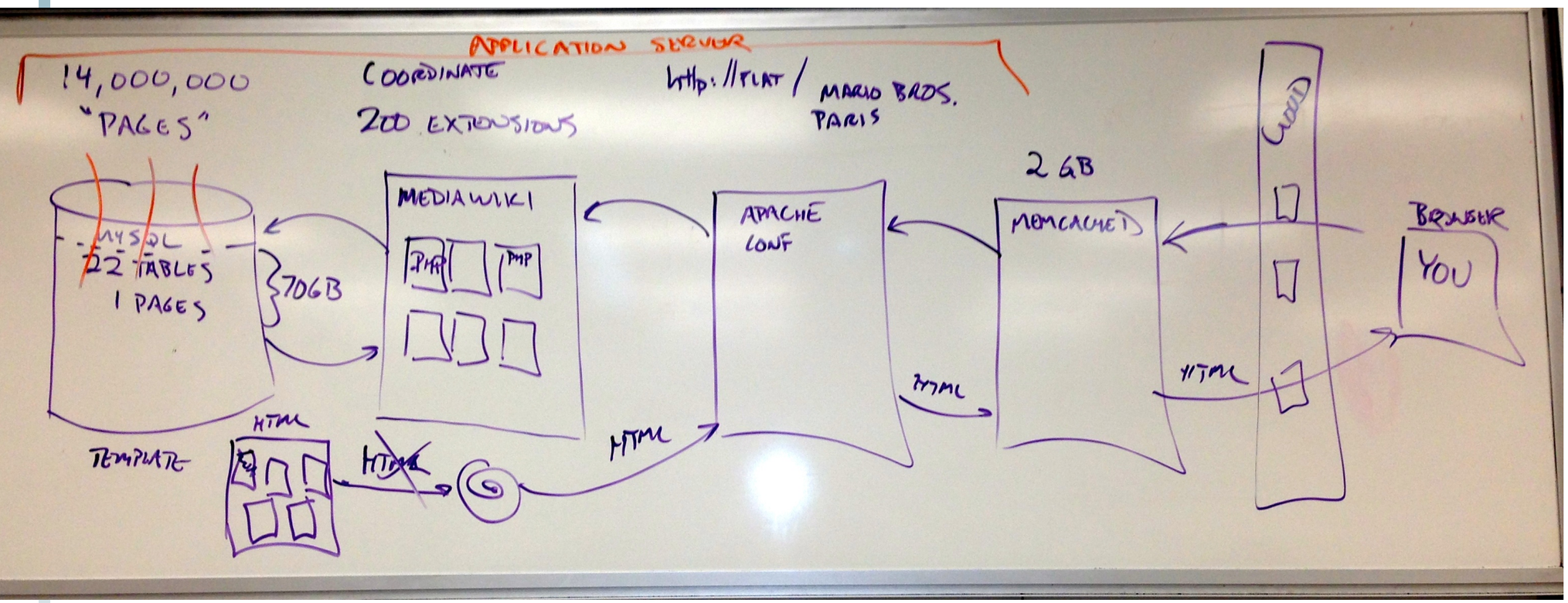
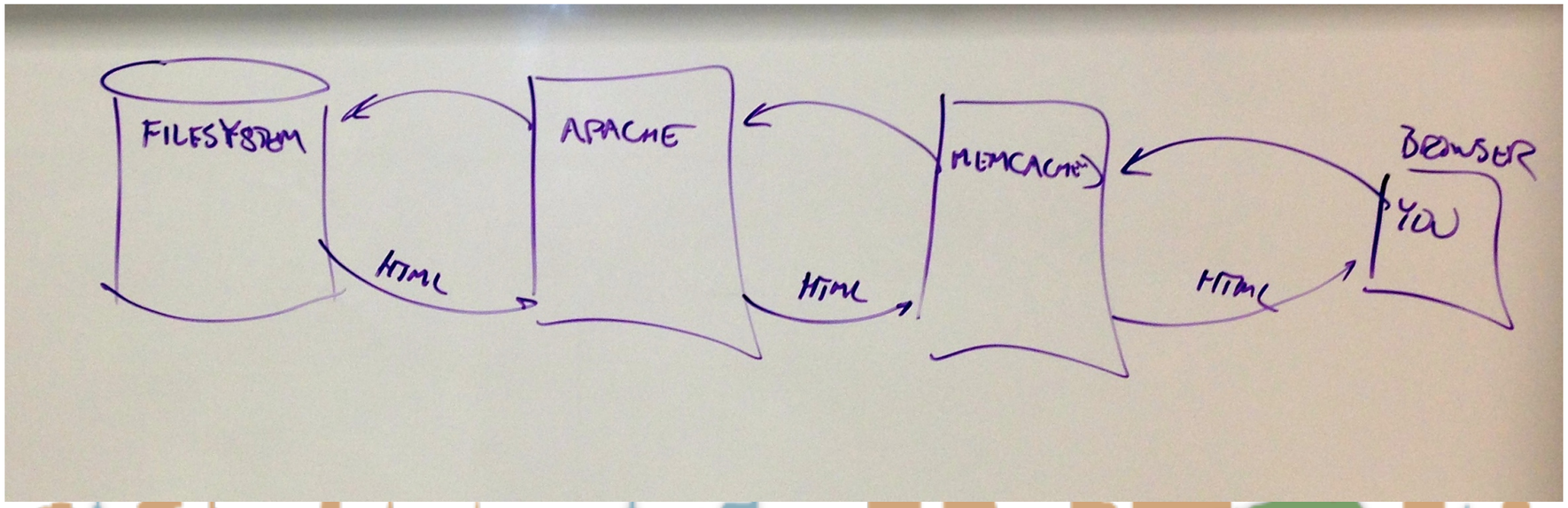
Deconstructing Wikipedia vs. Project Gutenberg





QUERYING

Introduction to Information Retrieval

CS 150
Donald J. Patterson

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

QUERYING

OVERVIEW

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

QUERYING

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

FROM THE BOTTOM

- "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

FROM THE BOTTOM

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

QUERYING

OVERVIEW

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

QUERYING

BOOLEAN RETRIEVAL

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
…						

BOOLEAN RETRIEVAL

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

QUERYING

BOOLEAN RETRIEVAL

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

Term \ Document	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 RETRIEVAL

- 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

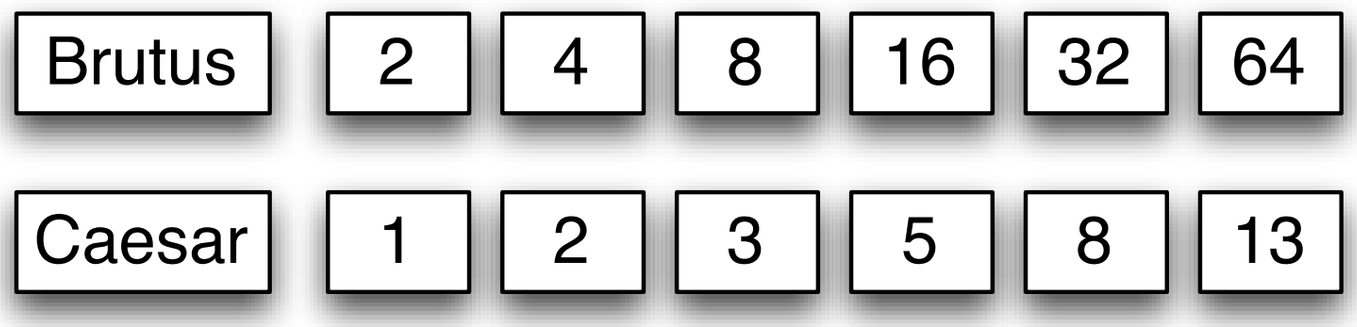
Term	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Anthony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0
…						

