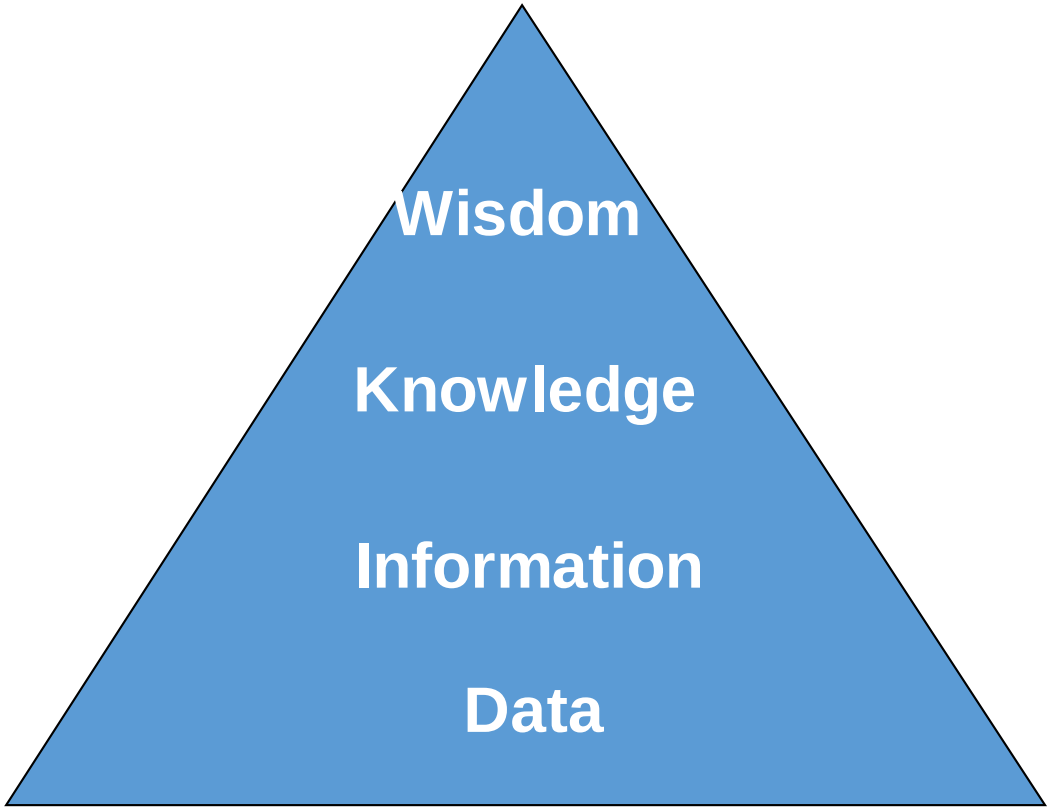
Biomedical Information Retrieval

Lecture 1: Introduction

Information ?

Information Hierarchy



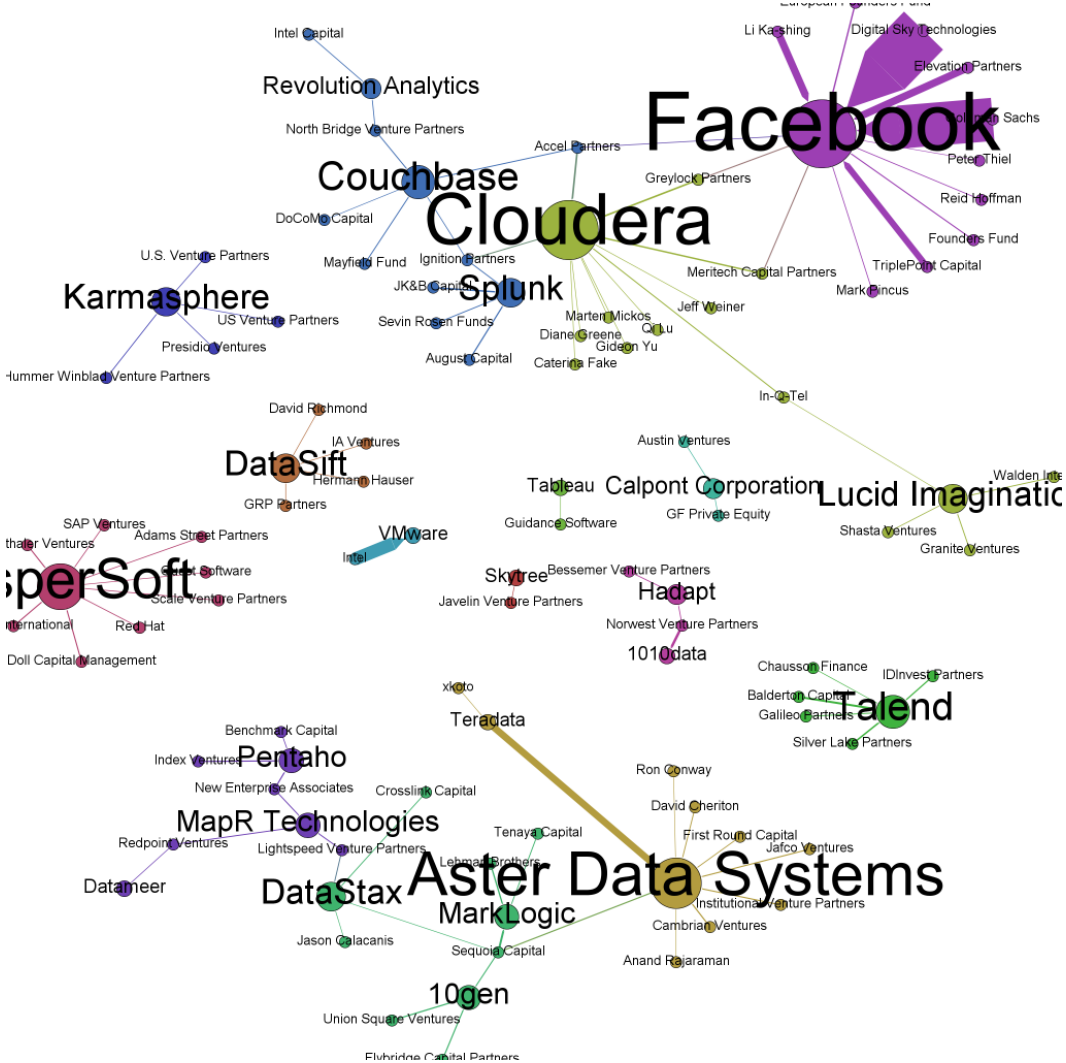
Information Hierarchy

- Data
  - The raw material of information
- Information
  - Data organized and presented by someone
- Knowledge
  - Information read, heard or seen and understood
- Wisdom
  - Distilled and integrated knowledge and understanding

What kinds of information are there?

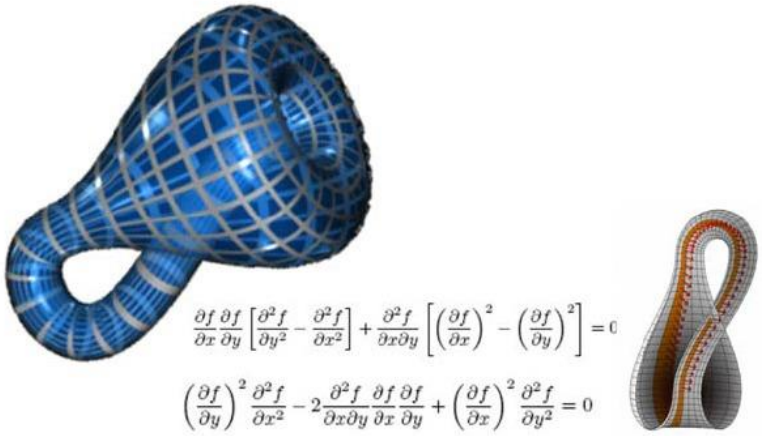
- Text
  - books, periodicals, WWW, memos, ads
  - published/refeered
- Film
- Photos, other Images
- Broadcast TV, Radio
- Telephone Conversations
- Databases…
- Currently, Internet data is the major player

Big Data !!

