

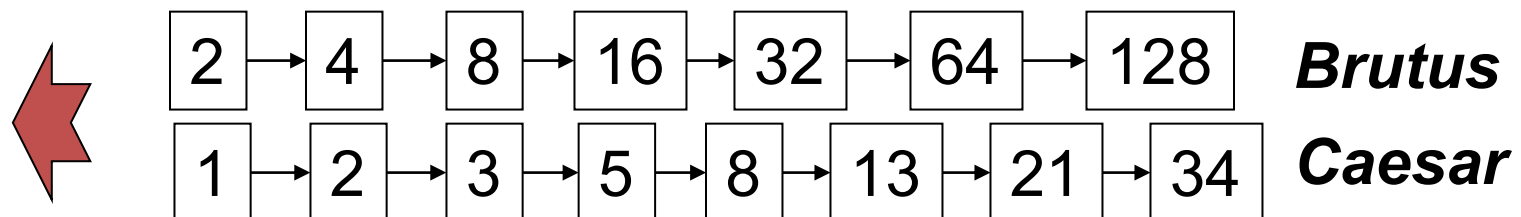
INFORMATION RETRIEVAL

Dr. Reda M. Hussien

Query Processing with the Inverted Index

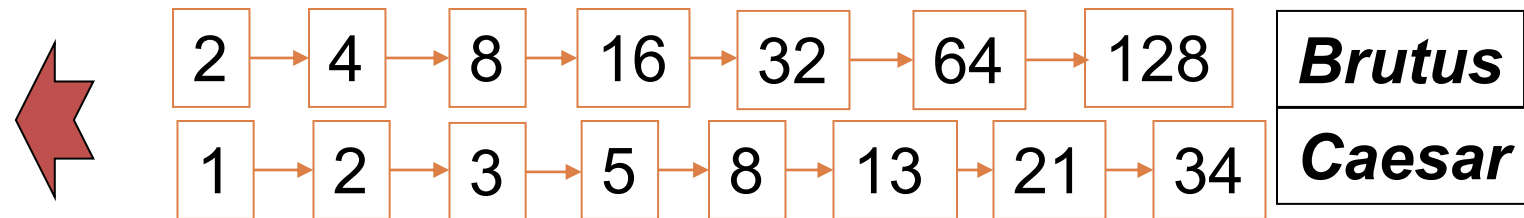
Query processing: AND

- 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 (intersect the document sets):



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.