BOOLEAN RETRIEVAL

- 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

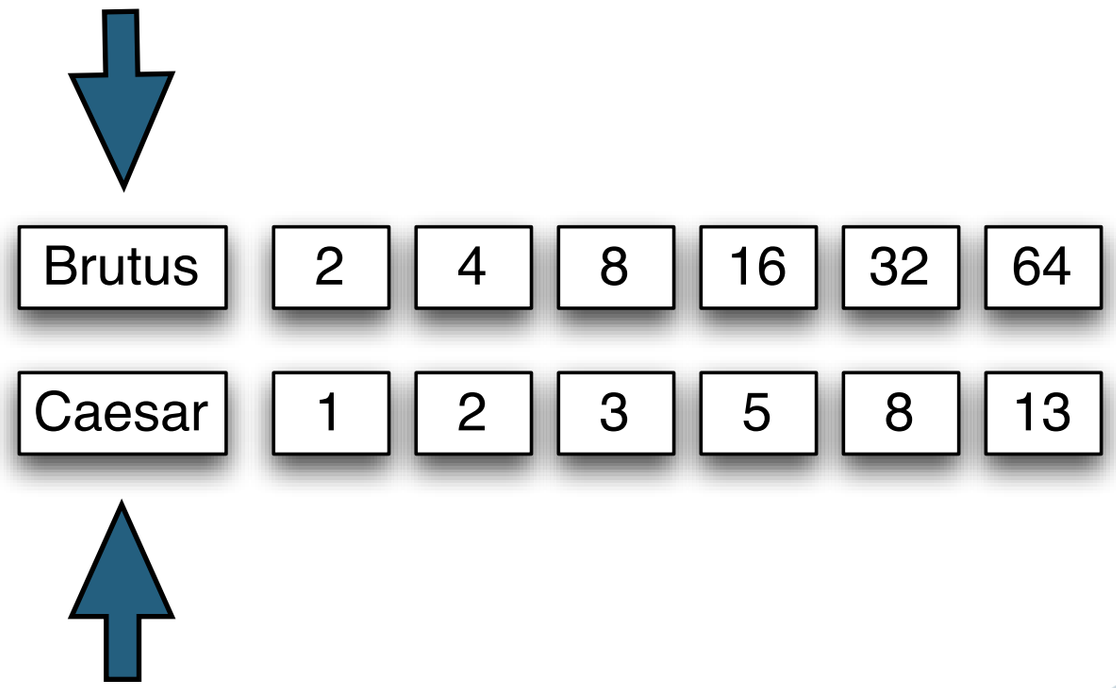
BOOLEAN RETRIEVAL

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



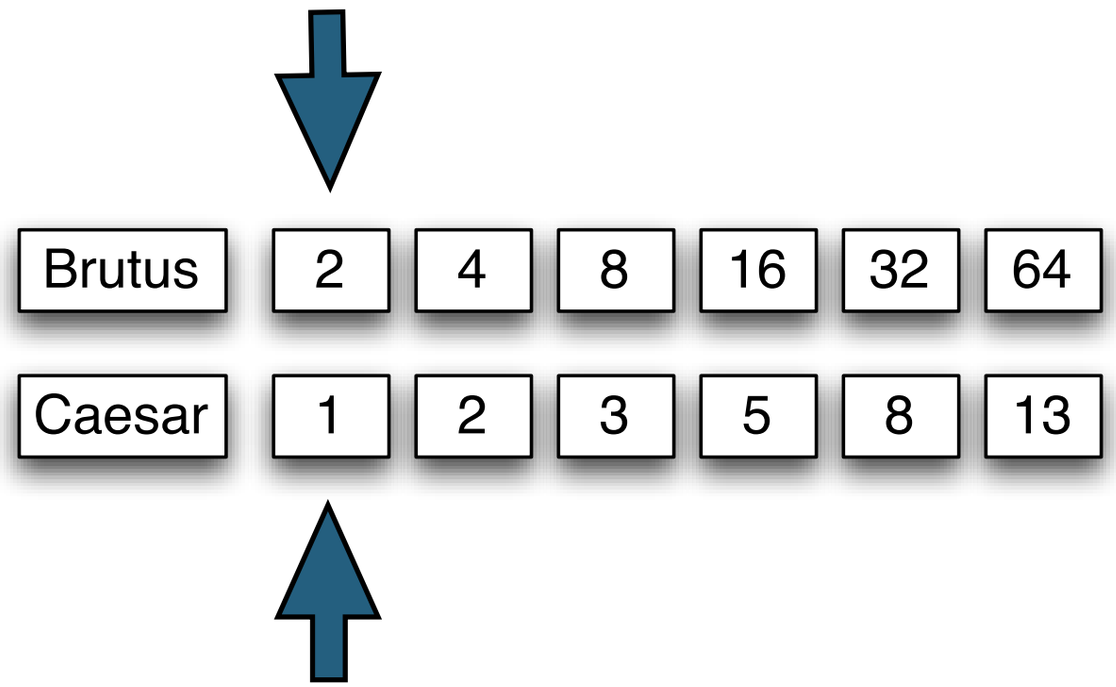
BOOLEAN RETRIEVAL

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



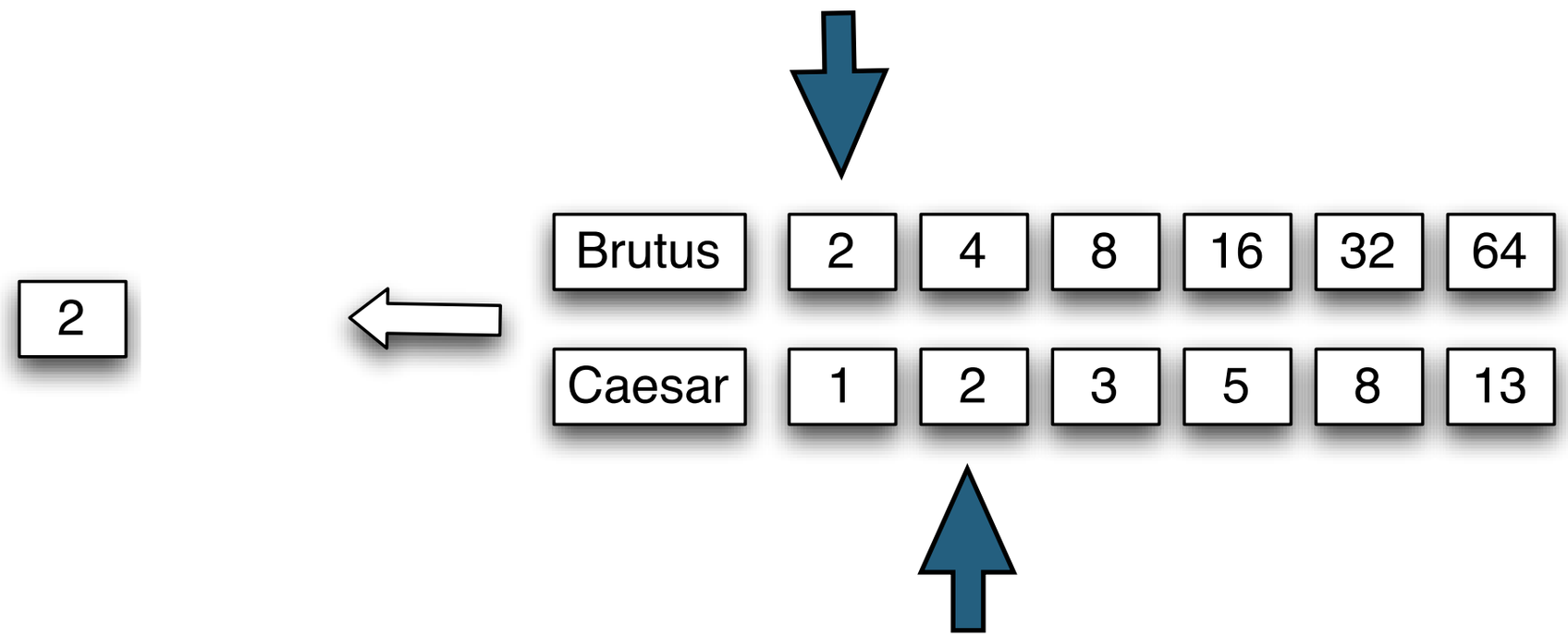
BOOLEAN RETRIEVAL

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



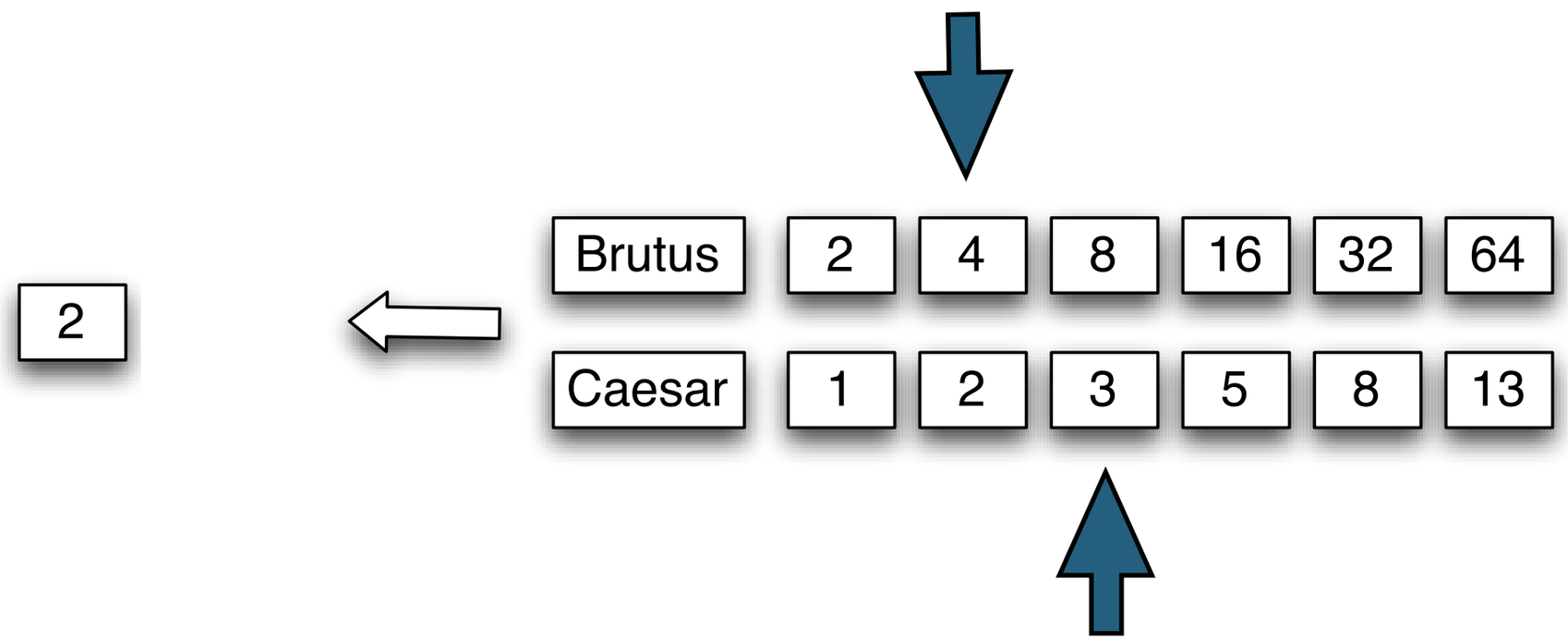
BOOLEAN RETRIEVAL

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



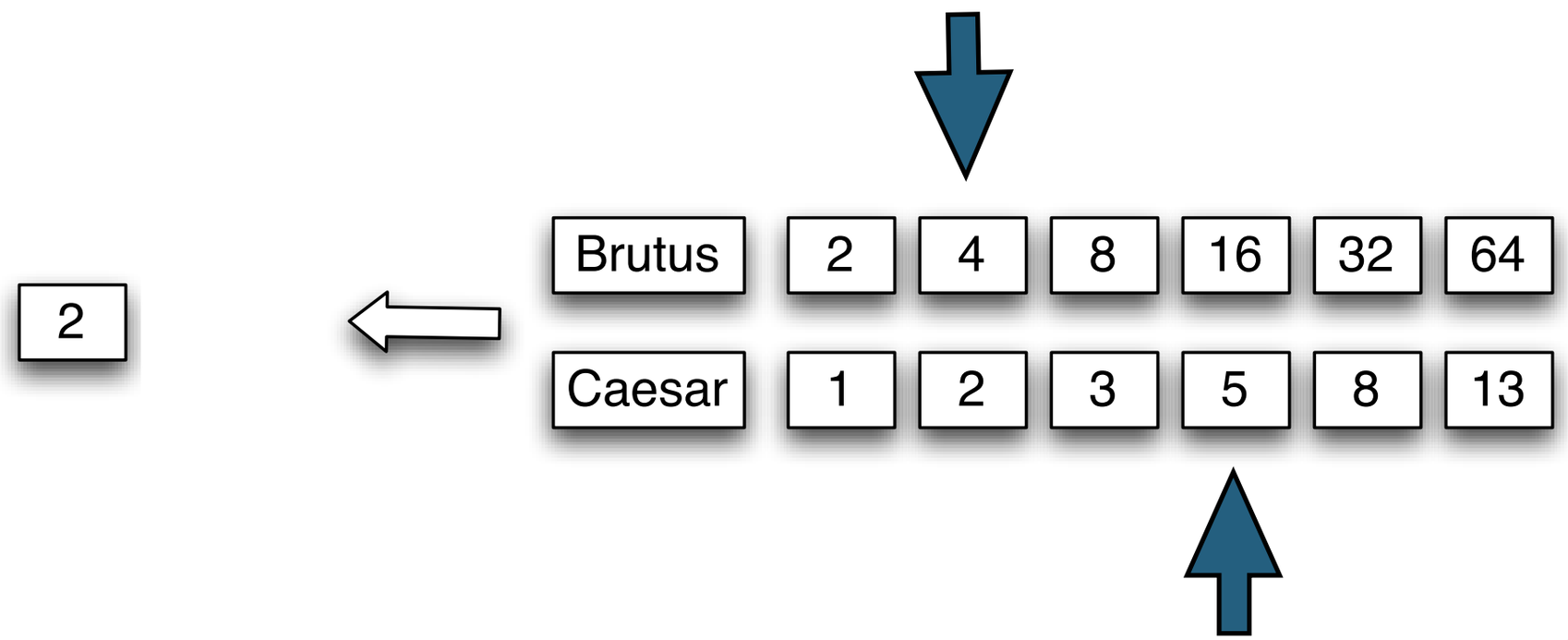
BOOLEAN RETRIEVAL

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



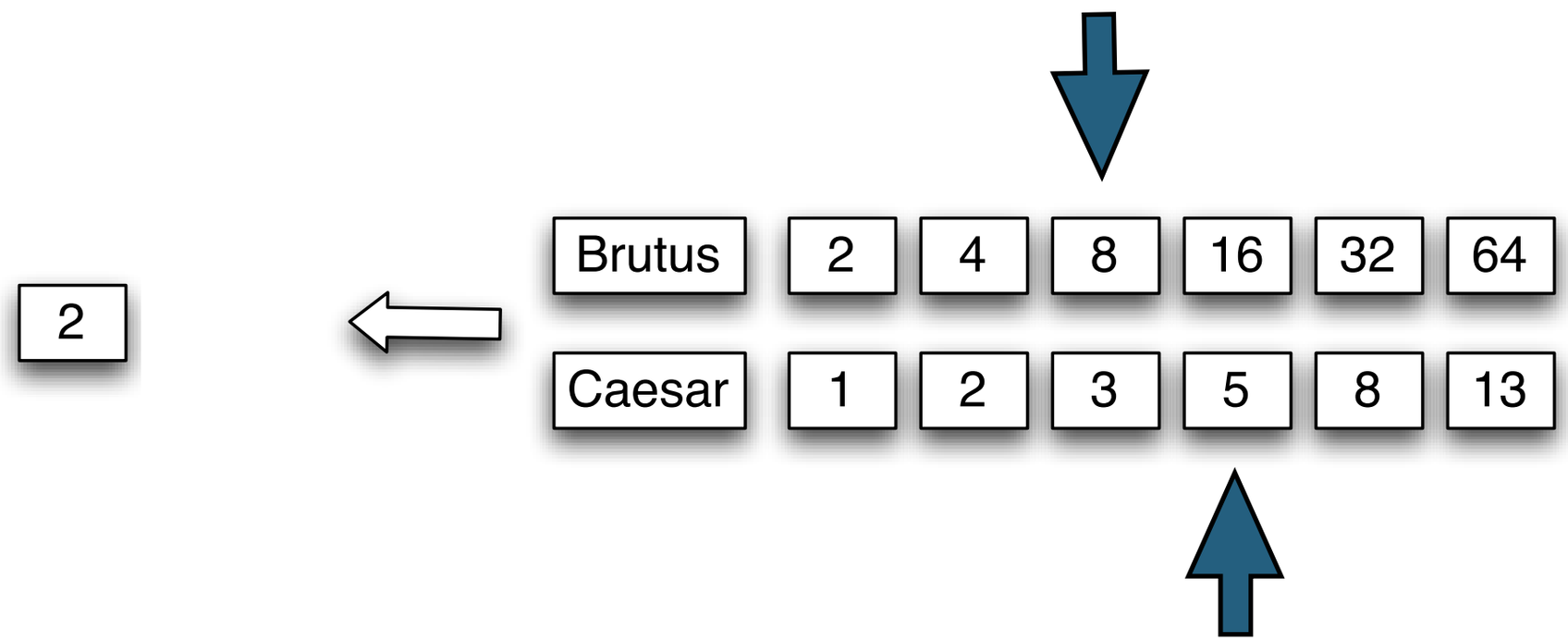
BOOLEAN RETRIEVAL

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



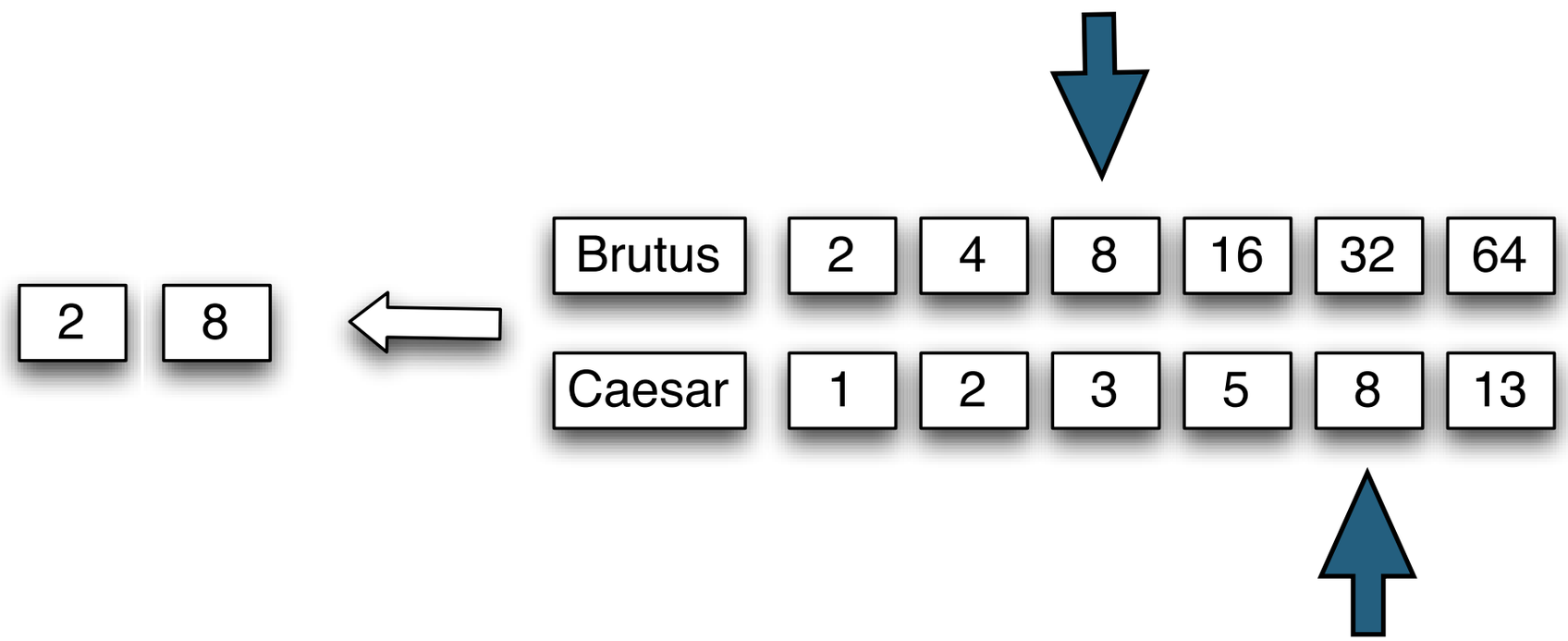
BOOLEAN RETRIEVAL

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



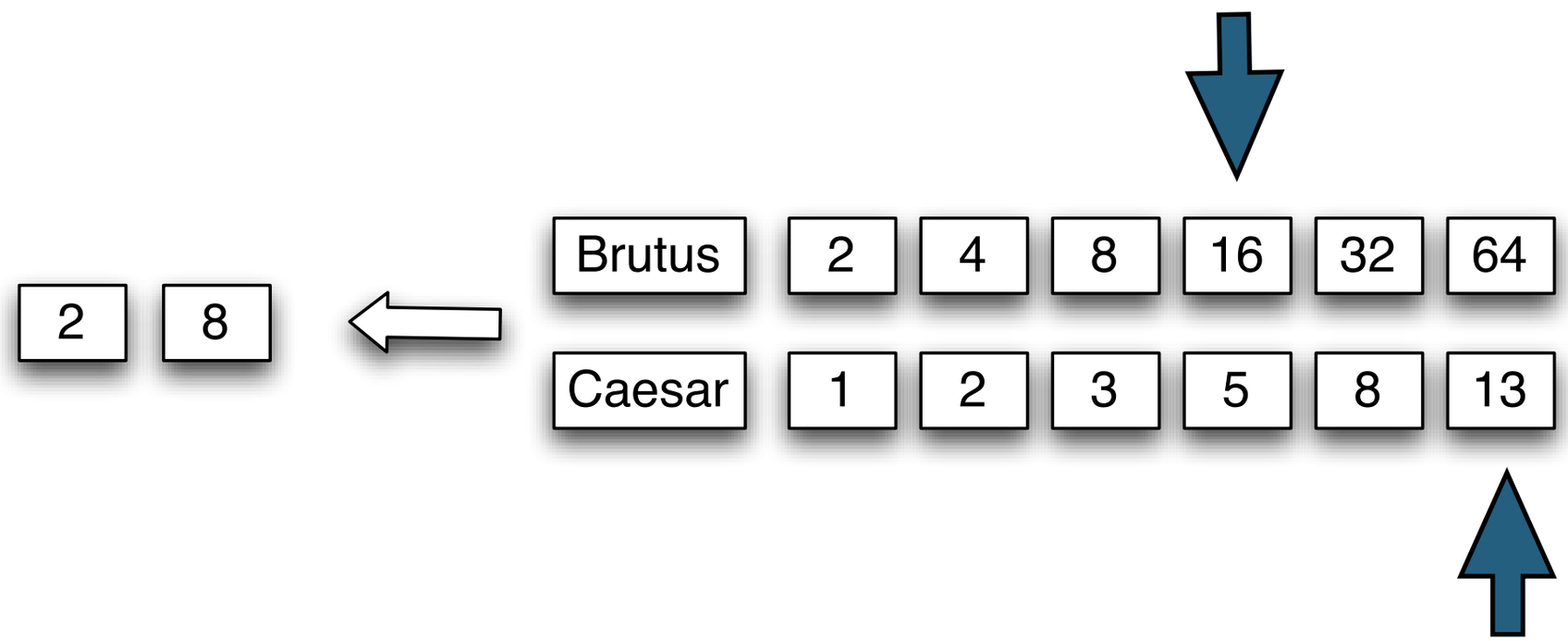
BOOLEAN RETRIEVAL

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