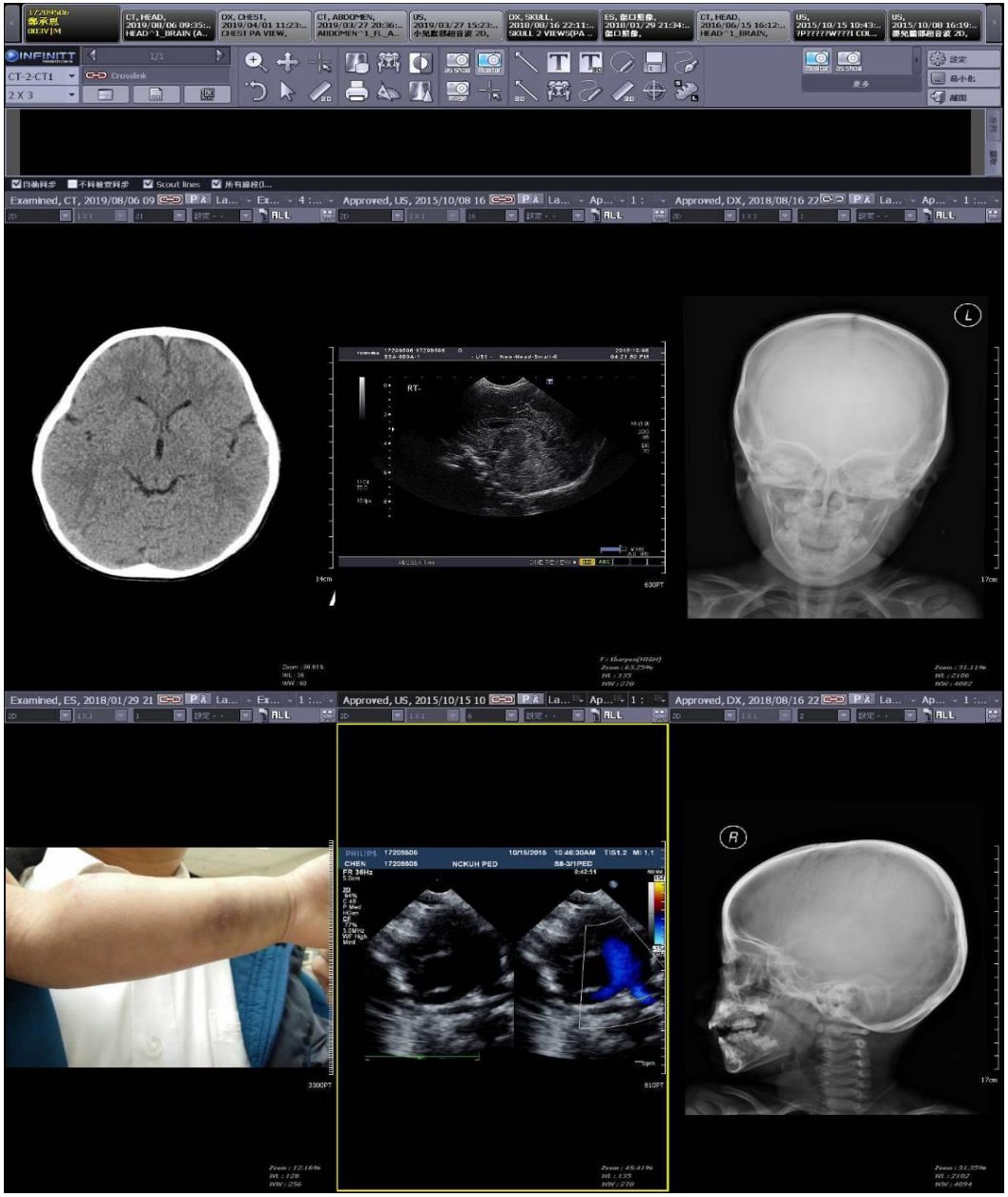

Intel Capital
Revolution Analytics
North Bridge Venture Partners
Couchbase
DoCoMo Capital
Mayfield Fund
Ignition Partners
Accel Partners
Facebook
Li Ka-shing
Digital Sky Technologies
Elevation Partners
Goldman Sachs
Peter Thiel
Greylock Partners
Reid Hoffman
Cloudera
Founders Fund
Meritech Capital Partners
TriplePoint Capital
Mark Pincus
U.S. Venture Partners
Karmasphere
US Venture Partners
Presidio Ventures
Hummer Winblad Venture Partners
JK&B Capital
Splunk
Sevin Rosen Funds
August Capital
Jeff Weiner
Marten Mickos
Diane Greene
Qi Lu
Gideon Yu
Caterina Fake
In-Q-Tel
David Richmond
IA Ventures
DataSift
Herman Hauser
GRP Partners
Austin Ventures
Tableau
Calpont Corporation
GF Private Equity
Guidance Software
Lucid Imaginatic
Shasta Ventures
Granite Ventures
SAP Ventures
Adams Street Partners
VMware
Intel
SperSoft
Red Hat
Doll Capital Management
Skytree
Bessemer Venture Partners
Javelin Venture Partners
Hadapt
Norwest Venture Partners
1010data
Chausson Finance
IDInvest Partners
Balderton Capital
Galileo Partners
Talend
Silver Lake Partners
Teradata
Benchmark Capital
Index Ventures
Pentaho
New Enterprise Associates
Crosslink Capital
MapR Technologies
Redpoint Ventures
Lightspeed Venture Partners
Datameer
Tenaya Capital
Lehman Brothers
DataStax
Aster Data Systems
MarkLogic
Jason Calacanis
Sequoia Capital
Ron Conway
David Cheriton
First Round Capital
Jafco Ventures
Institutional Venture Partners
Cambrian Ventures
Anand Rajaraman
10gen
Union Square Ventures

Big Data !!