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  $answer$ 
```

Boolean queries: Exact match

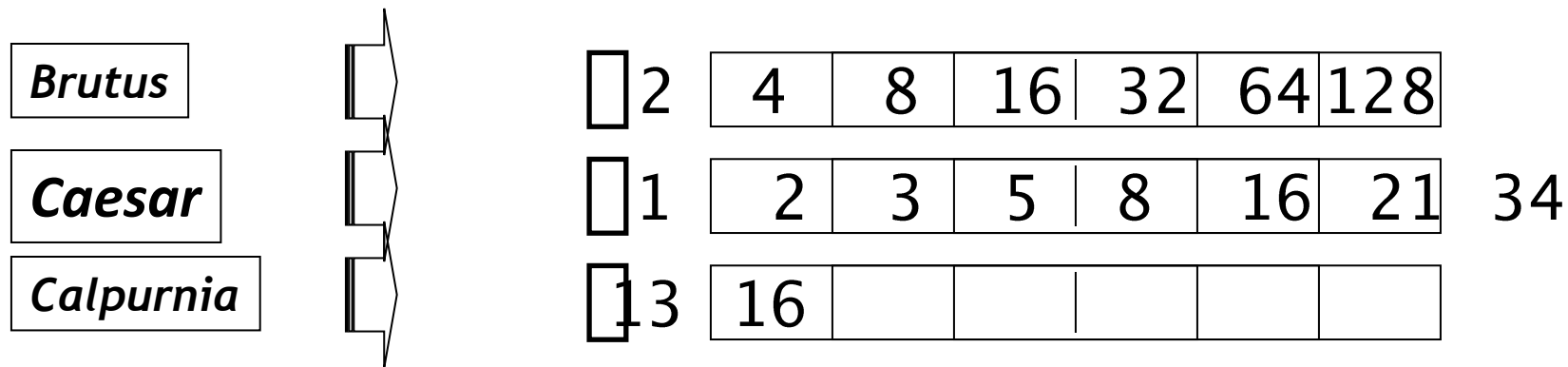
- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
 - Boolean Queries are queries using AND, OR and NOT to join query terms
 - Views each document as a set of words
 - Is precise: document matches condition or not.
 - Perhaps the simplest model to build an IR system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

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 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.*

<i>Brutus</i>	2	4	8	16	32	64	128	
<i>Caesar</i>	1	2	3	5	8	16	21	34
<i>Calpurnia</i>	13	16						

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

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

-
- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992; new federated search added 2010)
 - 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

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

- Another example query:
 - Requirements for disabled people to be able to access a workplace
 - disabl! /p access! /s work-site work-place (employment /3 place
- Note that SPACE is disjunction, not conjunction!
- Long, precise queries; proximity operators; incrementally developed; not like web search
- Many professional searchers still like Boolean search
 - You know exactly what you are getting
- But that doesn't mean it actually works better....



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 in increasing order of OR sizes.

Exercise

- Recommend a query processing order for

(tangerine OR trees) AND

(marmalade OR skies) AND

(kaleidoscope OR eyes)

- Which two terms should we process first?

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

Query processing exercises

- **Exercise:** 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.

Exercise

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

Phrase queries

- We want to be able to answer queries such as “stanford university” - as a phrase
- Thus the sentence “I went to university at Stanford” is not a match.
 - The concept of phrase queries has proven easily understood by users; one of the few “advanced search” ideas that works
 - Many more queries are implicit phrase queries
- For this, it no longer suffices to store only
- <term : docs> entries



A first attempt: Biword 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 can be processed by breaking them down
- stanford university palo alto can be broken into the Boolean query on biwords:
- stanford university AND university palo AND palo alto
- Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase.



Issues for biword indexes

- False positives, as noted before
- 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

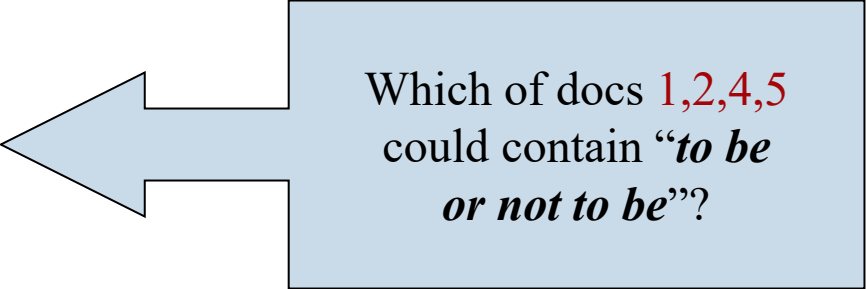
- In the postings, store, for each term the position(s) in which tokens of it appear:
 - <term, number of docs containing term;
 - doc1: position1, position2 ... ;
 - doc2: position1, position2 ... ;
 - etc.>



Positional index example

- For phrase queries, we use a merge algorithm recursively at the document level
- But we now need to deal with more than just equality

<*be*: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, ...>



Which of docs *1,2,4,5*
could contain “*to be*
or not to be”?



Processing a phrase query

- Extract inverted index entries for each distinct term: ***to***, ***be***, ***or***, ***not***.
- Merge their *doc:position* lists to enumerate all positions with “***to be or not to be***”.
 - ***to***:
 - 2:1,17,74,222,551; 4:8,16,190,429,433; 7:13,23,191; ...
 - ***be***:
 - 1:17,19; 4:17,191,291,430,434; 5:14,19,101; ...
- Same general method for proximity searches



Proximity queries

- LIMIT! /3 STATUTE /3 FEDERAL /2 TORT
 - Again, here, /k means “within k words of”.
- Clearly, positional indexes can be used for such queries; biword indexes cannot.
- Exercise: Adapt the linear merge of postings to handle proximity queries. Can you make it work for any value of k?
 - This is a little tricky to do correctly and efficiently
 - See Figure 2.12 of IIR



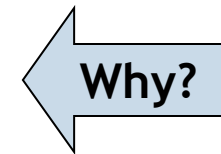
Positional index size

- A positional index expands postings storage substantially
 - Even though indices can be compressed
- Nevertheless, a positional index is now standardly used because of the power and usefulness of phrase and proximity queries ... whether used explicitly or implicitly in a ranking retrieval system.



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
1000	1	1
100,000	1	100



Rules of thumb

- A positional index is 2-4 as large as a non-positional index
- Positional index size 35-50% of volume of original text
 - Caveat: all of this holds for “English-like” languages



Combination schemes

- These two approaches can be profitably combined
 - For particular phrases (“Michael Jackson”, “Britney Spears”) it is inefficient to keep on merging positional postings lists
 - Even more so for phrases like “The Who”
- Williams et al. (2004) evaluate a more sophisticated mixed indexing scheme
 - A typical web query mixture was executed in $\frac{1}{4}$ of the time of using just a positional index
 - It required 26% more space than having a positional index alone

