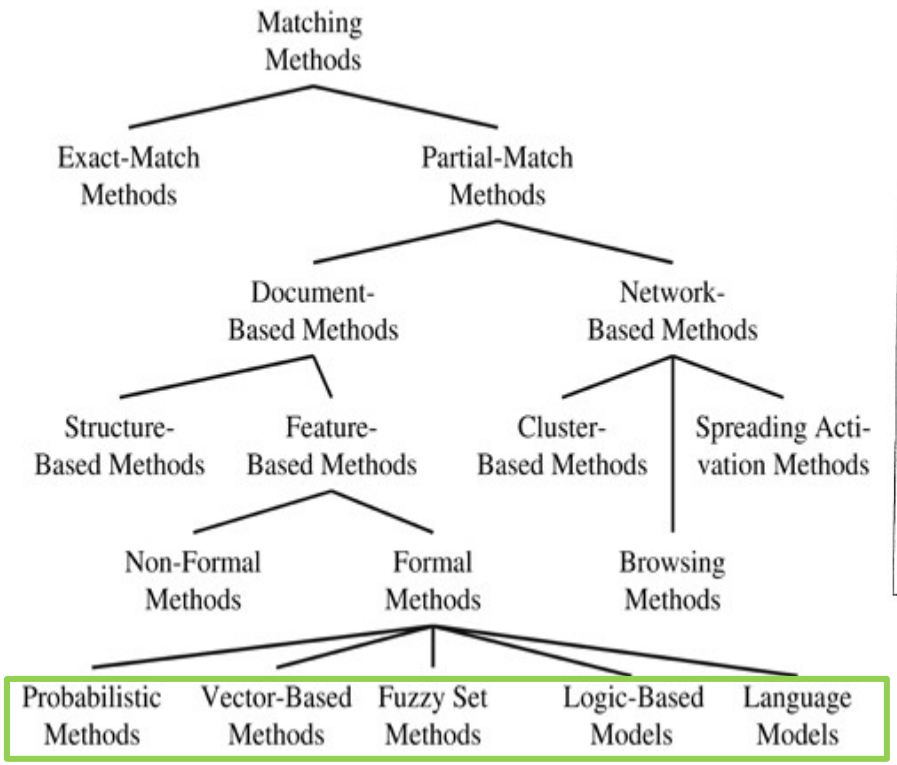
- Cranfield tests
  - A series of early experiments
  - led by Cyril W. Cleverdon (a librarian, see picture :)
  - at College of Aeronautics at Cranfield
  - on retrieval effectiveness of index languages/techniques
- The tests established an experimental IR tradition
  - A document collection
  - A set of queries
  - Relevance judgment
  - Evaluation metrics, e.g., precision and recall

IR Tradition continues

- The Text Retrieval Conference (TREC)
  - A platform where IR systems can compete/be compared
  - Large scale evaluation
  - Various “tracks” beyond text retrieval
  - Each TREC track, e.g., Web track for IR on the Web
    - A benchmark document collection (e.g., web pages)
    - Pre-defined tasks and queries
    - A common relevance base (manually judged by humans)
    - Standardized evaluation procedures and metrics