User Activity on Wikipedia



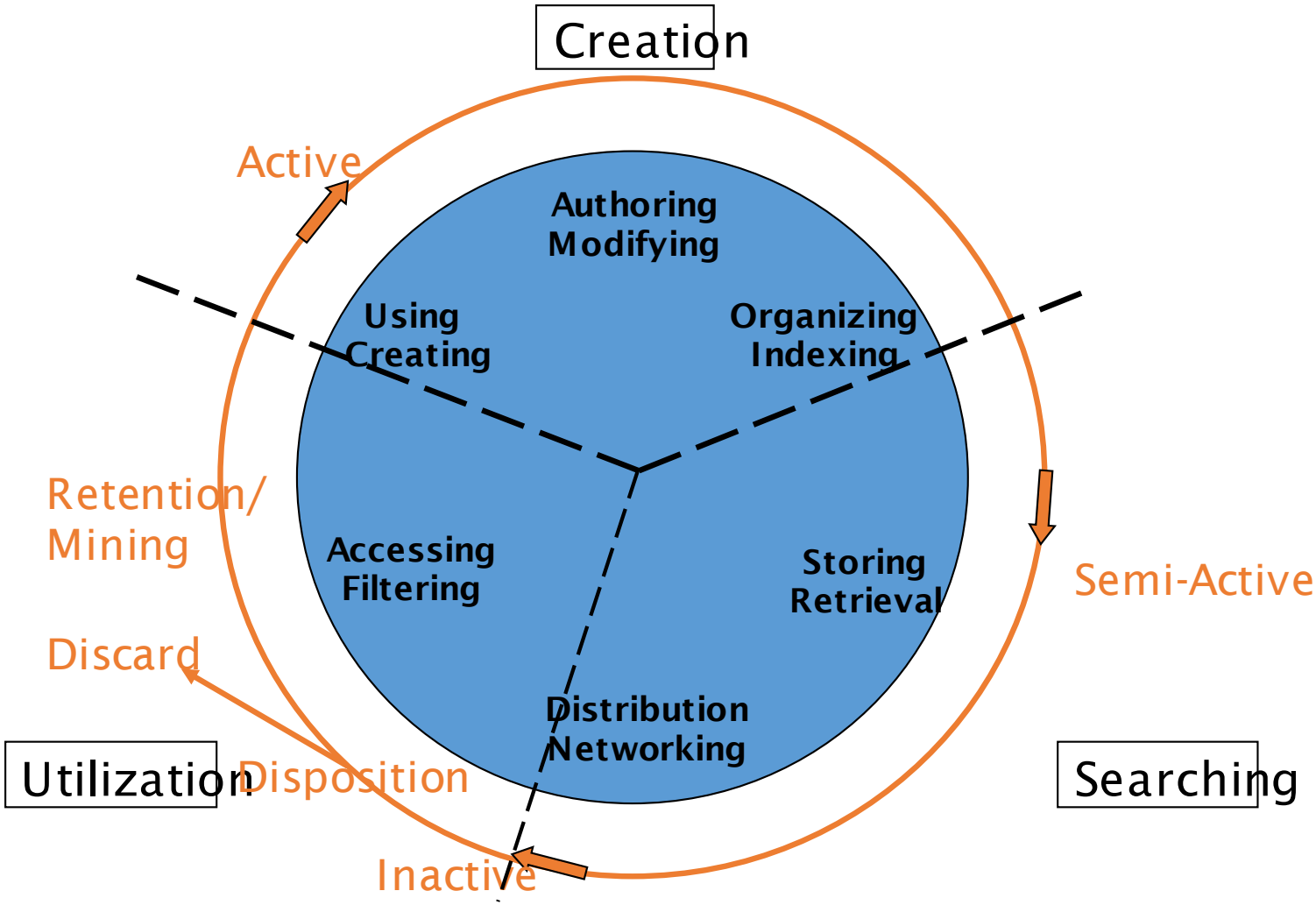
DARPA's Topological Data Analysis Program


INFINITT
CT-2-CT1
Crosslink
2 X 3
Examined, CT, 2019/08/06 09
Approved, US, 2015/10/08 16
Approved, DX, 2018/08/16 22
Examined, ES, 2018/01/29 21
Approved, US, 2015/10/15 10
Approved, DX, 2018/08/16 22
RT-
PHILIPS
CHEN
NCKUH PED
L
R

Counting the Library of Congress

- Library of Congress
  - Books: 20 Terabytes assuming
    - 22M books (22,765,967 catalogued books, 109,029,796 items)
    - 1 MB per book
  - Should also assume
    - 13M photographs, 1MB each = 13 TB
    - 4M maps, say 200 TB
    - 500K files, 1GB each = 500 TB
    - 3.5M sound recordings, ~2000 TB
- Grand total: 3 petabytes (3000 terabytes)

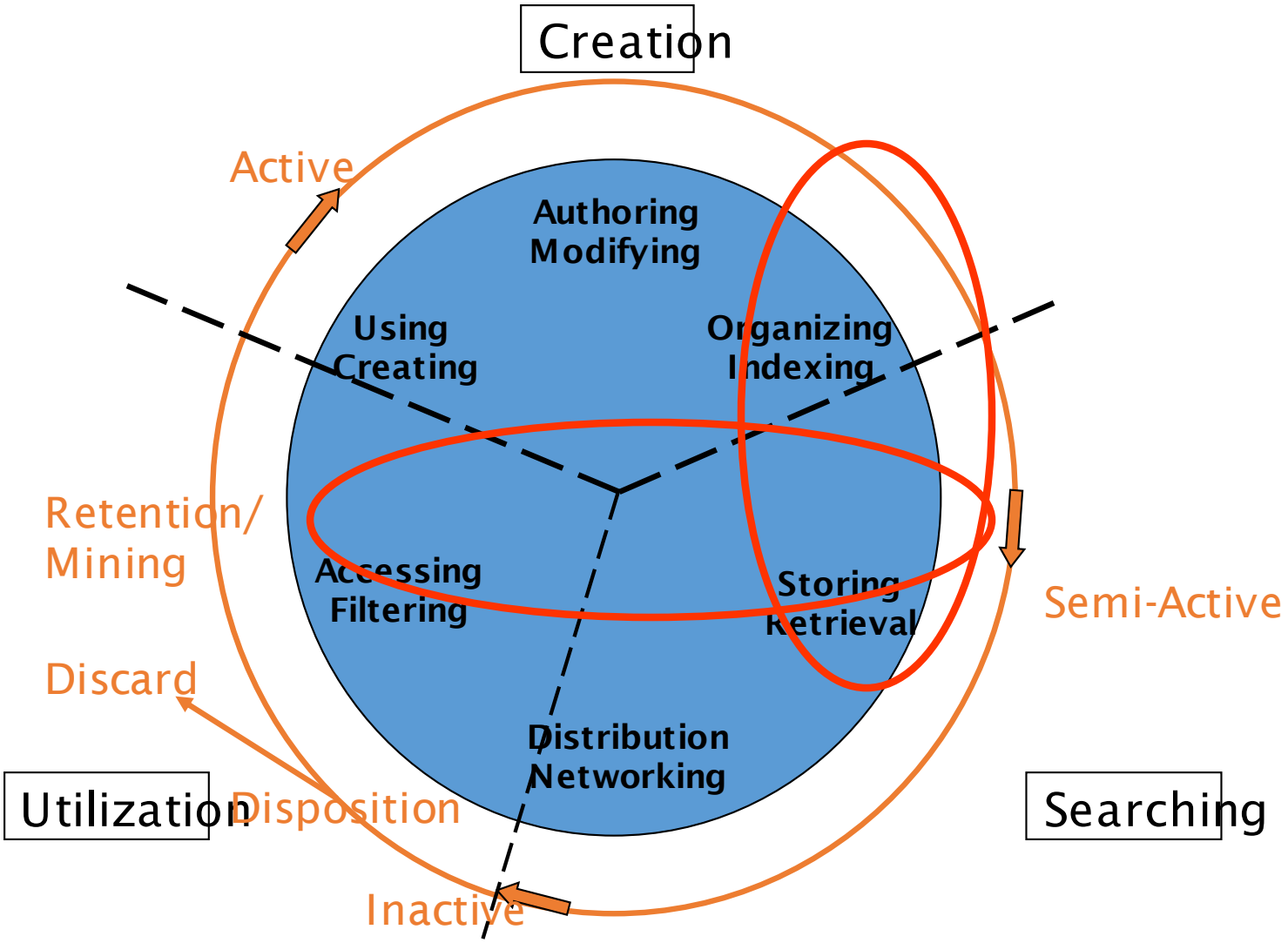
Information Life Cycle



Issues in Information

- Information Storage
  - How and Where is Information stored?
- Retrieving Information.
  - How is information recovered from storage
  - How to find needed information
  - Linked with Accessing/Filtering stage

Key Issues


Creation
Active
Authoring
Modifying
Using
Creating
Organizing
Indexing
Retention/
Mining
Accessing
Filtering
Storing
Retrieval
Semi-Active
Discard
Distribution
Networking
Utilization
Disposition
Searching
Inactive

Information Retrieval ?

