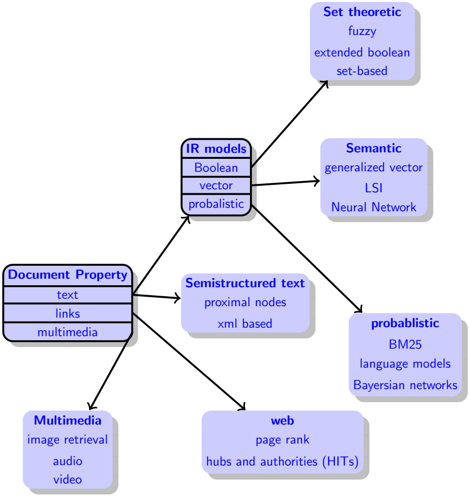
boolean model

September 8, 2024

Reading material: Chapter 1 of IIR

Outline

taxonomy of IR models



Outline

1. boolean queries
2. Inverted index
3. query processing
4. Query optimization

Boolean retrieval

- The Boolean model: the simplest model for an information retrieval system.
- Queries are Boolean expressions, e.g.,
  - CAESAR AND BRUTUS
- The search engine returns all documents that satisfy the Boolean expression.

Boolean retrieval

- The Boolean model: the simplest model for an information retrieval system.
- Queries are Boolean expressions, e.g.,
  - Caesar and Brutus
- The search engine returns all documents that satisfy the Boolean expression.
- Does Google use the Boolean model?

Does Google use the Boolean model?

- Not a simple yes or no question.
- On Google, the default interpretation of a query
  $w_1\ w_2 \ldots w_n$
  is
  w_1 AND w_2 AND ... AND w_n

Does Google use the Boolean model?

- Not a simple yes or no question.
- On Google, the default interpretation of a query
  $w_1 \ w_2 \ldots w_n$
  is
  w_1 AND w_2 AND ... AND w_n
- Cases where you get hits that do not contain one of the w_i?

Cases that are not exact boolean query

- page contains variant of w_i (morphology, spelling correction, synonym)

boolean expression generates very few hits

Cases that are not exact boolean query

- page contains variant of w_i (morphology, spelling correction, synonym)
- long queries (n large)

boolean expression generates very few hits

Cases that are not exact boolean query

- page contains variant of w_i (morphology, spelling correction, synonym)
- long queries (n large)
- anchor text

```
<a href="www.ibm.com"> big blue </a>
```

boolean expression generates very few hits

Cases that are not exact boolean query

- page contains variant of w_i (morphology, spelling correction, synonym)
- long queries (n large)
- anchor text

   big blue

- ...

boolean expression generates very few hits

Simple Boolean vs. Ranking of result set

- Simple Boolean retrieval returns matching documents in no particular order.
- Google (and others) ranks the result set
  - they rank good hits (according to some estimator of relevance) higher than bad hits.

Outline

Unstructured data in 1650

- 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.

```
grep Brutus *.txt
```

- Why is `grep` not the solution?

Unstructured data in 1650

- 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.

```
grep Brutus *.txt
```

- Why is `grep` not the solution?
  - Slow (for large collections)
  - grep is line-oriented, IR is document-oriented
  - "NOT CALPURNIA" is non-trivial
  - Other operations (e.g., find the word ROMANS near COUNTRYMAN) not feasible

Term-document incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	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	
...							

Entry is 1 if term occurs.
Example: CALPURNIA occurs in *Julius Caesar*.
Entry is 0 if term doesn't occur.
Example: CALPURNIA doesn't occur in *The tempest*.

Term-document incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	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	
...							

Entry is 1 if term occurs.

Example: CALPURNIA occurs in *Julius Caesar*.

Entry is 0 if term doesn't occur.

Example: CALPURNIA doesn't occur in *The tempest*.

Term-document incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	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	
...							

Entry is 1 if term occurs.
Example: CALPURNIA occurs in *Julius Caesar*.
Entry is 0 if term doesn't occur.
Example: CALPURNIA doesn't occur in *The tempest*.