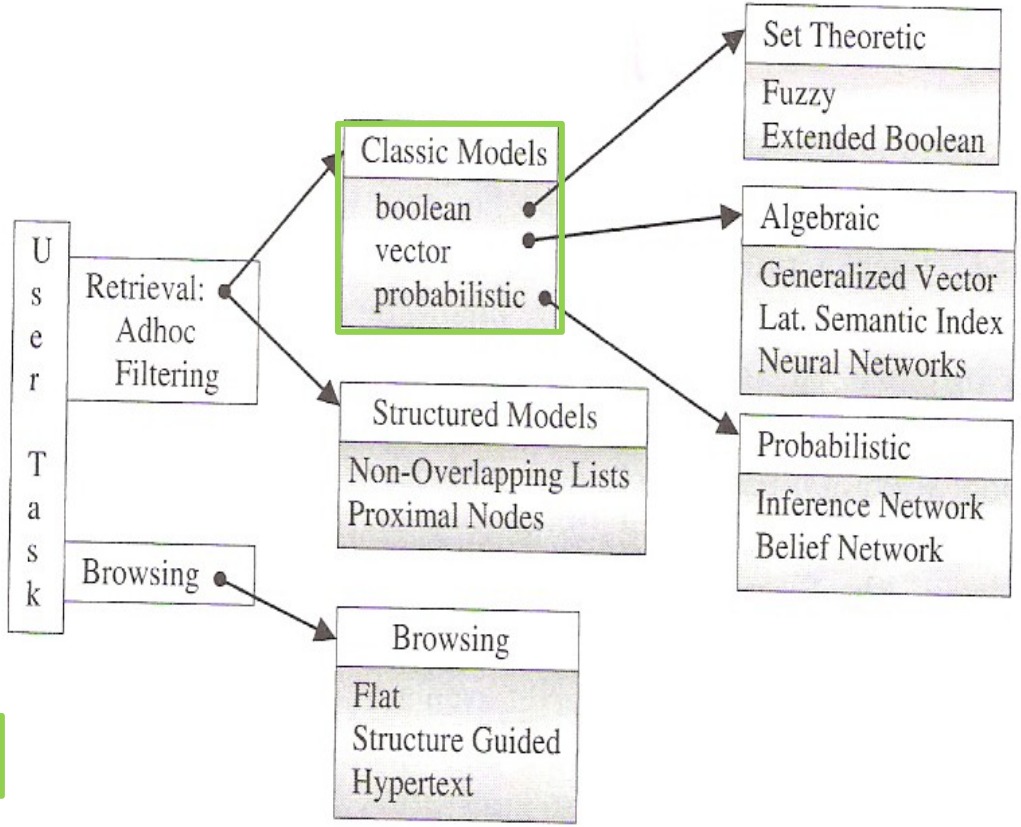
Information Retrieval Models

IR model taxonomy



Jarvelin, K. (2007)

Another taxonomy



Baeza-Yates andRibeiro-Neto (2004)

Information Retrieval

- Back to the MRS book
  - A simplified view of IR
- Manning, Raghavan, and Schütze (2008):
  - Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Basic assumptions of Information Retrieval

- Collection: Fixed set of documents
- Goal: Retrieve documents with information that is relevant to the user's information need and helps the user complete a task
  - Relevance is a key notion in IR but remains one of the LEAST understood concepts in the field.

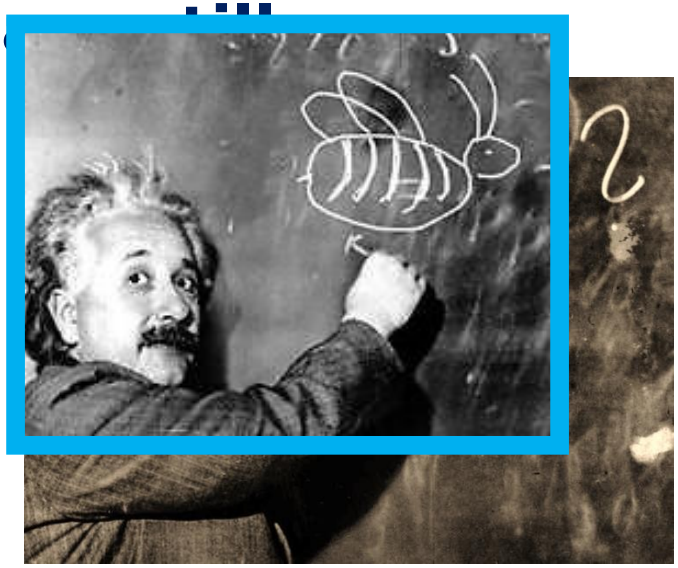
Put relevance aside

- Let's start discussions on retrieval algorithms with a basic searching (matching) problem without worrying about what exactly relevance is.

IR and Math

- Yes, there is math…
- But

"Do not worry about your difficulties in Mathematics. I can assure you mine [illegible] greater."