Some IR History

- Roots in the scientific "Information Explosion" following WWII
- Interest in computer-based IR from mid 1950's
  - H.P. Luhn at IBM (1958)
  - Probabilistic models at Rand (Maron & Kuhns) (1960)
  - Boolean system development at Lockheed ('60s)
  - Vector Space Model (Salton at Cornell 1965)
  - Statistical Weighting methods and theoretical advances ('70s)
  - Refinements and Advances in application ('80s)
  - User Interfaces, Large-scale testing and application ('90s)

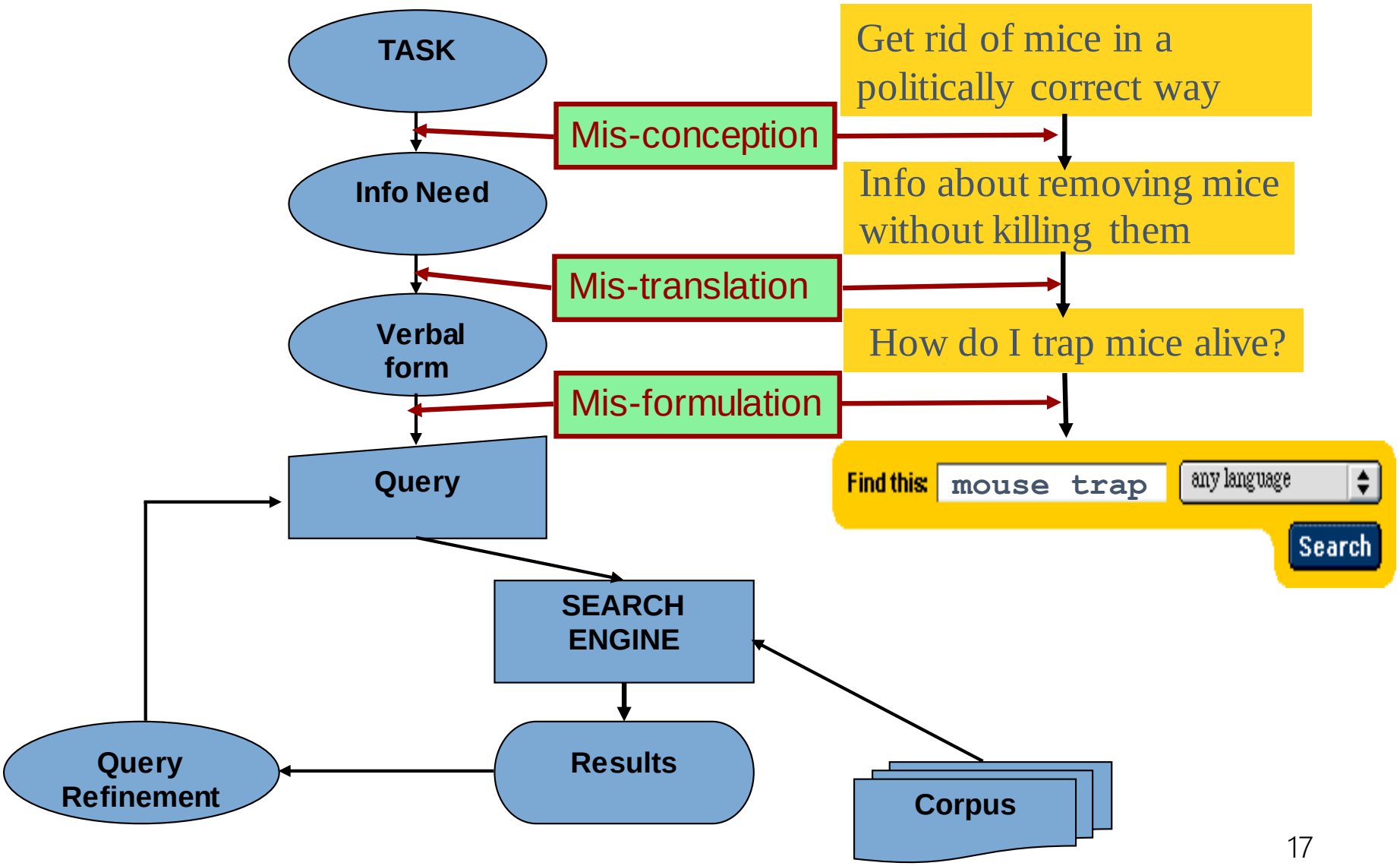
Search and Retrieval

- Human Aspects
- Information Retrieval Models
- Content Analysis/Zipf Distributions
- Evaluation of IR Systems
  - Precision/Recall
  - Relevance
  - User Studies
- System and Implementation Issues
- Web-Specific Issues
- User Interface Issues
- Special Kinds of Search

Basic assumptions of IR

- Collection: Fixed set of documents
- Goal: Retrieve documents with information that is relevant to user's information need and helps him (or her) complete a task

The classic search model



How good are the retrieved docs?

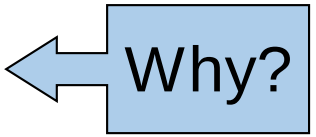
- *Precision* : Fraction of retrieved docs that are relevant to user's information need
- *Recall* : Fraction of relevant docs in collection that are retrieved
- More precise definitions and measurements to follow in later lectures