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer the query BRUTUSAND CAESAR AND NOT CALPURNIA:
  - Take the vectors for BRUTUS, CAESAR, and CALPURNIA
  - Complement the vector of CALPURNIA
  - Do a (bitwise) AND on the three vectors

BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
not CALPURNIA	1	0	1	1	1	1
AND	1	0	0	1	0	0

0/1 vectors and result of bitwise operations

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	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	
...							
result:	1	0	0	1	0	0	

Answers to query

Anthony 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 $N = 10^6$ documents, each with about 1000 tokens
  - $\Rightarrow$ total of 10^9 tokens
- On average 6 bytes per token, including spaces and punctuation
  - $\Rightarrow$ size of document collection is about $6 \cdot 10^9 = 6$ GB
- Assume there are $M = 500{,}000$ distinct terms in the collection
- (Notice that we are making a term/token distinction.)

Can't build the incidence matrix

- $M = 500,000 \times 10^6 =$ half a trillion 0s and 1s.
- But the matrix has no more than one billion 1s.
  - Matrix is extremely sparse.
- What is a better representations?

Can't build the incidence matrix

- $M = 500,000 \times 10^6 =$ half a trillion 0s and 1s.
- But the matrix has no more than one billion 1s.
  - Matrix is extremely sparse.
- What is a better representations?
  - We only record the 1s.

Inverted Index

For each term t, we store a list of all documents that contain t.

BRUTUS	$\longrightarrow$	1	2	4	11	31	45	173	174	
CAESAR	$\longrightarrow$	1	2	4	5	6	16	57	132	...
CALPURNIA	$\longrightarrow$	2	31	54	101					
⋮										

dictionary **postings**

Inverted Index

For each term t, we store a list of all documents that contain t.

BRUTUS	$\longrightarrow$	1, 2, 4, 11, 31, 45, 173, 174
CAESAR	$\longrightarrow$	1, 2, 4, 5, 6, 16, 57, 132, ...
CALPURNIA	$\longrightarrow$	2, 31, 54, 101
⋮		
dictionary		**postings**

Inverted Index

For each term t, we store a list of all documents that contain t.

BRUTUS	$\longrightarrow$	1	2	4	11	31	45	173	174	
CAESAR	$\longrightarrow$	1	2	4	5	6	16	57	132	...
CALPURNIA	$\longrightarrow$	2	31	54	101					

$\vdots$

dictionary **postings**

Inverted index construction

1. Collect the documents to be indexed:
   | Friends, Romans, countrymen. | | So let it be with Caesar | ...

2. Tokenize the text, turning each document into a list of tokens:
   | Friends | | Romans | | countrymen | | So | ...

3. Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms:
   | friend | | roman | | countryman | | so | ...

4. Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings.

Tokenization and preprocessing

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:

$\Longrightarrow$

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

Generate postings

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

$\Longrightarrow$

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

Sort postings

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

$\Longrightarrow$

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

Create postings lists, determine document frequency

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

$\Longrightarrow$

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

Split the result into dictionary and postings file

BRUTUS	→	1	2	4	11	31	45	173	174	
CAESAR	→	1	2	4	5	6	16	57	132	...
CALPURNIA	→	2	31	54	101					
⋮										
dictionary		**postings file**								

Outline

Simple conjunctive query (two terms)

- Consider the query: BRUTUS AND CALPURNIA
- To find all matching documents using inverted index:
  1. Locate BRUTUS in the dictionary
  2. Retrieve its postings list from the postings file
  3. Locate CALPURNIA in the dictionary
  4. Retrieve its postings list from the postings file
  5. Intersect the two postings lists
  6. Return intersection to user

Intersecting two postings lists

$$
\begin{array}{lll}
\textsc{Brutus} & \longrightarrow & \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101} \\
\\
\text{Intersection} & \Longrightarrow &
\end{array}
$$

Intersecting two postings lists

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$

Intersecting two postings lists

$$\textsc{Brutus} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$$

$$\textsc{Calpurnia} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$$

$$\text{Intersection} \Longrightarrow$$

Intersecting two postings lists

$\textsc{Brutus} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$

$\textsc{Calpurnia} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

Intersection $\Longrightarrow \boxed{2}$

Intersecting two postings lists

$$
\begin{array}{lcl}
\textsc{Brutus} & \longrightarrow & \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101} \\
\text{Intersection} & \Longrightarrow & \boxed{2}
\end{array}
$$

Intersecting two postings lists

$$
\begin{array}{lcl}
\textsc{Brutus} & \longrightarrow & \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101} \\
\text{Intersection} & \Longrightarrow & \boxed{2}
\end{array}
$$