Part 2

- Boolean Model
- Document-term Matrix
- Inverted Index

Unstructured data in 1680

- Collection
  - Shakespeare's Collected Works
- Information need:
  - Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- One could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar***, then strip out lines containing ***Calpurnia***?
  - Sequential search
- Why is that not the answer?
  - Slow (for large corpora)

https://www.kaggle.com/kingburrito666/shakespeare-plays

Term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

1 if play contains word, 0 otherwise

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for ***Brutus***, ***Caesar*** and ***Calpurnia*** (complemented) Ⓟ bitwise *AND*.
- 110100 *AND* 110111 *AND* 101111 = 100100.

Answers to query

- Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius ***Caesar*** dead,
He cried almost to roaring; and he wept
When at Philippi he found ***Brutus*** slain.

- Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius ***Caesar*** I was killed i' the Capitol; ***Brutus*** killed me.

Bigger collections

- Consider *N* = 1 million documents, each with about 1000 words.
- Avg 6 bytes/word including spaces/punctuation
  - 6GB of data in the documents.
- Say there are *M* = 500K *distinct* terms among these.

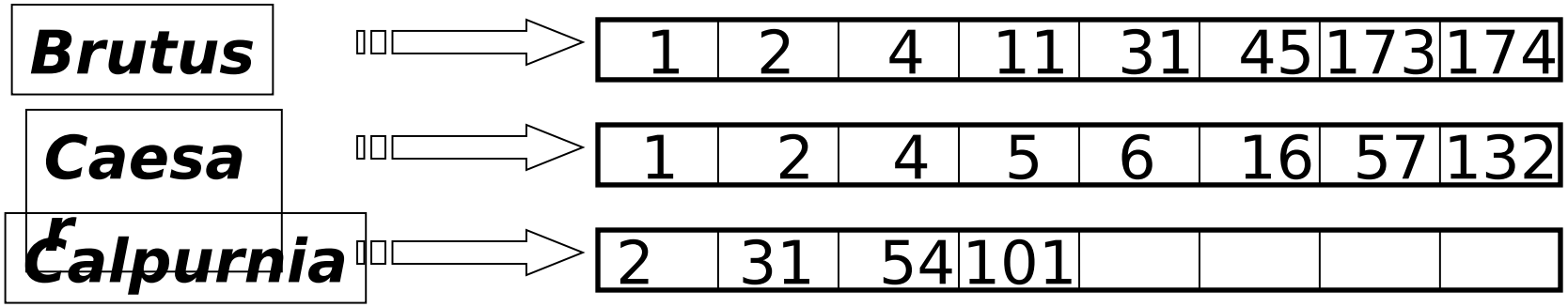
Can't build the matrix

- 500K x 1M matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's.
  - matrix is extremely sparse.
- What's a better representation?
  - We only record the 1 positions.

Why?