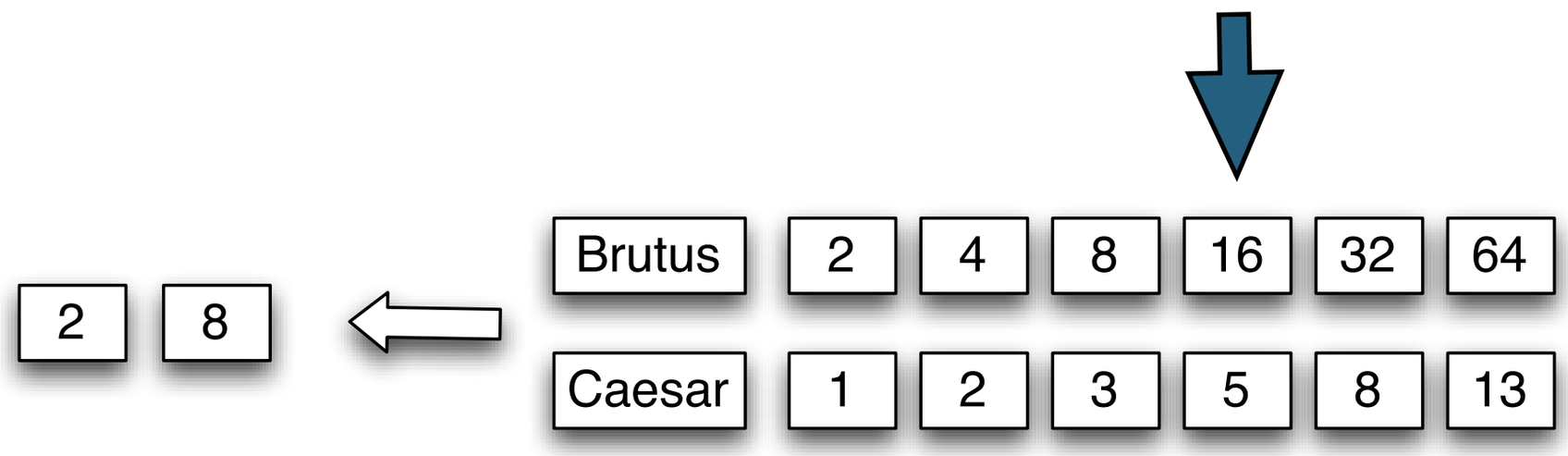
BOOLEAN RETRIEVAL

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



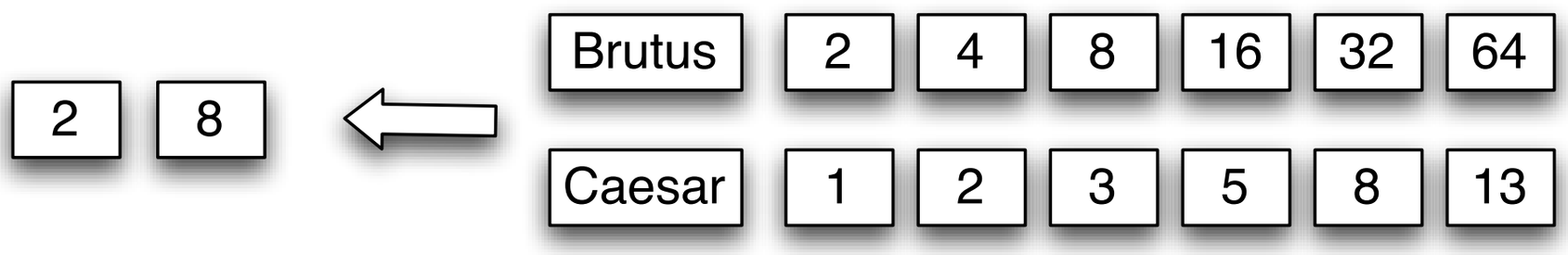
BOOLEAN RETRIEVAL

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



BOOLEAN RETRIEVAL

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



BOOLEAN RETRIEVAL

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

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

```
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
```

BOOLEAN RETRIEVAL

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

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

```
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
```

BOOLEAN RETRIEVAL

- 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
    - Why?
      - No question about what you are getting

BUILDING UP OUR QUERY TECHNOLOGY

- Linear on-demand retrieval (aka grep)
- 0/1 Vector-Based Boolean Queries
- Posting-Based Boolean Queries

BUILDING UP OUR QUERY TECHNOLOGY

- Linear on-demand retrieval (aka grep)