Intersecting two postings lists

$\textsc{Brutus} \longrightarrow \boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

$\textsc{Calpurnia} \longrightarrow \boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow \boxed{2}$

Intersecting two postings lists

$$\begin{array}{lll}
\textsc{Brutus} & \longrightarrow & \boxed{1}\to\boxed{2}\to\boxed{4}\to\boxed{11}\to\boxed{31}\to\boxed{45}\to\boxed{173}\to\boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2}\to\boxed{31}\to\boxed{54}\to\boxed{101} \\
\text{Intersection} & \Longrightarrow & \boxed{2}\to\boxed{31}
\end{array}$$

Intersecting two postings lists

$\textsc{Brutus} \longrightarrow \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$

$\textsc{Calpurnia} \longrightarrow \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

$\text{Intersection} \Longrightarrow \boxed{2} \to \boxed{31}$

Intersecting two postings lists

BRUTUS	$\longrightarrow$	$\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$
Intersection	$\Longrightarrow$	$\boxed{2} \to \boxed{31}$

Intersecting two postings lists

$$
\begin{array}{lcl}
\textsc{Brutus} & \longrightarrow & \boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101} \\
\text{Intersection} & \Longrightarrow & \boxed{2} \to \boxed{31}
\end{array}
$$

Intersecting two postings lists

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

Intersection $\Longrightarrow$ $\boxed{2} \rightarrow \boxed{31}$

Intersecting two postings lists

$$
\begin{array}{lcl}
\textsc{Brutus} & \longrightarrow & \boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174} \\
\textsc{Calpurnia} & \longrightarrow & \boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101} \\
\text{Intersection} & \Longrightarrow & \boxed{2} \rightarrow \boxed{31}
\end{array}
$$

- This is linear in the length of the postings lists.

Intersecting two postings lists

BRUTUS	$\longrightarrow$	$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$
Intersection	$\Longrightarrow$	$\boxed{2} \rightarrow \boxed{31}$

- This is linear in the length of the postings lists.
- Note: This only works if postings lists are sorted.

Intersecting two postings lists

$\textsc{Intersect}(p_1, p_2)$

- $\mathit{answer} \leftarrow \langle\ \rangle$
- **while** $p_1 \neq \textsc{nil}$ and $p_2 \neq \textsc{nil}$
- **do if** $\mathit{docID}(p_1) = \mathit{docID}(p_2)$
  - **then** $\textsc{Add}(\mathit{answer}, \mathit{docID}(p_1))$
    - $p_1 \leftarrow \mathit{next}(p_1)$
    - $p_2 \leftarrow \mathit{next}(p_2)$
  - **else if** $\mathit{docID}(p_1) < \mathit{docID}(p_2)$
    - **then** $p_1 \leftarrow \mathit{next}(p_1)$
    - **else** $p_2 \leftarrow \mathit{next}(p_2)$
- **return** *answer*

Question: can we make it faster?

Query processing: Exercise

FRANCE $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{3} \rightarrow \boxed{4} \rightarrow \boxed{5} \rightarrow \boxed{7} \rightarrow \boxed{8} \rightarrow \boxed{9} \rightarrow \boxed{11} \rightarrow \boxed{12} \rightarrow \boxed{13} \rightarrow \boxed{14} \rightarrow \boxed{15}$

PARIS $\longrightarrow$ $\boxed{2} \rightarrow \boxed{6} \rightarrow \boxed{10} \rightarrow \boxed{12} \rightarrow \boxed{14}$

LEAR $\longrightarrow$ $\boxed{12} \rightarrow \boxed{15}$

Compute hit list for ((paris AND NOT france) OR lear)

Boolean retrieval model: Assessment

- The Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
  - Views each document as a set of terms.
  - Is precise: Document matches condition or not.
- Primary commercial retrieval tool for 3 decades
- Many professional searchers (e.g., lawyers) still like Boolean queries.
  - You know exactly what you are getting.
- Many search systems you use are also Boolean: spotlight, email, intranet etc.

Outline

Query optimization

- Consider a query that is an AND of n terms, $n > 2$
- For each of the terms, get its postings list, then AND them together
- Example query: BRUTUS AND CALPURNIA AND CAESAR
- What is the best order for processing this query?