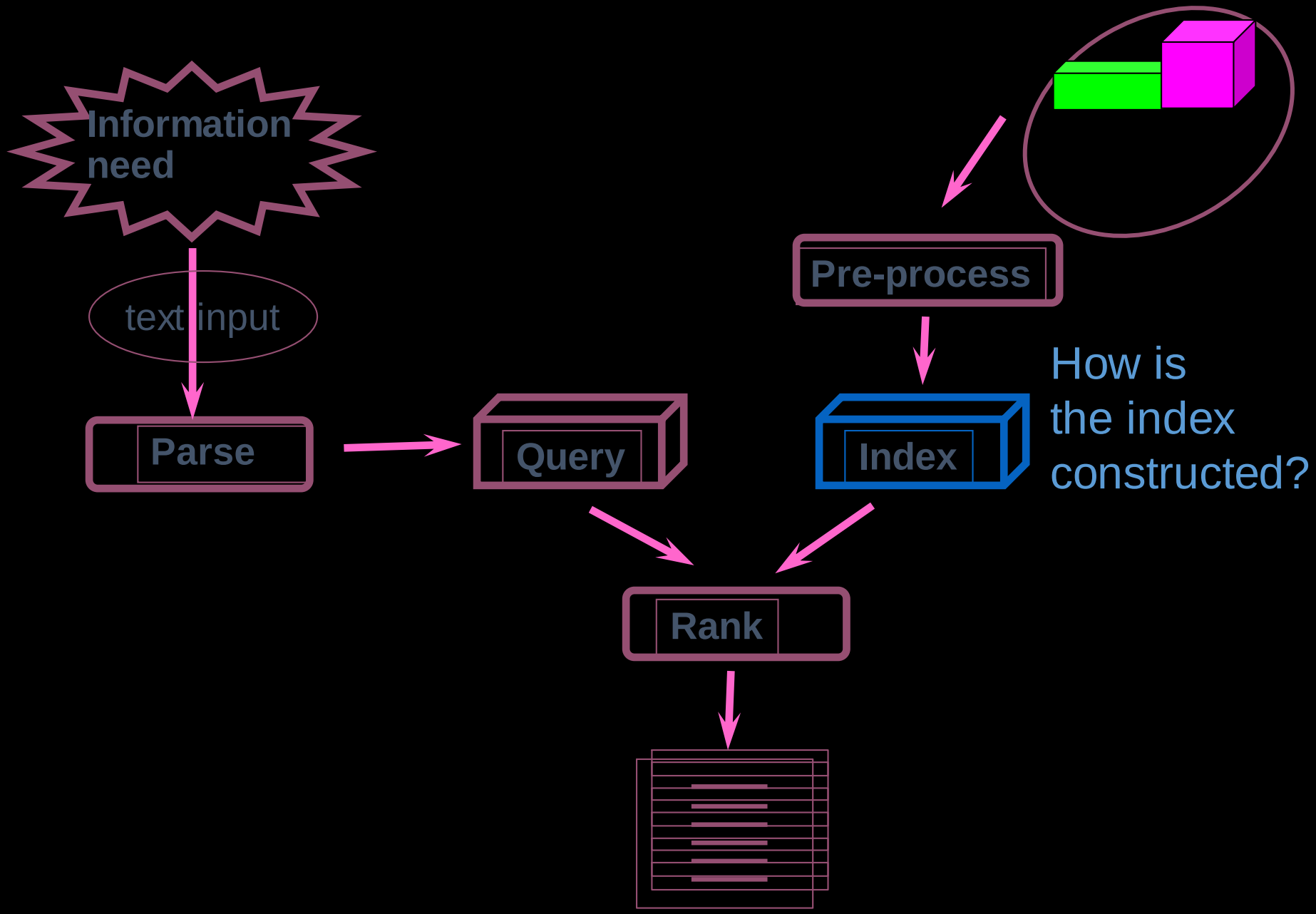
Bigger collections

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

Can’t build the matrix

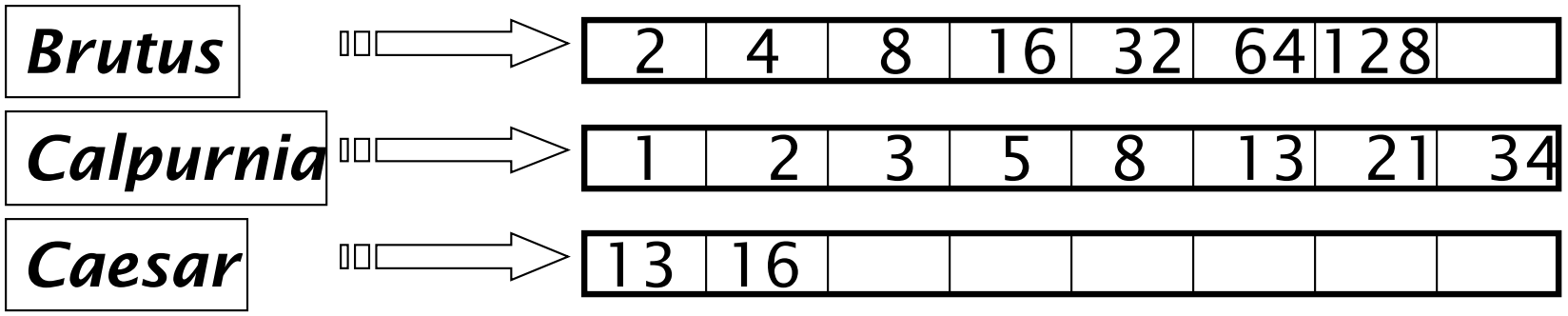
- 500K x 1M matrix has half-a-trillion 0’s and 1’s.
- But it has no more than one billion 1’s.
  - matrix is extremely sparse.
- What’s a better representation?
  - We only record the 1 positions.




Information need
text input
Parse
Query
Pre-process
Index
How is the index constructed?
Rank

Inverted index

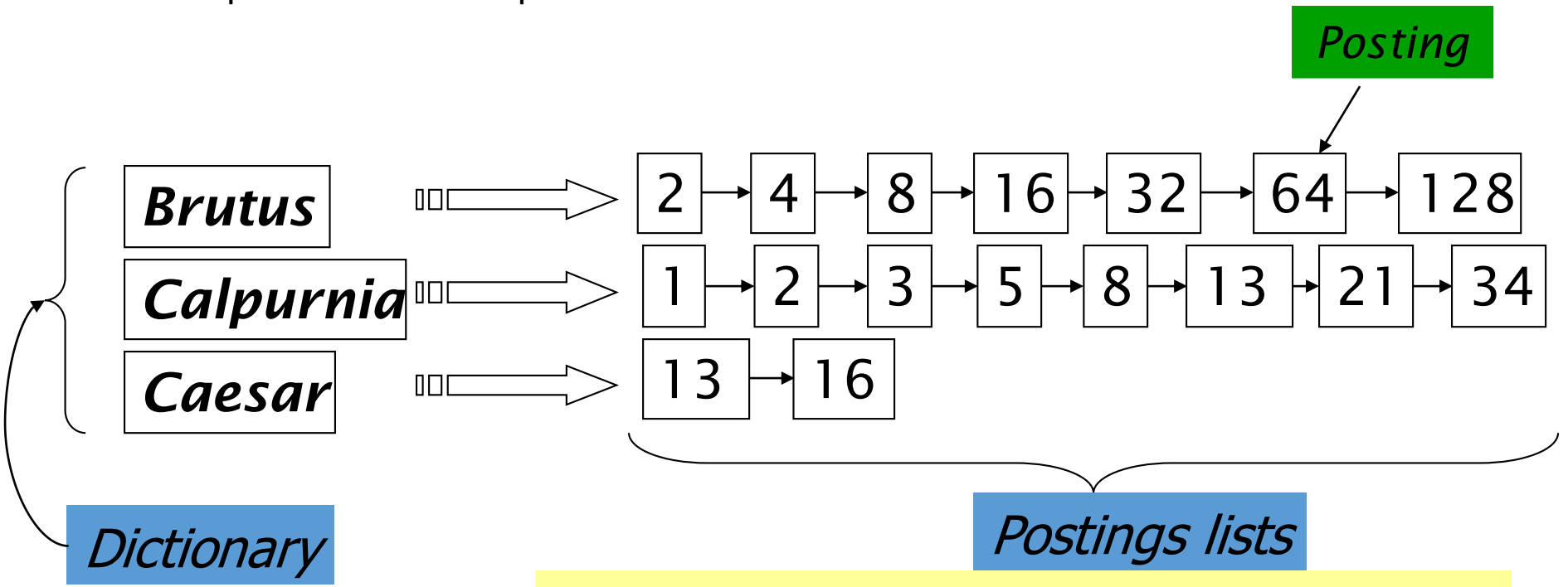
- For each term *T*, we must store a list of all documents that contain *T*.
- Do we use an array or a list for this?



What happens if the word ***Caesar*** is added to document 14?

