Querying

Introduction to Information Retrieval
INF 141/ CS 121
Donald J. Patterson

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

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

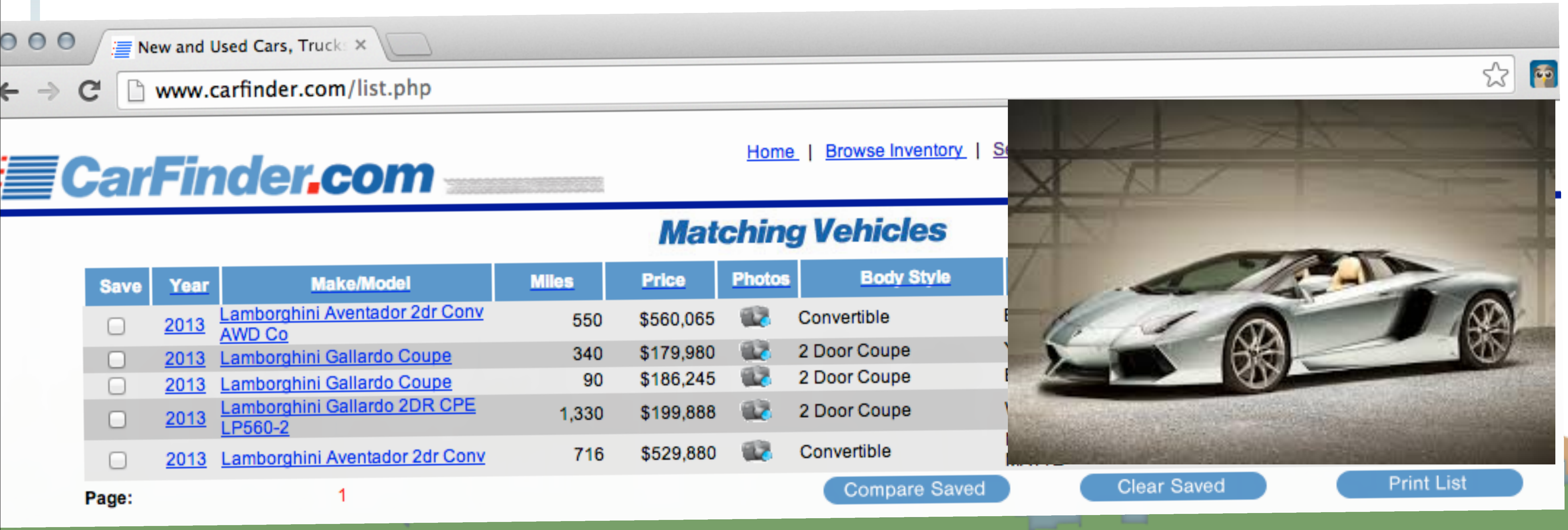
- Metadata = structured additional information about a document.
  - Examples:
    - The author of a document
    - The creation date of a document
    - The title of a document
    - The location where a document was created
  - author, creation date, title, location are fields
  - searching for “William 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:
    - www.carfinder.com

Querying

- Parametric Search
    - Example:
        - Result is a large table
        - Columns are fields
        - Searching for “2013” only applied to year field
        - www.carfinder.com



New and Used Cars, Truck ×
www.carfinder.com/list.php
CarFinder.com
Home | Browse Inventory | Se
Compare Saved
Clear Saved
Print List