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR

BRUTUS	$\longrightarrow$	$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$
CAESAR	$\longrightarrow$	$\boxed{5} \rightarrow \boxed{31}$

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- Simple and effective optimization: Process in order of increasing frequency

BRUTUS	$\longrightarrow$	$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$
CALPURNIA	$\longrightarrow$	$\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$
CAESAR	$\longrightarrow$	$\boxed{5} \rightarrow \boxed{31}$

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- Simple and effective optimization: Process in order of increasing frequency
- Start with the shortest postings list, then keep cutting further

BRUTUS $\longrightarrow$ $\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

CAESAR $\longrightarrow$ $\boxed{5} \to \boxed{31}$

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- Simple and effective optimization: Process in order of increasing frequency
- Start with the shortest postings list, then keep cutting further

BRUTUS $\longrightarrow$ $\boxed{1} \to \boxed{2} \to \boxed{4} \to \boxed{11} \to \boxed{31} \to \boxed{45} \to \boxed{173} \to \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \to \boxed{31} \to \boxed{54} \to \boxed{101}$

CAESAR $\longrightarrow$ $\boxed{5} \to \boxed{31}$

Query optimization

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- Simple and effective optimization: Process in order of increasing frequency
- Start with the shortest postings list, then keep cutting further
- In this example, first CAESAR, then CALPURNIA, then BRUTUS

BRUTUS $\longrightarrow$ $\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4} \rightarrow \boxed{11} \rightarrow \boxed{31} \rightarrow \boxed{45} \rightarrow \boxed{173} \rightarrow \boxed{174}$

CALPURNIA $\longrightarrow$ $\boxed{2} \rightarrow \boxed{31} \rightarrow \boxed{54} \rightarrow \boxed{101}$

CAESAR $\longrightarrow$ $\boxed{5} \rightarrow \boxed{31}$

Optimized intersection algorithm for conjunctive queries

```
INTERSECT(⟨t_1, ..., t_n⟩)
terms ← SORTBYINCREASINGFREQUENCY(⟨t_1, ..., t_n⟩)
result ← postings(first(terms))
terms ← rest(terms)
while terms ≠ NIL and result ≠ NIL
do result ← INTERSECT(result, postings(first(terms)))
    terms ← rest(terms)
return result
```

More general optimization

- Example query: (MADDING OR CROWD) AND (IGNOBLE OR STRIFE)
- Get frequencies for all terms
- Estimate the size of each OR by the sum of its frequencies (conservative)
- Process in increasing order of OR sizes

Advantages and disadvantages of Boolean Model

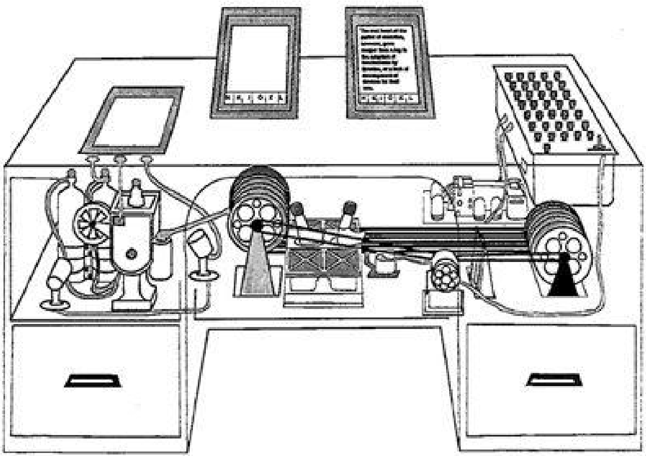
Advantages:

- Easy for the system
- Users get transparency: it is easy to understand why a document was or was not retrieved
- Users get control: it easy to determine whether the query is too specific (few results) or too broad (many results)

Disadvantages:

- The burden is on the user to formulate a good boolean query

search engine envisioned in 1945



memex (memory extender), described by Vannevar Bush in 1945.
"As We May Think".

- The memex (memory extender) is the name of the hypothetical proto-hypertext system
- Bush envisioned the memex as a device in which individuals would compress and store all of their books, records, and communications,
- "mechanized so that it may be consulted with exceeding speed and flexibility."
- The memex would provide an "enlarged intimate supplement to one's memory".
- The concept of the memex influenced the development of early hypertext systems (eventually leading to the creation of the World Wide Web).
- used a form of document bookmark list, of static microfilm pages, rather than a true hypertext system where parts of pages would have internal structure beyond the common textual format.