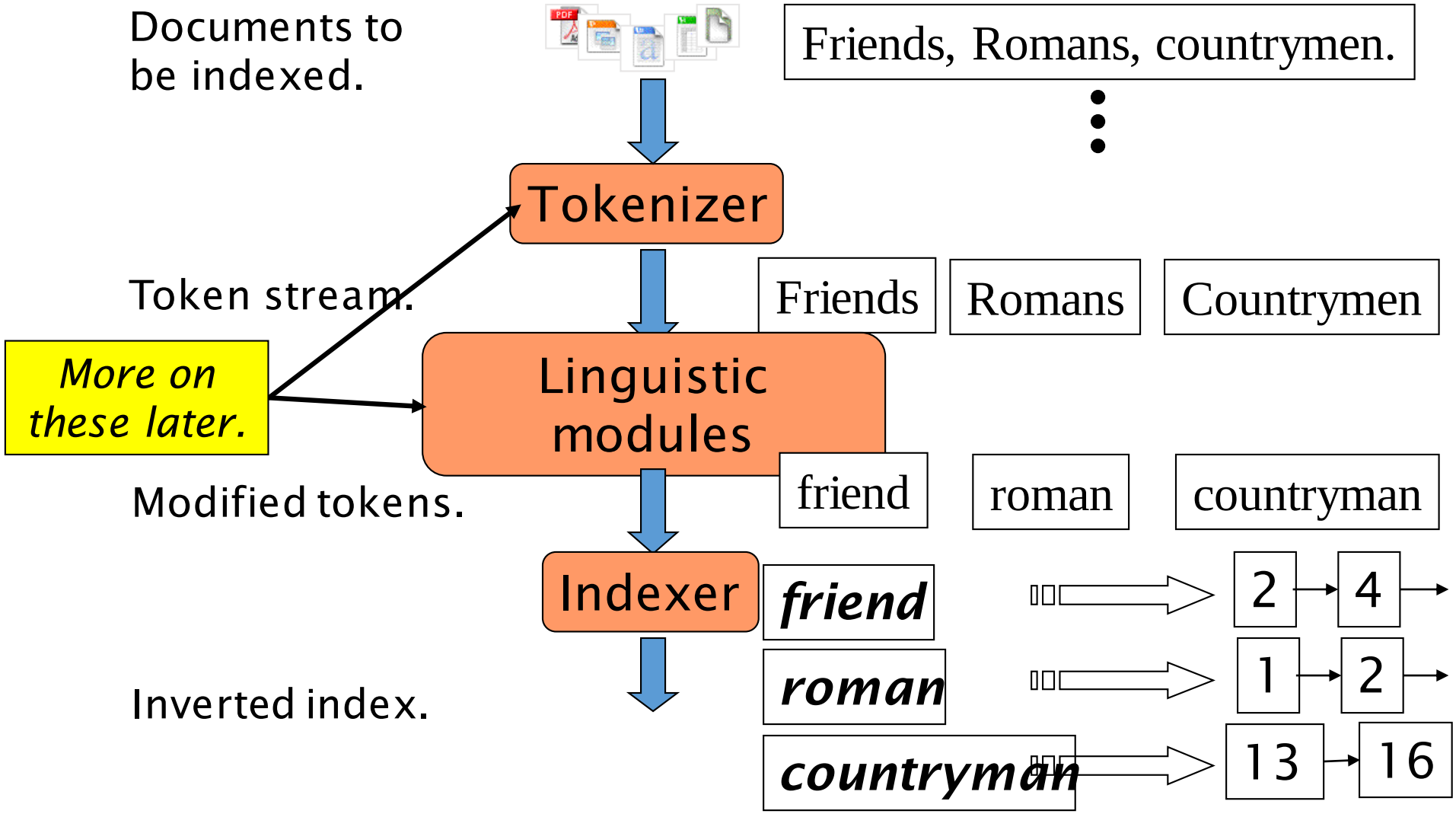
Inverted index

- Linked lists generally preferred to arrays
  - Dynamic space allocation
  - Insertion of terms into documents easy
  - Space overhead of pointers



Sorted by docID (more later on why).

Inverted index construction



Indexer steps

- 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	Doc #
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

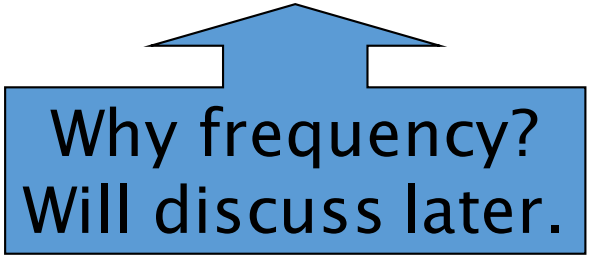
- Sort by terms.

Core indexing step.

Term	Doc #
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	Doc #
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

- Multiple term entries in a single document are merged.
- Frequency information is added.

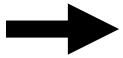


Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

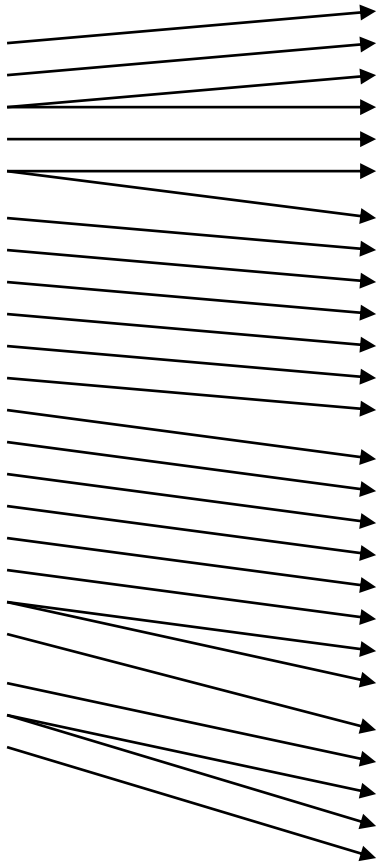
Term	Doc #	Term freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

- The result is split into a *Dictionary* file and a *Postings* file.

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

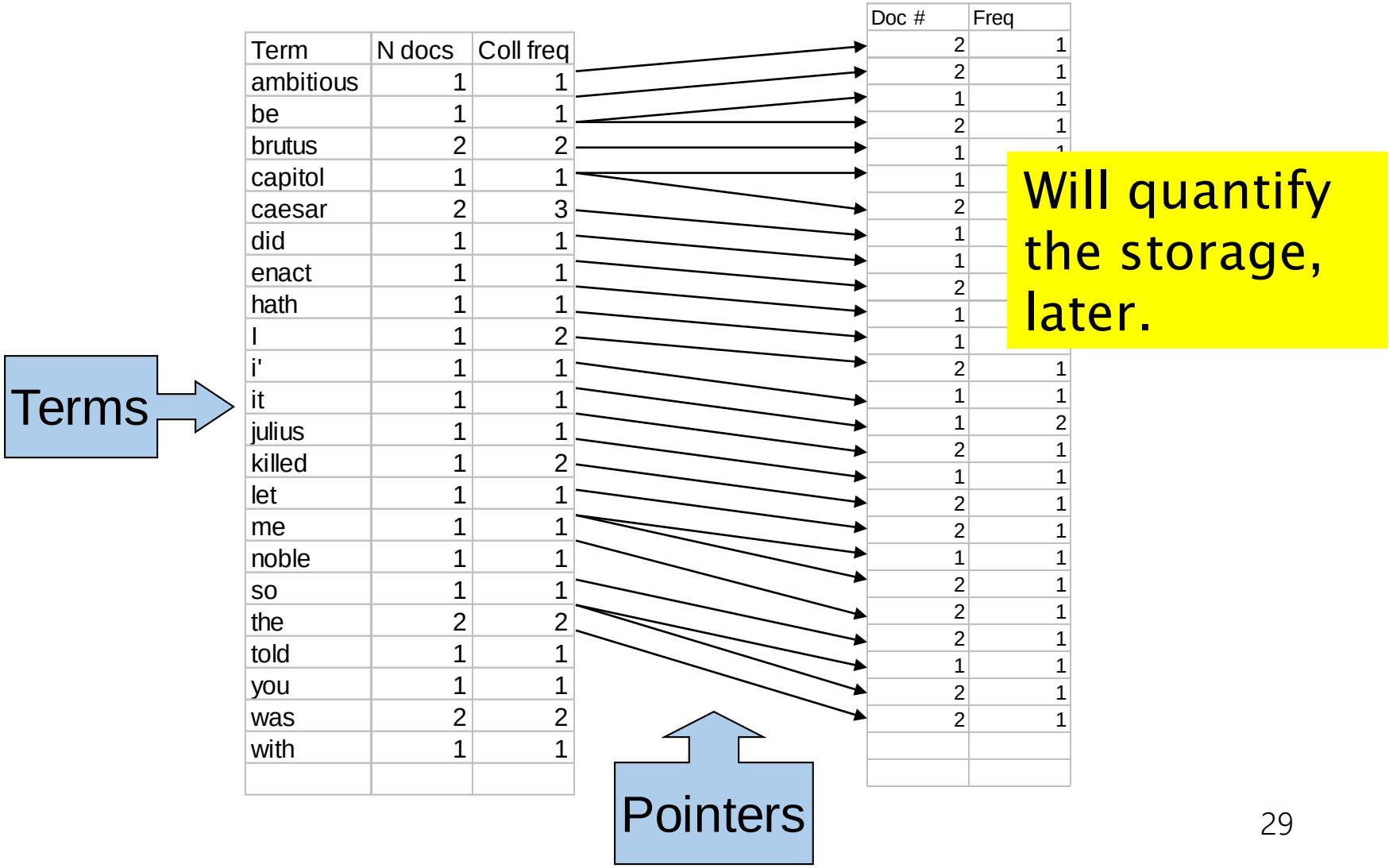


Term	N docs	Coll freq
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
i'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	2	2
told	1	1
you	1	1
was	2	2
with	1	1



Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

•Where do we pay in storage?

