Lecture 1
Introduction

Some material is from:

Yannis Tzitzikas (UoC) slides, &

Class material of the two textbooks

Information Retrieval

Collection: Fixed set of documents (information items)

Goal: Retrieve documents with information that is relevant to user's information need and helps the user complete a task

SIGIR 2005

Information Retrieval

Information item:

Usually text (often with structure), but possibly also image, audio, video, etc.

Text items are often referred to as *documents*, and may be of different scope (book, article, paragraph, etc.).

Examples

IR Systems
- Verity, Fulcrum, Excalibur, Eurospider
- Hummingbird, Documentum
- Inquery, Smart, Okapi, Lemur, Indri

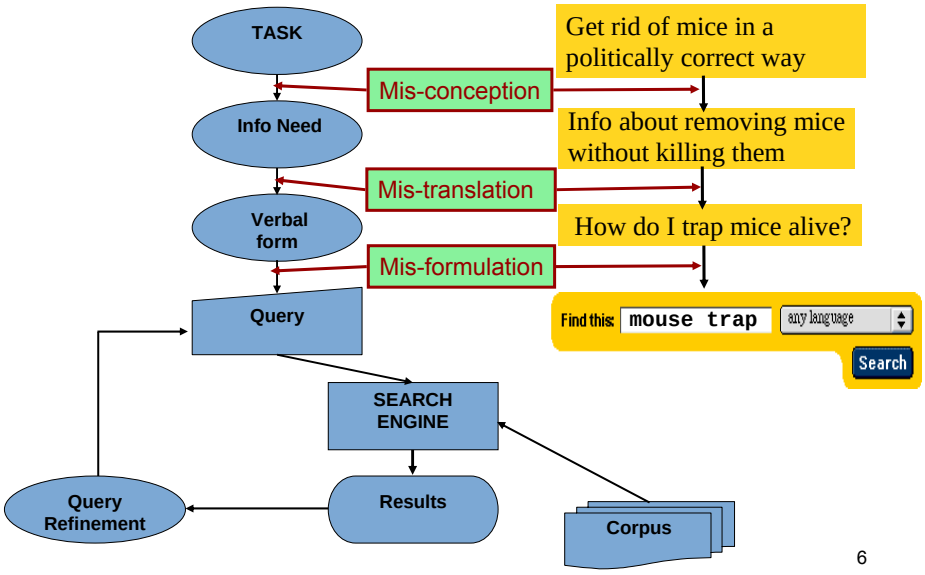
Web search and in-house systems
- West, LEXIS/NEXIS, Dialog
- Lycos, AltaVista, Excite, Yahoo, Google, Nothern Light, Teoma, HotBot,Direct Hit, …
- Ask Jeeves
- eLibrary, Inquira
- vivisimo (www.vivisimo.com)
- …

IR

Emphasis on User information need:

- Find all docs containing information on college tennis teams which: (1) are maintained by a USA university and (2) participate in the NCAA tournament.
- Translate this to a query (natural language, keyword, proximity, XQuery, sketch-based, etc)

The classic search model



IR

IR:

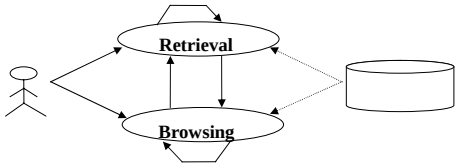
- representation,
- storage,
- organization of, and
- access to information items

- Emphasis is on the retrieval of information (not data)

Data vs Information Retrieval

- **Data retrieval**
  - which docs contain a set of keywords?
  - Well defined semantics
  - a single erroneous object implies failure! (sound and complete)
- **Information retrieval**
  - information about a subject or topic
  - semantics is frequently loose
  - small errors are tolerated
- **IR system:**
  - interpret content of information items
  - generate a *ranking* which reflects relevance
  - notion of *relevance* is most important

Basic Concepts: User Task



- Two complementary forms of information or data retrieval:

  Retrieval (ανάκτηση)

  Browsing (πλοήγηση)

Querying (retrieval) vs. Browsing

Querying:

- Information need (retrieval goal) is focused and crystallized.
- Contents of repository are well-known.
- Often, user is sophisticated.