- 0/1 Vector-Based Boolean Queries
- Posting-Based Boolean Queries

- Deconstruct what is happening here:
  - http://www.rhymezone.com/shakespeare/

QUERYING

BOOLEAN MODEL VS. RANKED RETRIEVAL METHODS

- Only game for 30 years
- uses precise queries
- user decides relevance
- stayed current with proximity queries
- precise controlled queries
- transparent queries
- controlled queries

- Appeared with www
- uses “free-text” queries
- system decides relevance
- works with enormous corpora
- “no guarantees” in queries

QUERYING - BOOLEAN SEARCH EXAMPLE

WESTLAW CASE STUDY

- Largest commercial (paying subscribers) legal search service (started in 1975, ranking added in 1992)
- Tens of terabytes of data, 700,000 users
- Majority of users still use boolean queries (default in 2005)
  - Example:
    - What is the statute of limitations in cases involving federal tort claims act?
    - LIMIT! /3 STATUTE ACT /S FEDERAL /2 TORT /3 CLAIM
    - /3 = within 3 words. /S same sentence

QUERYING - BOOLEAN SEARCH EXAMPLE

WESTLAW CASE STUDY

- Example:
  - Requirements for disabled people to be able to access a workplace
  - disabl! /p access! /s work-site work-place employment /3 place
  - space is a disjunction not a conjunction
  - long precise queries, proximity operators, incrementally developed, not like web search
  - preferred by professionals, but not necessarily better

BUILDING UP OUR QUERY TECHNOLOGY

- "Matching" search
  - Linear on-demand retrieval (aka grep)
  - 0/1 Vector-Based Boolean Queries
  - Posting-Based Boolean Queries
- Ranked search
  - Parametric Search

QUERYING

RANKED SEARCH

- Rather than saying
  - (query, document) matches or not (0,1)
    - (“Capulet”,”Romeo and Juliet”) = 1
- Now we are going to assign rankings
  - (query, document) in {0,1}
    - (“capulet”,”Romeo and Juliet”) = 0.7

QUERYING

METADATA

- “structured additional information about a document.”
  - The author of a document
  - The creation date of a document
  - The title of a document
  - The location where a document was created
- author, creation date, title, location are fields
- searching for “William Shakespeare” in a doc differs from searching for “William Shakespeare” in the author of a doc

QUERYING

PARAMETRIC SEARCH

- supports searching on meta-data explicitly
- a parametric search interface allows a mix of full-text query and meta-data queries
- Example:
  - http://carsquare.com/

QUERYING

PARAMETRIC SEARCH

- Result is a large table
- Columns are fields
- Searching for “Lamborghini” only applied to make/model field

New and Used Cars, Truck

www.carfinder.com/list.php

CarFinder.com

Home | Browse Inventory | Se

Matching Vehicles

Save	Year	Make/Model	Miles	Price	Photos	Body Style
☐	2013	Lamborghini Aventador 2dr Conv AWD Co	550	$560,065		Convertible
☐	2013	Lamborghini Gallardo Coupe	340	$179,980		2 Door Coupe
☐	2013	Lamborghini Gallardo Coupe	90	$186,245		2 Door Coupe
☐	2013	Lamborghini Gallardo 2DR CPE LP560-2	1,330	$199,888		2 Door Coupe
☐	2013	Lamborghini Aventador 2dr Conv	716	$529,880		Convertible

