Querying

Introduction to Information Retrieval

Informatics 141 / CS 121
Donald J. Patterson

Content adapted from Hinrich Schütze
http://www.informationretrieval.org

Overview

- Boolean Retrieval
- Weighted Boolean Retrieval
- Zone Indices
- Term Frequency Metrics
- The full vector space model

From the bottom

- "Grep"
  - Querying without an index or a crawl
  - Whenever you want to find something you look through the entire document for it.
  - Example:
    - You have the collected works of Shakespeare on disk
    - You want to know which play contains the words
      - "Brutus AND Caesar"

Querying

- "Grep"
  - "Brutus AND Caesar" is the query.
  - This is a boolean query. Why?
  - What other operators could be used?
  - The grep solution:
    - Read all the files and all the text and output the intersection of the files

Querying

- "Grep"
  - Slow for large corpora
  - Calculating "NOT" is non-trivial
  - Some operations not feasible
    - Query: "Romans NEAR Countrymen"
  - Doesn't support ranked retrieval
- Moving beyond grep is the motivation for the inverted index.

Querying

- "Grep"
  - Slow for large corpora
  - Calculating "NOT" is non-trivial
  - Some operations not feasible
    - Query: "Romans NEAR Countrymen"
  - Doesn't support ranked retrieval
- Moving beyond grep is the motivation for the inverted index.

Our inverted index is a 2-D array or Matrix

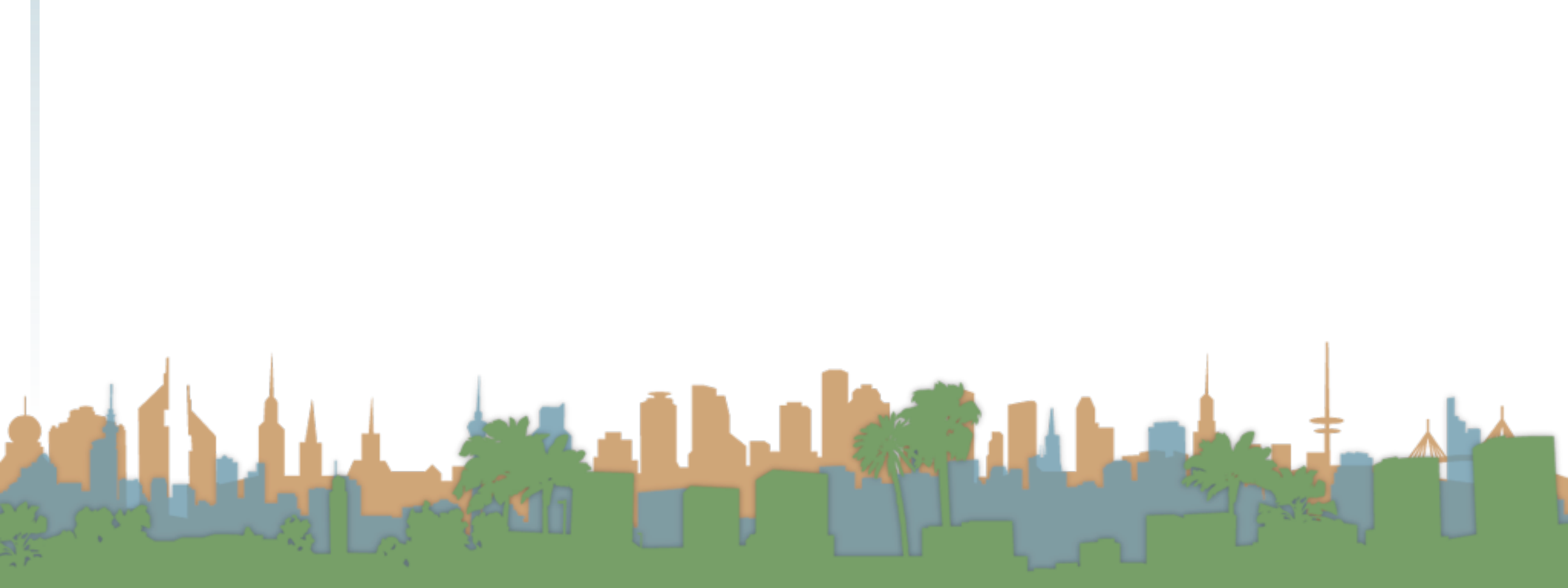
A Column For Each Document

A Row for Each Word (or "Term")

	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
...						

Querying

- Boolean Query
  - Queries are boolean expressions
  - Search returns all documents which satisfy the expression
  - Does Google use the Boolean model?

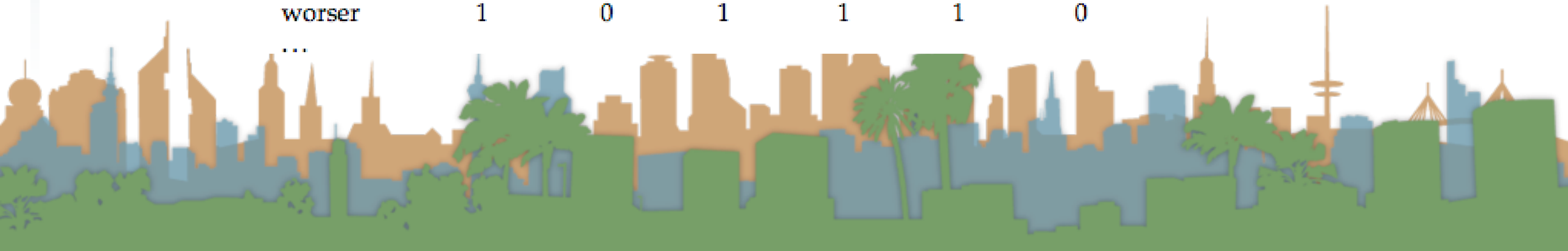


Querying

- Boolean Query
  - Straightforward application of inverted index
  - where cells of inverted index are (0,1)
    - indicating presence or absence of a term