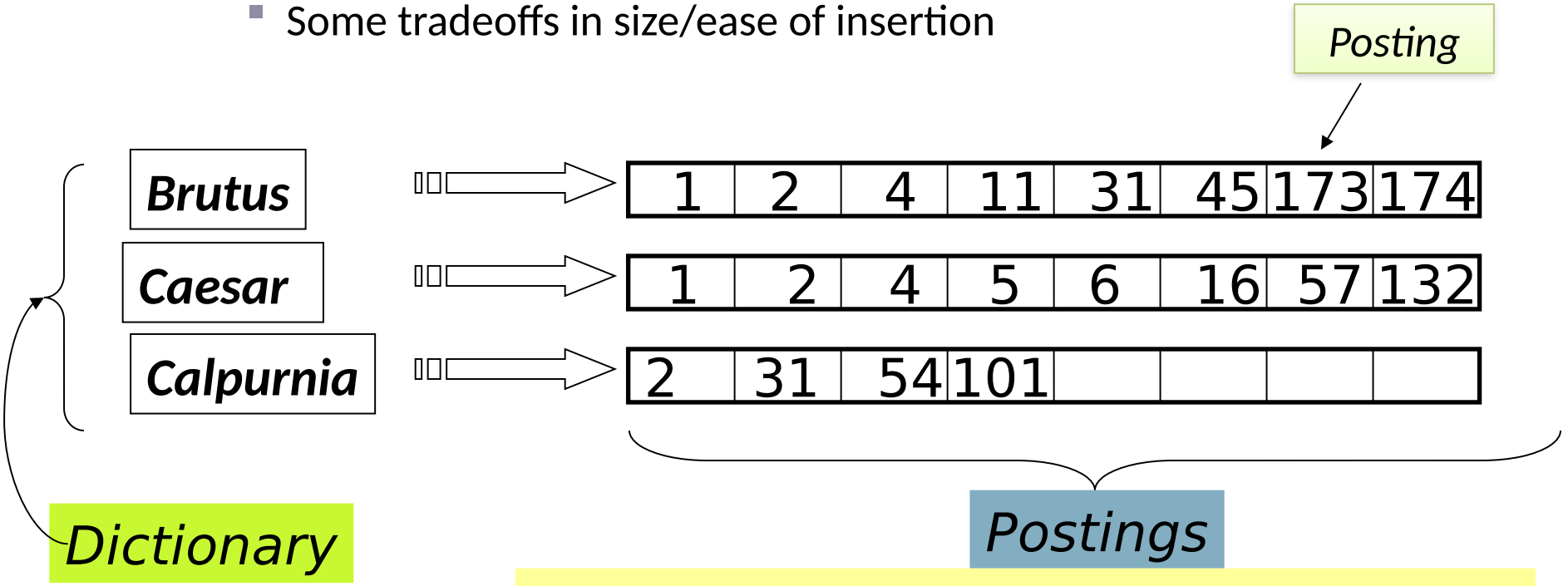
Inverted index

- For each term *t*, we must store a list of all documents that contain *t*.
  - Identify each by a **docID**, a document serial number