Browsing:

- Information need (retrieval goal) is vague and imprecise (or there is no goal!)
- Contents of repository are not well-known.
- Often, user is naive.

Querying(retrieval) vs. Browsing (cont.)

- Flat (list of documents)
- Structure guided (hierarchical structure: file folders – yahoo! Directory, ODP)
  - also, inside a document (abstract, sessions, etc)
- Hypertext (following links)

Querying(retrieval) vs. Browsing (cont.)

- Querying and browsing are often interleaved (in the same session).
  - Example: present a query to a search engine, browse in the results, restate the original query, etc.

Pulling (ad hoc querying) vs. Pushing (filtering) information

- Querying and browsing are both initiated by users (information is "pulled" from the sources).
- Alternatively, information may be "pushed" to users.
  - Dynamically compare newly received items against standing statements of interests of users (profiles) and deliver matching items to user mail files.
  - Asynchronous (background) process.
  - Profile defines all areas of interest (whereas an individual query focuses on specific question).
  - Each item compared against many profiles (whereas each query is compared against many items).

Σχήμα

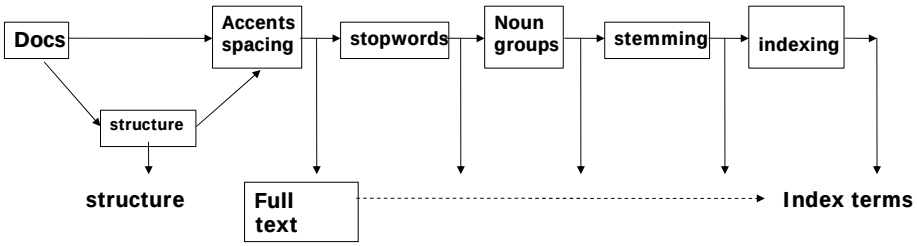
Basic Concepts: Logical View of Documents

Logical view of the documents

Keywords (tagging, or extracted automatically)

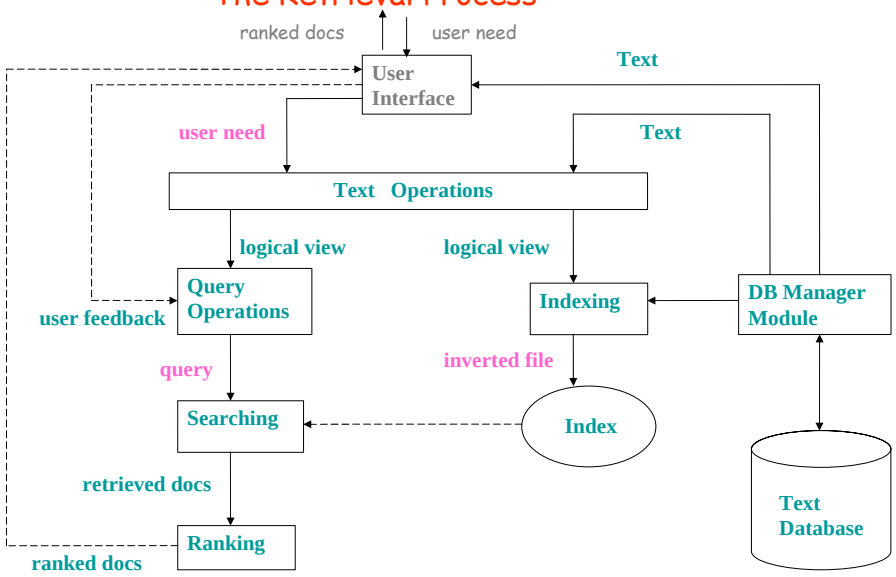
Full-text ->(text operations) -> Index terms
(also structure)

Basic Concepts: Logical View of Documents



Full text
Document representation viewed as a continuum: logical view of docs might shift

The Retrieval Process



The Retrieval Process

- Model
  - documents to be used
  - text operations
  - text model
- Build an index
- User needs
  - query operations
- Ranking based on likelihood of relevance
- Result representation
- User feedback phase

Objectives

- Overall objective (efficiency):
  - Minimize search overhead
- Measurement of success (effectiveness):
  - Precision and recall
- Facilitate the overall objective:
  - Good search tools
  - Helpful presentation of results