Page: 1

Compare Saved | Clear Saved | Print List

QUERYING

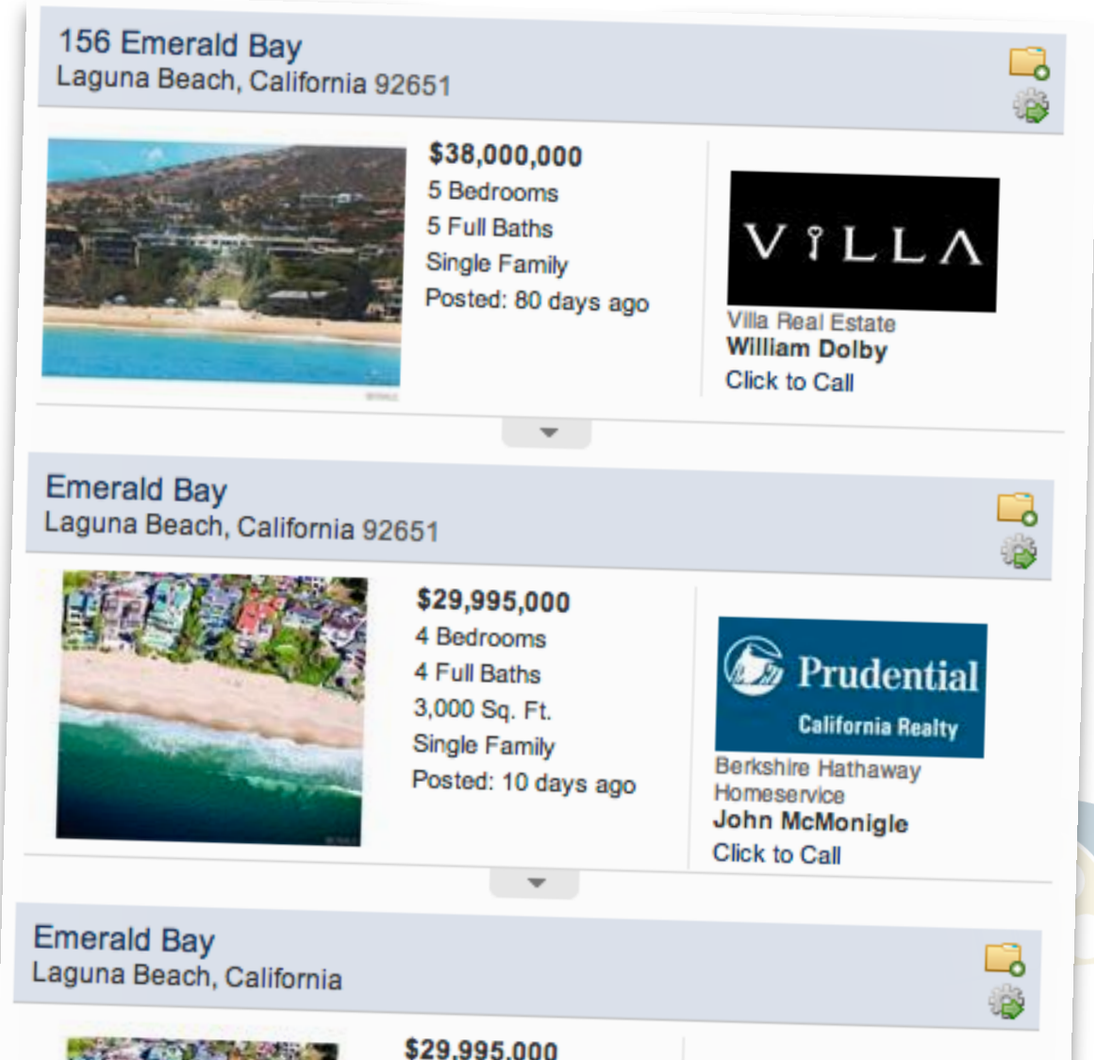
PARAMETRIC SEARCH

- Example:
  - http://www.ocregister.com/realestate/

QUERYING

PARAMETRIC SEARCH

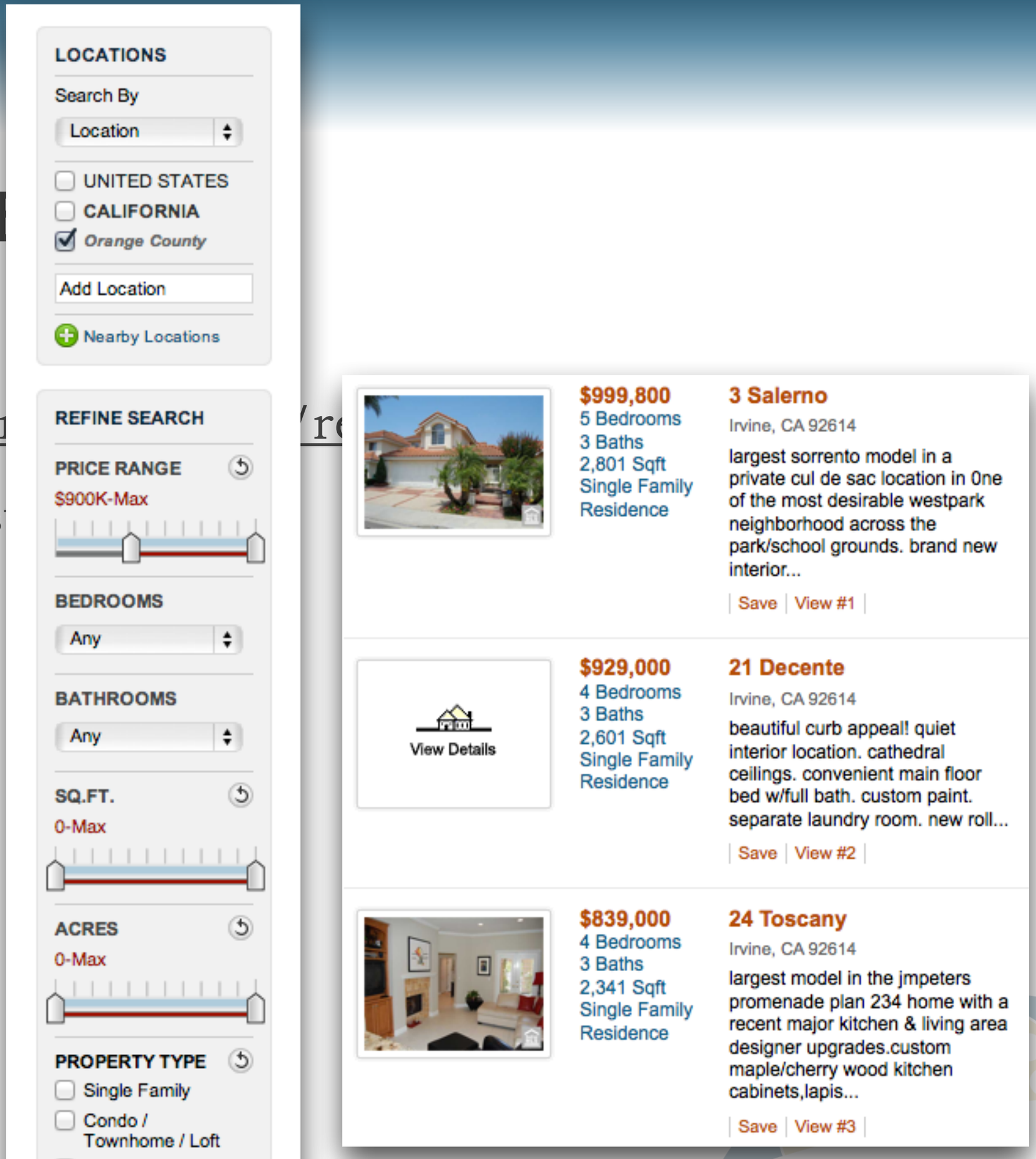
- Example:
  - http://www.ocregister.com/realestate/
  - 92614: 218 results



QUERYING

PARAMETRIC SE

- Example:
  - http://www.oc /re
  - 92614: 218 res



PARAMETRIC SEARCH

- In these examples we select field values
  - Values could be hierarchical
    - USA -> California -> SB County -> Montecito
- It is a paradigm for navigating through a corpus
  - e.g, “Aerospace companies in Brazil” can be found by combining “Geography” and “Industry”
- Approach:
  - Filter for relevant documents
  - Run text searches on subset