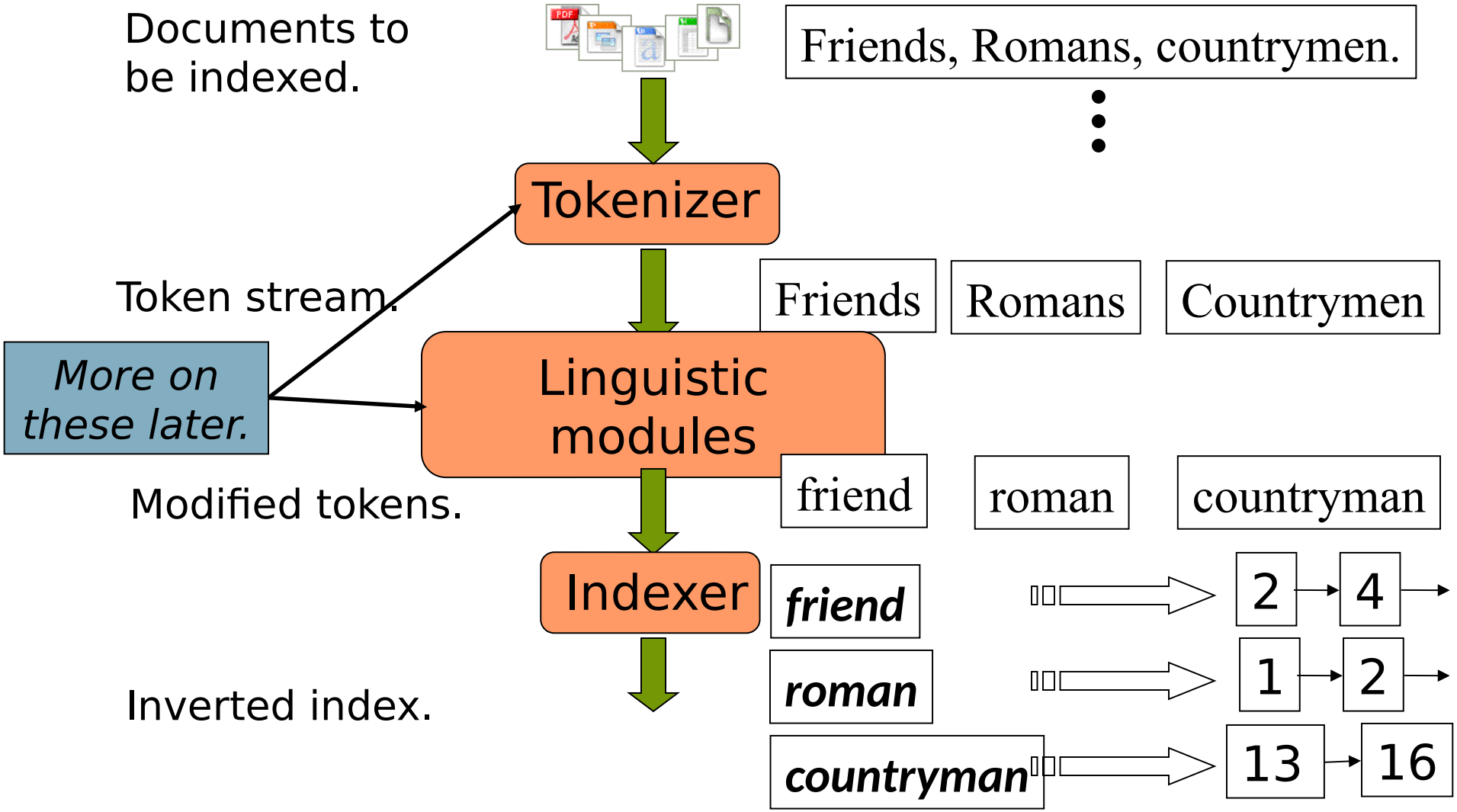
Inverted index

- We need variable-size postings lists
  - On disk, a continuous run of postings is normal and best
  - In memory, can use linked lists or variable length arrays
    - Some tradeoffs in size/ease of insertion



Sorted by docID (more later on why).

Inverted index construction



Indexer steps: Token sequence

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

Doc 1

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

Doc 2

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
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

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

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

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Exercise

- Given the following collection of documents
  - Doc 1: new home sales top forecasts
  - Doc 2: home sales rise in july
  - Doc 3: increase in home sales in july
  - Doc 4: july new home sale rise
- Tasks
  - Draw the inverted index
  - Draw the term-document incidence matrix

Results

Term	DF	Posting			
forecasts		1			
home		1	2	3	4
in		2	3		
increases		3			
july		2	3	4	
new		1			
rise		2	4		
sales		1	2	3	4
top		1			

Results with Document Freq. values

Term	DF	Posting			
forecasts	1	1			
home	4	1	2	3	4
in	2	2	3		
increases	1	3			
july	3	2	3	4	
new	1	1			
rise	2	2	4		
sales	4	1	2	3	4
top	1	1			

QUERY PROCESSING

The index we just built

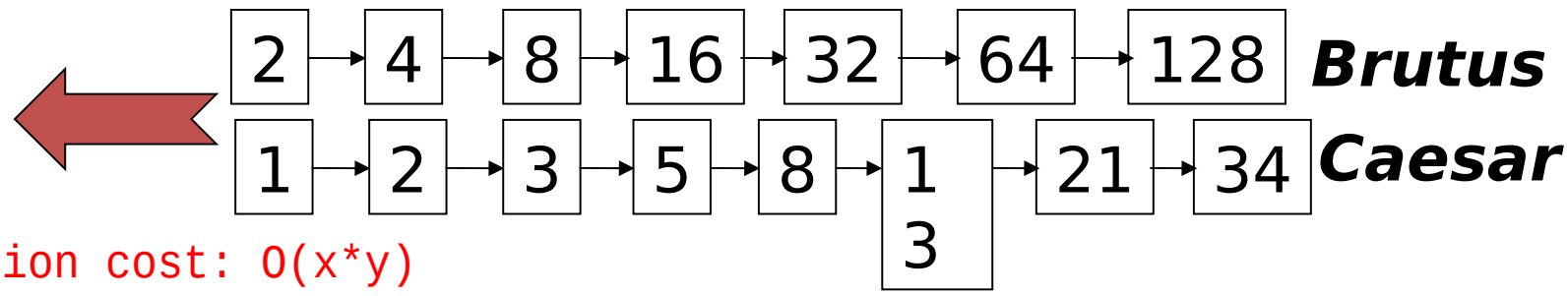
- How do we process a query?