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

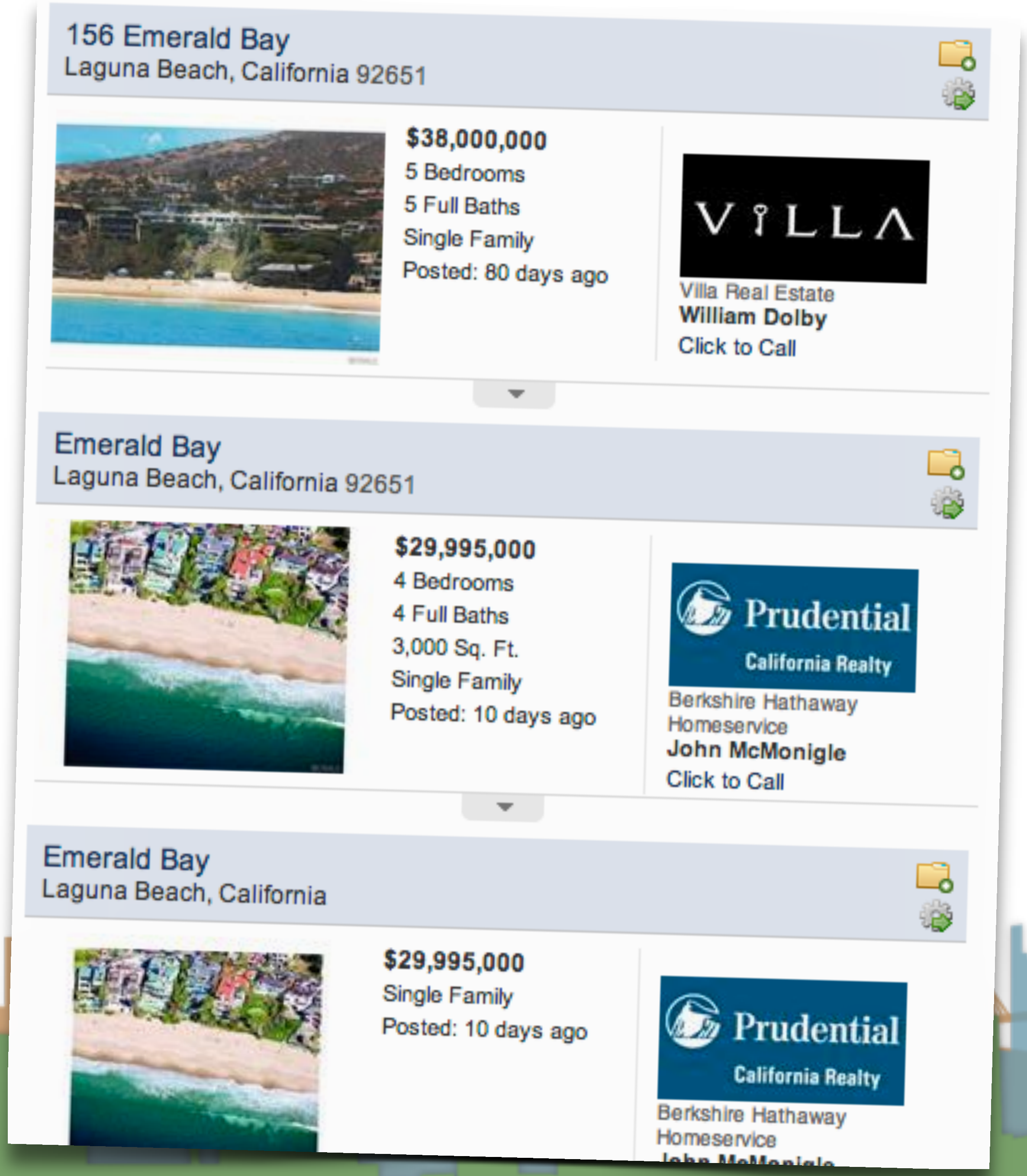
Page: 1

Querying

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

Querying

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



156 Emerald Bay
Laguna Beach, California 92651

$38,000,000
5 Bedrooms
5 Full Baths
Single Family
Posted: 80 days ago

Villa Real Estate
William Dolby
Click to Call

Emerald Bay
Laguna Beach, California 92651

$29,995,000
4 Bedrooms
4 Full Baths
3,000 Sq. Ft.
Single Family
Posted: 10 days ago