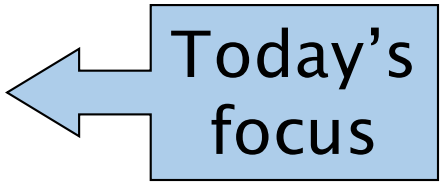


Term	N docs	Coll freq
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
i'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	2	2
told	1	1
you	1	1
was	2	2
with	1	1

Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	1
1	1
1	1
2	1
1	1
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

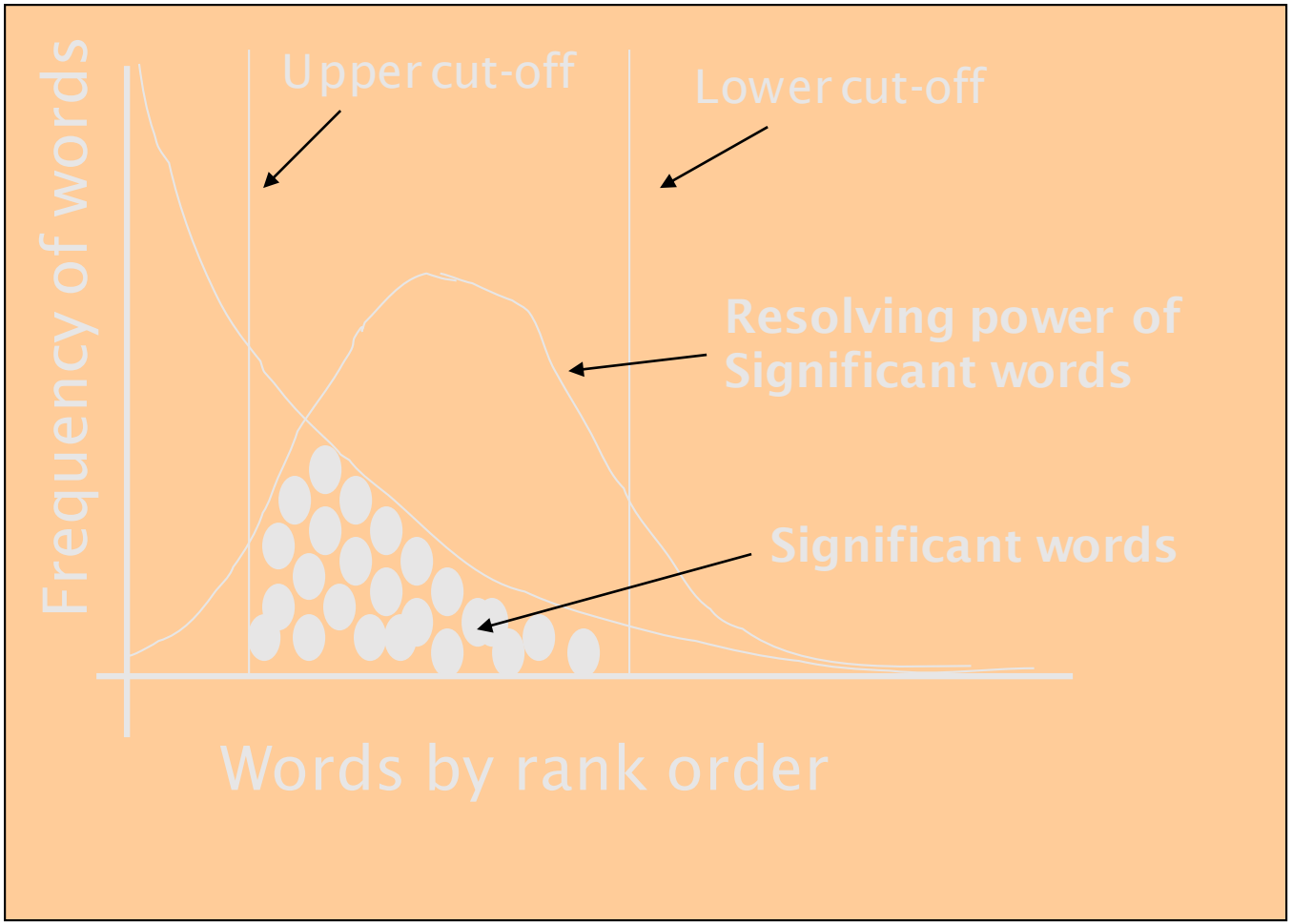
The index we just built

- How do we process a query?
  - Later - what kinds of queries can we process?



Word Frequency vs. Resolving Power (from van Rijsbergen 79)

The most frequent words are *not* the most descriptive

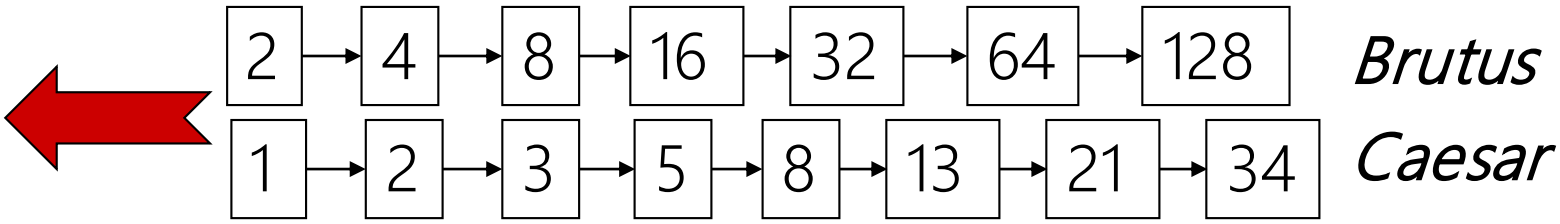


Query processing: AND

- Consider processing the query:

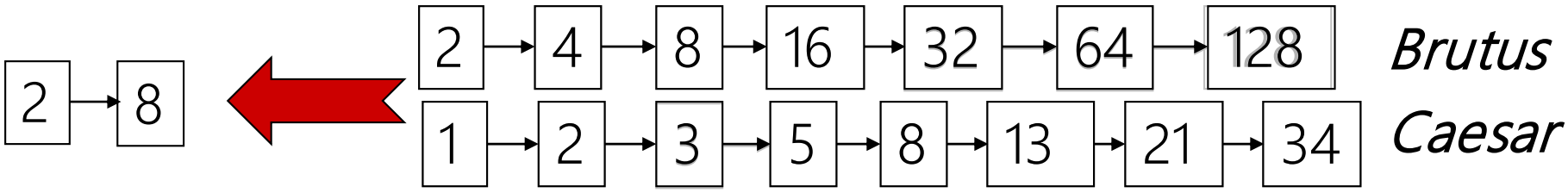
  Brutus *AND* ***Caesar***

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



The merge

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



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

Crucial: postings sorted by docID.