Minimize search overhead

- Minimize *overhead* of a user who is locating *needed information*.
- *Overhead*: Time spent in all steps leading to the reading of items containing the needed information (query generation, query execution, scanning results, reading non-relevant items, etc.).
- *Needed information*: Either
  - Sufficient information in the system to complete a task.
  - All information in the system relevant to the user needs.
  - Example –shopping:
    - Looking for an item to purchase.
    - Looking for an item to purchase at minimal cost.
  - Example –researching:
    - Looking for a bibliographic citation that explains a particular term.
    - Building a comprehensive bibliography on a particular subject.

Measurement of success

Two dual measures:

- Precision: Proportion of items retrieved that are relevant.

  Precision = relevant retrieved / total retrieved

  $= |Answer \cap Relevant| \ / \ |Answer|$

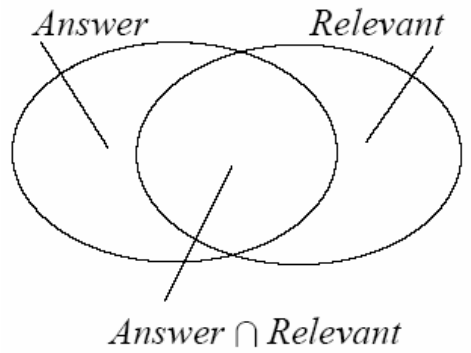
- Recall: Proportion of relevant items that are retrieved.

  Recall = relevant retrieved / relevant exist

  $= |Answer \cap Relevant| \ / \ |Relevant|$

- Most popular measures, but others exist.

Measurement of success (cont.)



Measurement of success (cont.)

Relevance?

[Information need]

Related to the topic

Timely

From a reliable source

…

Presentation of results

Present search results in format that helps user determine relevant items:

- Arbitrary (physical) order
- Relevance order
- Clustered (e.g., conceptual similarity)
- Graphical (visual) representation

Support user search

Support user search, providing tools to overcome obstacles such as:

- Ambiguities inherent in languages.
  - Homographs: Words with identical spelling but with multiple meanings.
  - Example: *Chinon*—Japanese electronics, French chateau.
- Limits to user's ability to express needs.
  - Lack of system experience or aptitude.
- Lack of expertise in the area being searched.
  - Initially only vague concept of information sought.
  - Differences between user's vocabulary and authors' vocabulary: different words with similar meanings.

History

Library search

Past, present and future

1960s-1970s

Initial exploration of text retrieval systems for “small” corpora of scientific abstracts and law and business documents

Basic boolean and vector-space models of retrieval

Salton (Cornell)

1980s

Legal document database systems, many run by companies

- Lexis-Nexis
- Dialog
- Medline

Past, present and future

1990's

Searching FTP-able documents on the Internet

- Archie
- WAIS

Searching the World-Wide-Web

- Lycos
- Yahoo
- Altavista

Recommender Systems

- Ringo
- Amazon
- NetPerceptions

Automatic Text Categorization and Clustering

Past, present and future

2000's

Link Analysis for Web Search

- Google

Automated Information Extraction

- Whizbang
- Fetch
- Burning Glass

Question Answering

- TREC Q/A track

Multimedia IR

Cross-Language IR

Document Summarization

WEB changed everything

New issues:

Trust

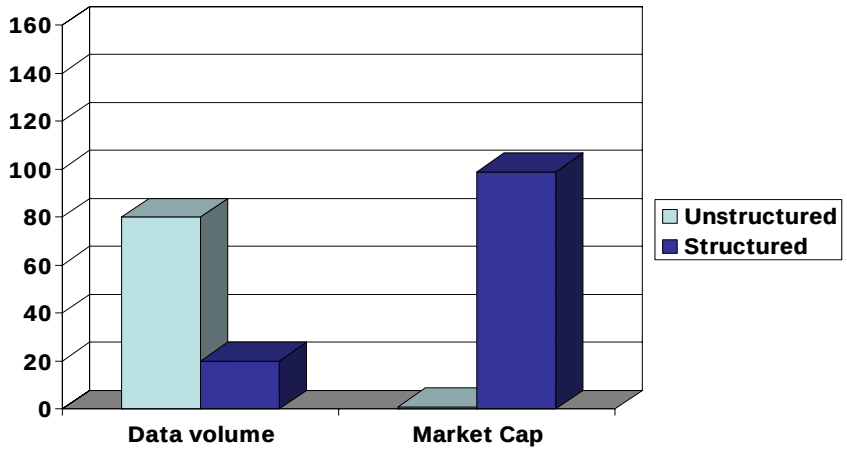
Privacy, etc

Additional sources:

Social networking

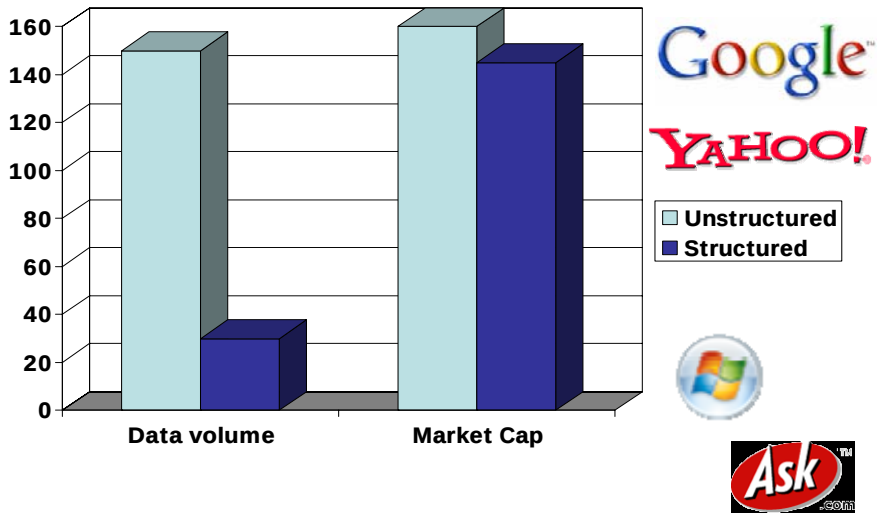
Wikipedia, etc

Unstructured (text) vs. structured (database) data in 1996



29

Unstructured (text) vs. structured (database) data in 2006



Search Engines and Web Today

Indexed web: at least 45.84 billion pages

2 exabytes (2^{60}) per year -- 90% in digital form

50% increase per year

Case Study

Unstructured data in 1680

- 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***?
  - Slow (for large corpora)
  - *NOT* ***Calpurnia*** is non-trivial
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
  - Ranked retrieval (best documents to return)

Term-document incidence

	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* = 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

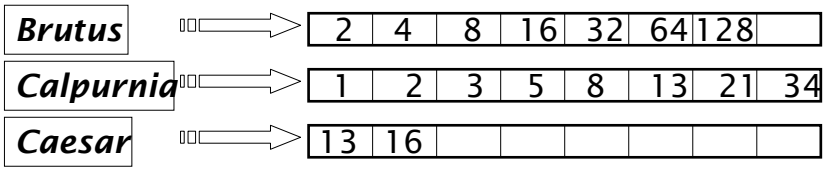
- 500K x 1M matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's.
  - matrix is extremely sparse.

Why?

- What's a better representation?
  - We only record the 1 positions.

Inverted index

- For each term *T*, we must store a list of all documents that contain *T*.

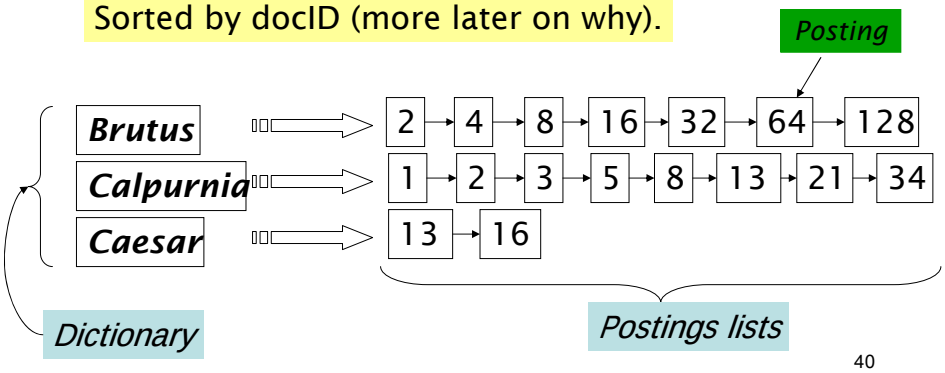


What happens if the word ***Caesar*** is added to document 14?