Query processing: AND

- Consider processing the query:

  Brutus *AND* ***Caesar***

  - Locate ***Brutus*** in the Dictionary;
    - Retrieve its postings.
  - Locate *Caesar* in the Dictionary;
    - Retrieve its postings.
  - "Merge" the two postings:

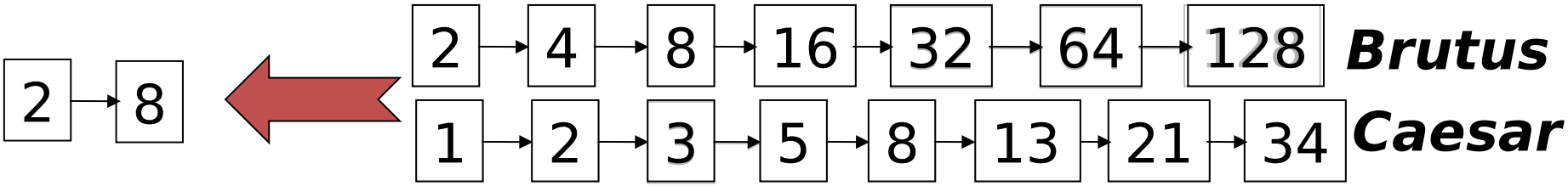


Computation cost: O(x*y)

The merge

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

Moving forward if one index is small than the other one.



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

Crucial: postings sorted by docID.

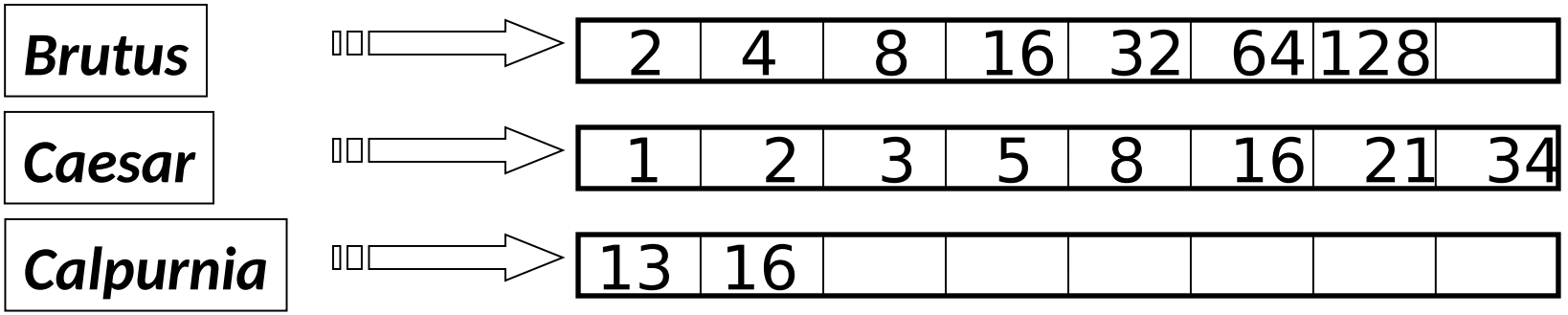
Further improvement: see txtbook.

Boolean queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries are queries using *AND*, *OR* and *NOT* to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not.
  - Perhaps the simplest model to build an IR system on
- Primary commercial retrieval tool for 3 decades.
- Many search systems you still use are Boolean:
  - Email, library catalog, Mac OS X Spotlight