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.
- Primary commercial retrieval tool for 3 decades.
- Professional searchers (e.g., lawyers) still like Boolean queries:
  - You know exactly what you're getting.

Boolean queries:
More general merges

- Exercise: Adapt the merge for the queries:

  Brutus *AND NOT* ***Caesar***

  Brutus *OR NOT* ***Caesar***

Can we still run through the merge in time O(*x*+*y*)?

What can we achieve?

Merging

What about an arbitrary Boolean formula?

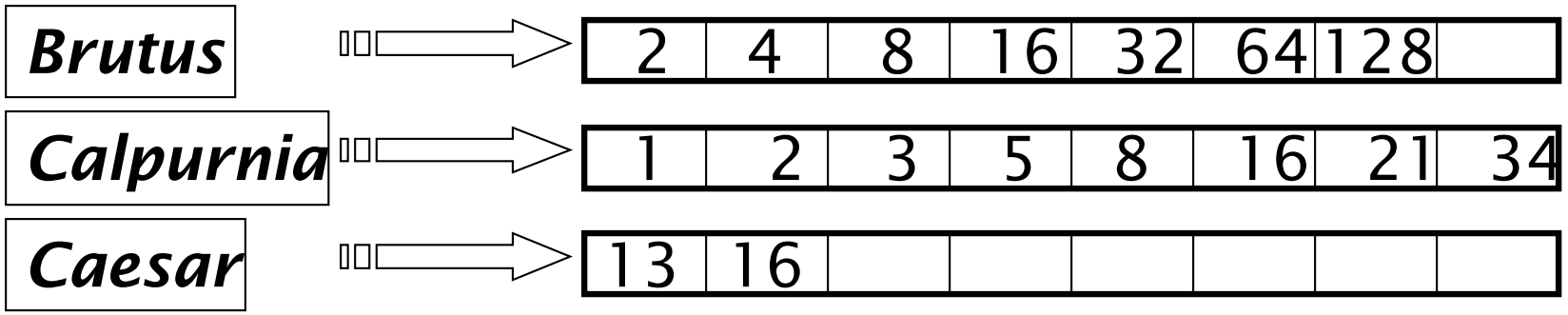
***(Brutus** OR **Caesar)** AND NOT*

(Antony** OR **Cleopatra)

- Can we always merge in “linear” time?
  - Linear in what?
- Can we do better?

Query optimization

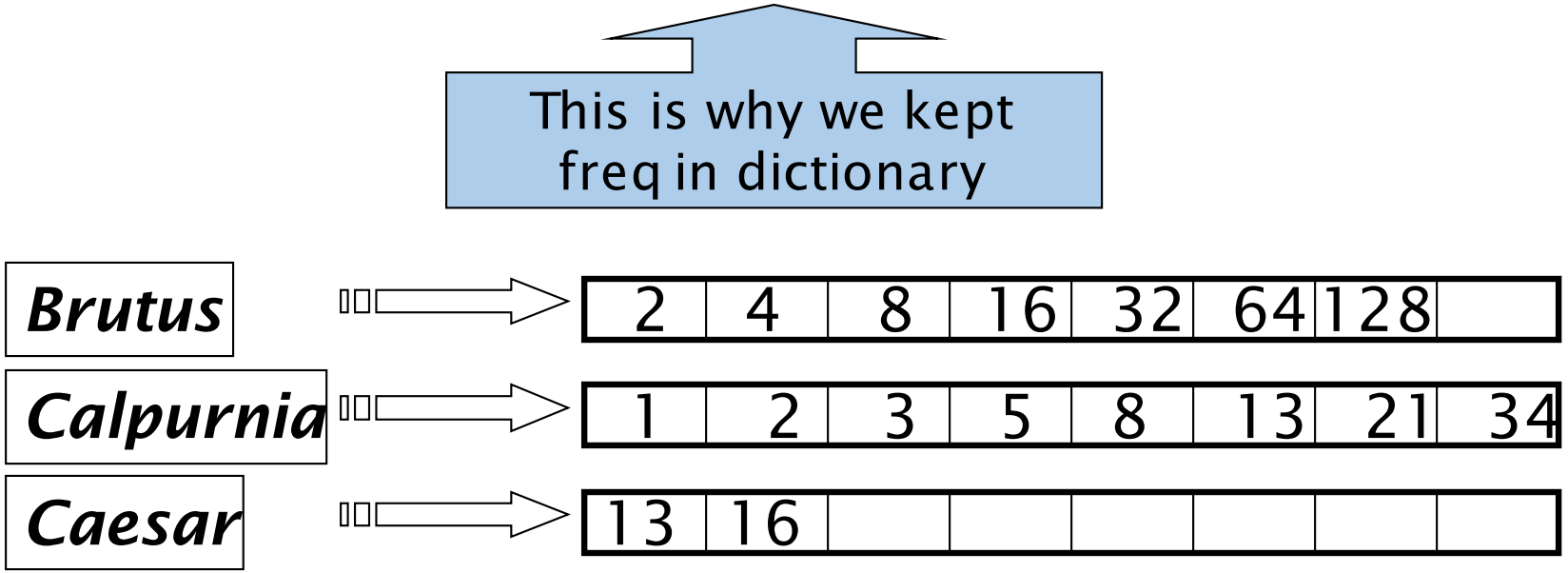
- What is the best order for query processing?
- Consider a query that is an *AND* of *t* terms.
- For each of the *t* terms, get its postings, then *AND* them together.



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 (*Caesar AND Brutus)* AND *Calpurnia*.

More general optimization

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

Exercise

- Recommend a query processing order for

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

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

Query processing exercises

- If the query is ***friends*** *AND* ***romans*** *AND (NOT* ***countrymen****),* how could we use the freq of ***countrymen***?
- Exercise: Extend the merge to an arbitrary Boolean query. Can we always guarantee execution in time linear in the total postings size?
- Hint: Begin with the case of a Boolean *formula* query: in this, each query term appears only once in the query.

What's ahead in IR? Beyond term search

- What about phrases?
  - ***Stanford University***
- Proximity: Find ***Gates*** *NEAR* ***Microsoft***.
  - Need index to capture position information in docs. More later.
- Zones in documents: Find documents with (*author =* ***Ullman***) *AND* (text contains ***automata***).

Evidence accumulation

- 1 vs. 0 occurrence of a search term
  - 2 vs. 1 occurrence
  - 3 vs. 2 occurrences, etc.
  - Usually more seems better
- Need term frequency information in docs

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.

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 text
- Allows
  - Keyword queries including operators
  - More sophisticated "concept" queries e.g.,
    - find all web pages dealing with *drug abuse*
- Classic model for searching text documents

Semi-structured data

- In fact almost no data is “unstructured”
- E.g., this slide has distinctly identified zones such as the *Title* and *Bullets*
- Facilitates “semi-structured” search such as
  - *Title* contains data AND *Bullets* contain search

… to say nothing of linguistic structure

More sophisticated semi-structured search

- *Title* is about Object Oriented Programming AND *Author* something like stro*rup
- where * is the wild-card operator
- Issues:
  - how do you process "about"?
  - how do you rank results?
- The focus of XML search.

Clustering and classification

- Given a set of docs, group them into clusters based on their contents.
- Given a set of topics, plus a new doc *D*, decide which topic(s) *D* belongs to.

The web and its challenges

- Unusual and diverse documents
- Unusual and diverse users, queries, information needs
- Beyond terms, exploit ideas from social networks
  - link analysis, clickstreams ...

- How do search engines work? And how can we make them better?

More sophisticated *information* retrieval

- Cross-language information retrieval
- Question answering
- Document Summarization
- Text mining
- …