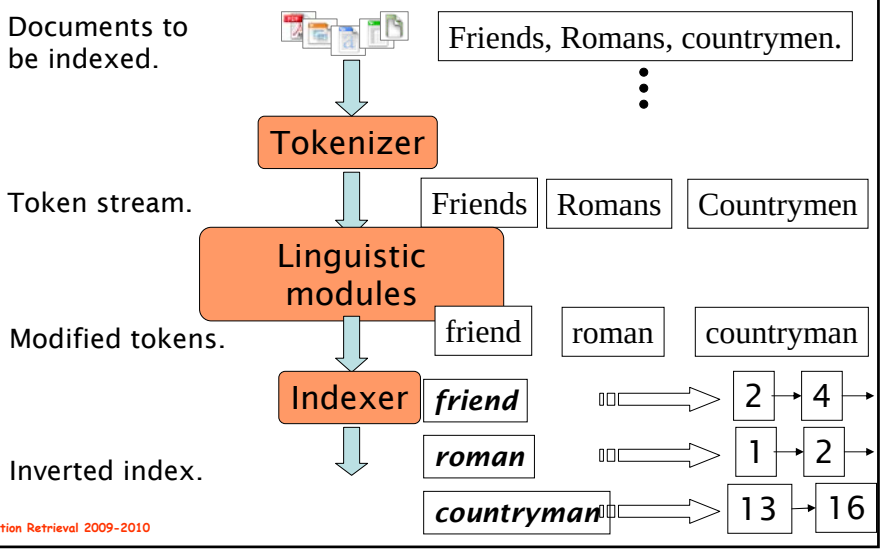
Inverted index (continued)

- 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

Indexer steps (continued)

- 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

Indexer steps (continued)

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

Why frequency?
Will discuss later.

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

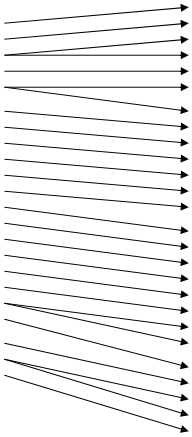
Term	Doc #	Term freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

Indexer steps (continued)

- 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



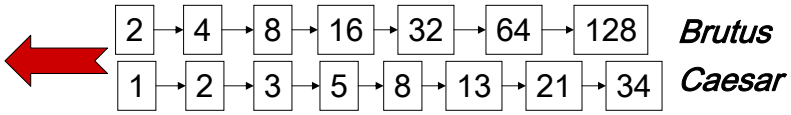
Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
1	2
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

Using the index

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

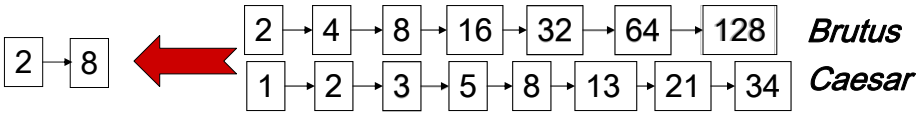
Query processing: AND

- 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.

Example: WestLaw http://www.westlaw.com/

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- Tens of terabytes of data; 700,000 users
- Majority of users *still* use boolean queries
- Example query:
  - What is the statute of limitations in cases involving the federal tort claims act?
  - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
- /3 = within 3 words, /S = in same sentence

Exercise

- Try the search feature at http://www.rhymezone.com/shakespeare/
- Write down five search features you think it could do better

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***).
- Frequency information
- One document as a singleton or group
- Content clustering and classification
- Concept (vs keyword queries)

Course Content

Retrieval Models

Retrieval Evaluation

Indexing

Query operations (relevance feedback, query expansion, clustering, etc)

Web search

Parallel and distributed (P2P and MapReduce)

Social Networks