Document

Term	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
…						



Querying

- Boolean Query
  - 0/1 vector for each term
  - “Brutus AND Caesar AND NOT Calpurnia =
  - Perform bitwise Boolean operation on each row:
  - 110100 AND 110111 AND !(010000) = 100100

Document

Term	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
…						

Querying

- Boolean Query
  - A big corpus means a sparse matrix
  - A sparse matrix motivates the introduction of the posting
    - Much less space to store
    - Only recording the “1” positions

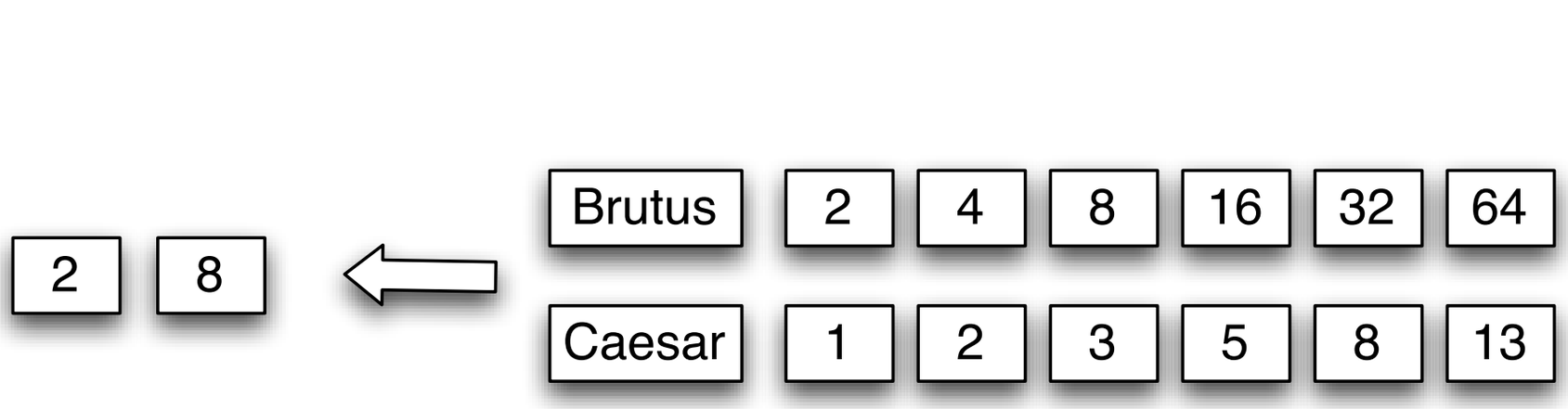
Querying

- Boolean Query
  - Query processing on postings
  - Brutus AND Caesar
    - Locate the postings for Brutus
    - Locate the postings for Caesar
    - Merge the postings

Brutus	2	4	8	16	32	64
Caesar	1	2	3	5	8	13

Querying

- Boolean Query
  - Merging -> walk through the two posting simultaneously
  - postings sorted by doc ID



Querying

- Boolean Query
  - An algorithm based on postings
  - Linear in the size of the postings

$\textsc{Intersect}(p_1, p_2)$
1 $answer \leftarrow <>$
2 **while** $p_1 \neq nil$ *and* $p_2 \neq nil$
3 **do if** $docID(p_1) = docID(p_2)$
4 **then** $\textsc{Add}(answer, docID(p_1))$
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*

Querying

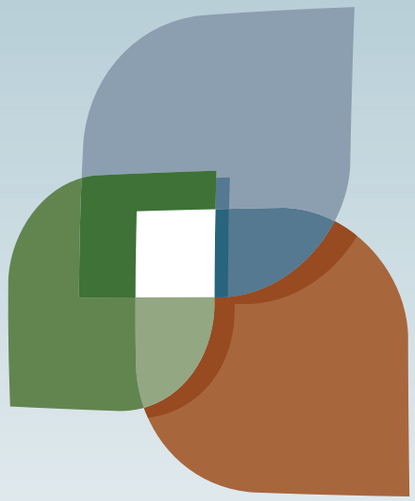
- Boolean Query
  - Is the algorithmic complexity better than scanning?
  - Where would you put more complex formulae?

```
INTERSECT(p1, p2)
 1  answer ← <>
 2  while p1 ≠ nil and p2 ≠ nil
 3        do if docID(p1) = docID(p2)
 4              then ADD(answer, docID(p1))
 5                   p1 ← next(p1)
 6                   p2 ← next(p2)
 7              else if docID(p1) < docID(p2)
 8                      then p1 ← next(p1)
 9                      else p2 ← next(p2)
10  return answer
```

Querying

- Boolean Queries
  - Exact match
  - Views each document as a "bag of words"
  - Precise: a document matches or it doesn't
  - Primary commercial retrieval tool for 3 decades
  - Professional searchers (e.g., lawyers) still like Boolean queries
    - No question about what you are getting

Not quite End of Chapter 1



LUCI