Query optimization

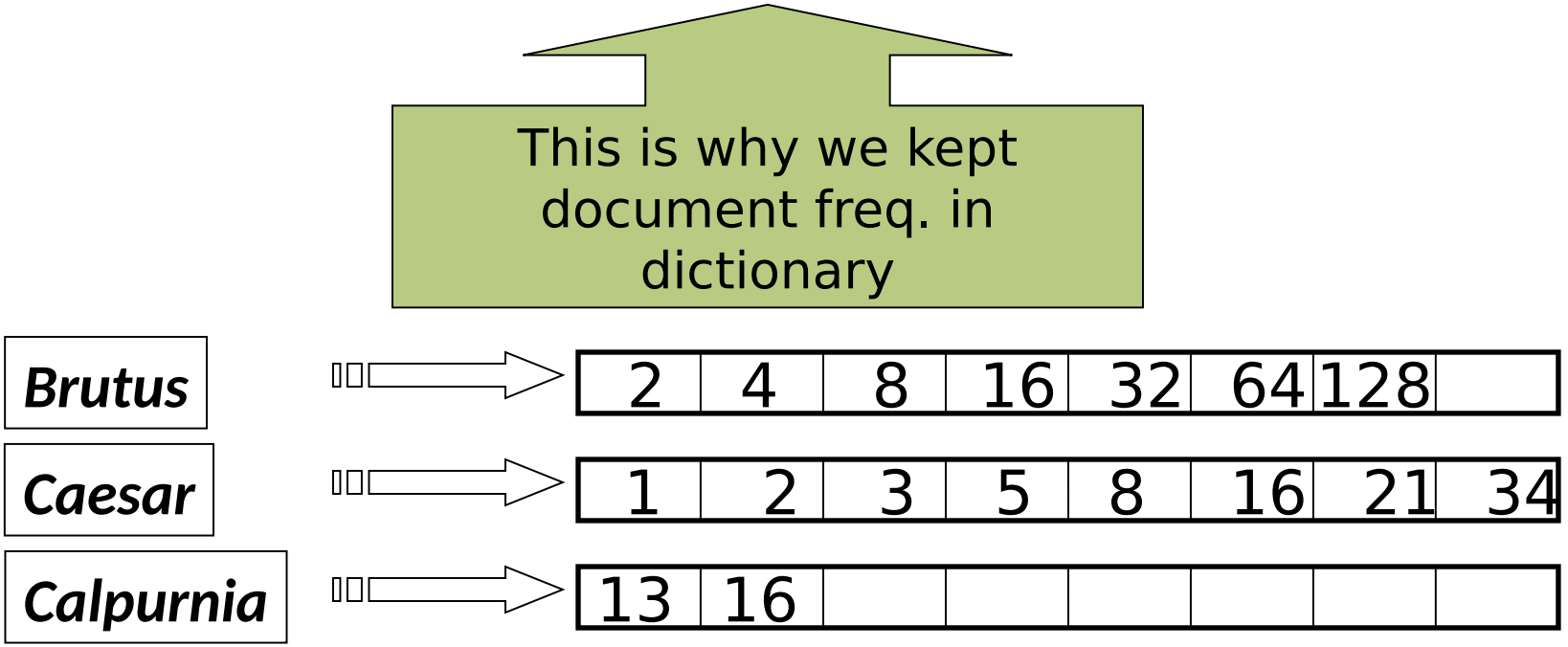
- What is the best order for query processing?
- Consider a query that is an *AND* of *n* terms.
- For each of the *n* terms, get its postings, then *AND* them together.



Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

Query optimization example

- Process in order of increasing freq:
  - *start with smallest set, then keep cutting further.*



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

More general optimization

- *(…) AND (…) AND (…)*
- Steps:
  - Get document frequencies for all terms in the query
  - Estimate document frequency for each component of AND (in parentheses)
  - Merge their postings in increasing order of document frequency
- Note: posting length == document frequency

Example

- Query for the exercise collection
  - “home” AND “increase” AND NOT “july”

Term	DF	Posting			
forecasts	1	1			
home	4	1	2	3	4
in	2	2	3		
increases	1	3			
july	3	2	3	4	
new	1	1			
rise	2	2	4		
sales	4	1	2	3	4
top	1	1			

Example (cont.)

- DF values of the terms
  - Home: 4
  - Increase: 1
  - July: 3
- There are 4 documents in the example collection
- So, DF values of the query components:
  - Home: 4
  - Increase: 1
  - NOT july: 1 (i.e., #total docs - #docs "july" appears = 4 – 3)
- Optimized query processing order: process (increase AND not july) first and then AND home
  - In the increasing order of document frequency (DFs)

Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Often we want to rank/group results
  - Need to measure proximity from query to each doc.
  - Need to decide whether docs presented to user are singletons, or a group of docs covering various aspects of the query.

More…

- *Introduction to Information Retrieval*, MRS chapter 1

Any questions?

You may email your instructor and/or post your questions to the discussion board.