Berkshire Hathaway Homeservice
John McMonigle
Click to Call

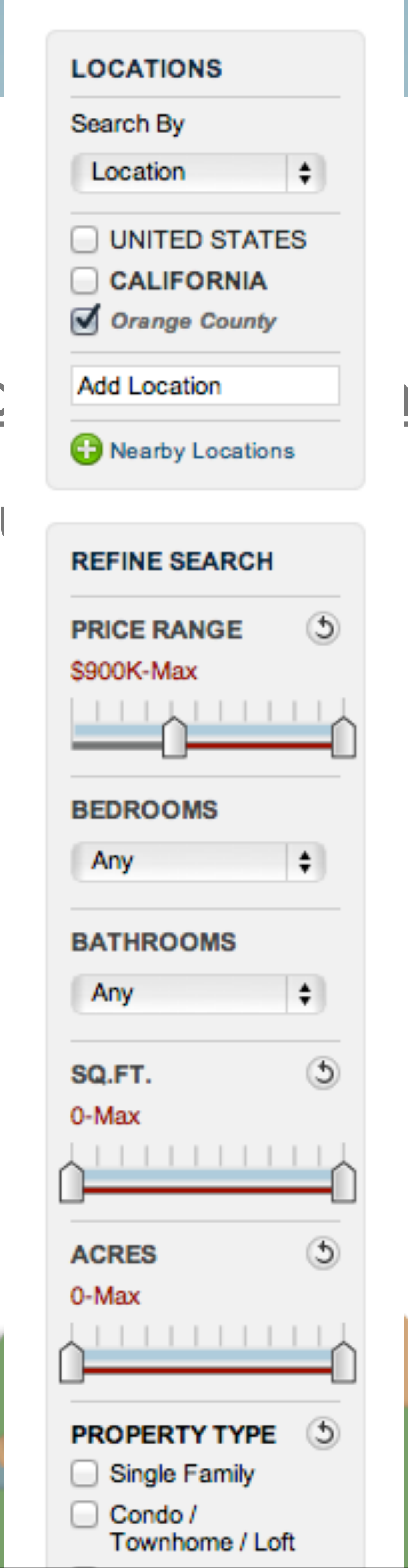
Emerald Bay
Laguna Beach, California

$29,995,000
Single Family
Posted: 10 days ago

Berkshire Hathaway Homeservice

Querying

- Parametric Search
  - Example:
    - http://www.c n/realestate/
    - 92614: 218 resu



LOCATIONS

Search By

Location

□ UNITED STATES

□ CALIFORNIA

☑ *Orange County*

Add Location

Nearby Locations

REFINE SEARCH

PRICE RANGE

$900K-Max

BEDROOMS

Any

BATHROOMS

Any

SQ.FT.