PARAMETRIC SEARCH

- Index support for parametric search
  - Must be able to support queries of the form:
    - Find pdf documents that contain “Westmont”
      - Field selection and text query
- Field selection approach
  - Use inverted index of field values
    - (field value, docID)
    - organized by field name
    - Using same compression and sorting techniques

BUILDING UP OUR QUERY TECHNOLOGY

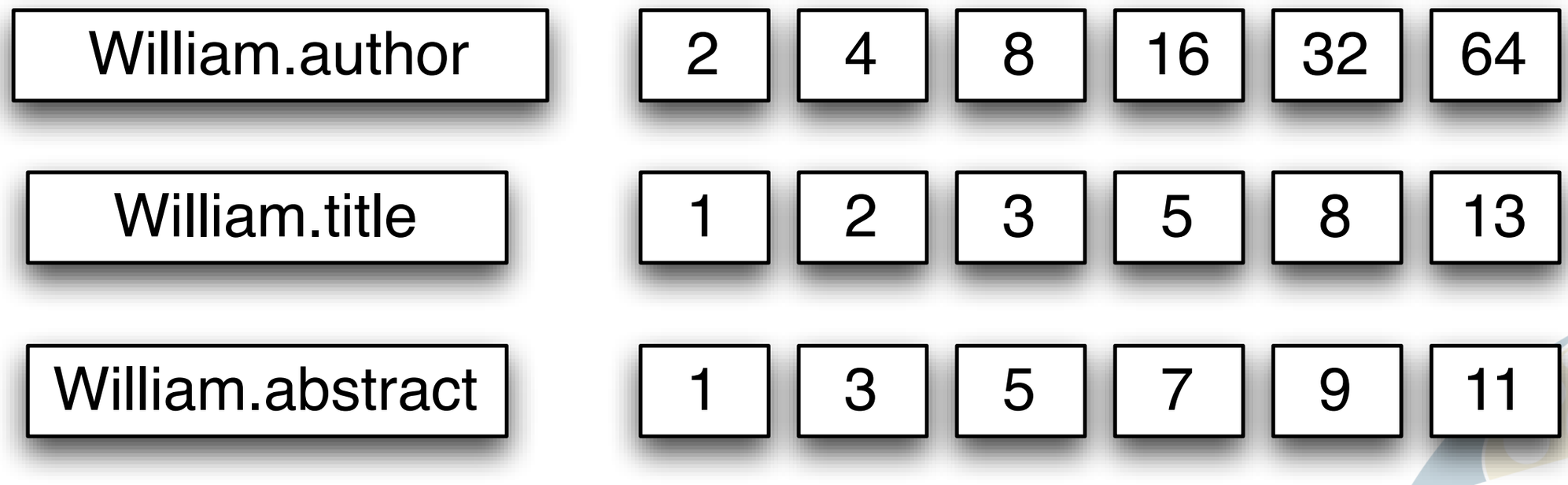
- "Matching" search
  - Linear on-demand retrieval (aka grep)
  - 0/1 Vector-Based Boolean Queries
  - Posting-Based Boolean Queries
- Ranked search
  - Parametric Search
  - Zones

ZONES

- A zone is an extension of a field
- A zone is an identified region of a document
  - e.g., title, abstract, bibliography
  - Generally identified by mark-up in a document
    - <title>Romeo and Juliet</title>
- Contents of zone are free text
  - Not a finite vocabulary
- Indices required for each zone to enable queries like:
  - (instant in TITLE) AND (oatmeal in BODY)
- Doesn't cover "all papers whose authors cite themselves"
  - Why?

PARAMETRIC/ZONE SEARCH

- Now, we crawl the corpus
- We parse the document keeping track of terms, fields and docIDs
- Instead of building just a (term, docID) pair
- We build (term, field, docID) triples
- These can then be combined into postings like this:

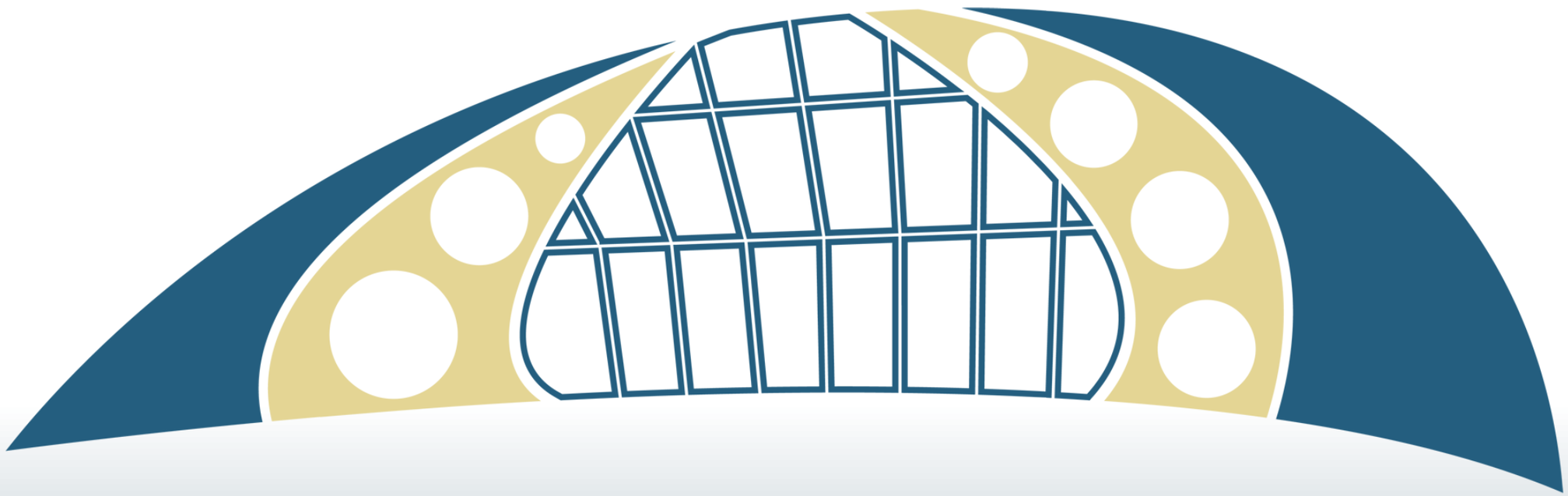


PARAMETRIC/ZONE SEARCH

- So are we just creating a database?
  - Not really.
  - Databases have more functionality
    - Transactions
    - Recovery
      - Our index can be recreated. Not so with database.
    - Text is never stored outside of indices
- We are focusing on optimized indices for text-oriented queries not a full SQL engine

BUILDING UP OUR QUERY TECHNOLOGY

- "Matching" search
  - Linear on-demand retrieval (aka grep)
  - 0/1 Vector-Based Boolean Queries
  - Posting-Based Boolean Queries
- Ranked search
  - Parametric Search
  - Zones
  - Scoring



WESTMONT INSPIRED

COMPUTING LAB