0-Max

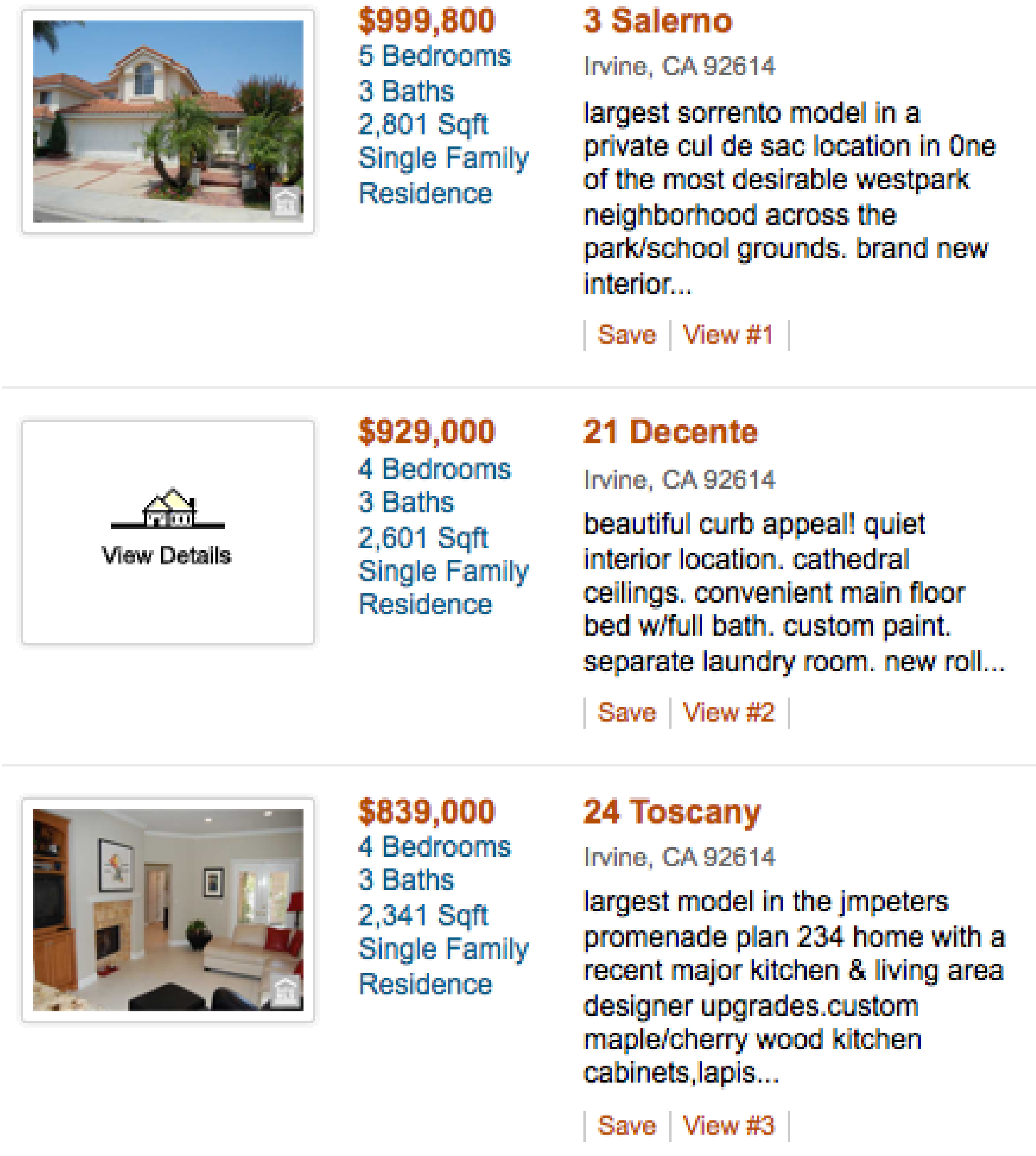
ACRES

0-Max

PROPERTY TYPE

□ Single Family

□ Condo / Townhome / Loft



$999,800
5 Bedrooms
3 Baths
2,801 Sqft
Single Family Residence

3 Salerno
Irvine, CA 92614
largest sorrento model in a private cul de sac location in 0ne of the most desirable westpark neighborhood across the park/school grounds. brand new interior...
Save | View #1

View Details

$929,000
4 Bedrooms
3 Baths
2,601 Sqft
Single Family Residence

21 Decente
Irvine, CA 92614
beautiful curb appeal! quiet interior location. cathedral ceilings. convenient main floor bed w/full bath. custom paint. separate laundry room. new roll...
Save | View #2

$839,000
4 Bedrooms
3 Baths
2,341 Sqft
Single Family Residence

24 Toscany
Irvine, CA 92614
largest model in the jmpeters promenade plan 234 home with a recent major kitchen & living area designer upgrades.custom maple/cherry wood kitchen cabinets,lapis...
Save | View #3

Parametric Search

- In these examples we select field values
  - Values could be hierarchical
    - USA -> California -> Orange County -> Newport Beach
- 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 “UCI”
        - 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:

William.author	2	4	8	16	32	64
William.title	1	2	3	5	8	13
William.abstract	1	3	5	7	9	11

Parametric 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

Querying

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

Scoring

- Boolean queries “match” or “don’t match”
- Good for experts with needs for precision and coverage
  - knowledge of corpus
  - need 1000’s of results
- Not good with non-expert users
  - who don’t understand boolean operators
  - or how they apply to search
  - or who don’t want 1000’s of results

Scoring

- Boolean queries require careful crafting to get the right number of results (Ferrari example)
- Ranked lists eliminate this concern
  - Doesn’t matter how big the list is
- Scoring is the basis for ranking or sorting document that are returned from a query.
  - Ideally the score is high when the document is relevant
  - WLOG we will assume scores are between 0 and 1 for each doc.

Scoring

- First generation of scoring used a linear combination of Booleans

$$\begin{aligned} Score \quad = \quad & 0.6(oatmeal \in TITLE) + \\ & 0.3(oatmeal \in BODY) + \\ & 0.1(oatmeal \in ABSTRACT) \end{aligned}$$

- Explicit decision about importance of zone
- Each subquery is 0 or 1
- This example has a finite number of possible values
  - What are they?

Scoring

$$\begin{aligned} Score &= 0.6(oatmeal \in TITLE) + \\ &\quad 0.3(oatmeal \in BODY) + \\ &\quad 0.1(oatmeal \in ABSTRACT) \end{aligned}$$

- Subqueries could be *any* Boolean query
- Where do we get the weights? (e.g., 0.6,0.3,0.1)
  - Rarely from the user
  - Usually built into the query engine
    - Where does the query engine get them from?
      - Machine learning

Scoring Exercise

- Calculate the score for each document based on the weightings (0.1 author), (0.3 body), (0.6 title)
- For the query
  - “bill” or “rights”

bill.author	1	2		
rights.author				
bill.title	3	5	8	
rights.title	3	5	9	
bill.body	1	2	5	9
rights.body	3	5	8	9

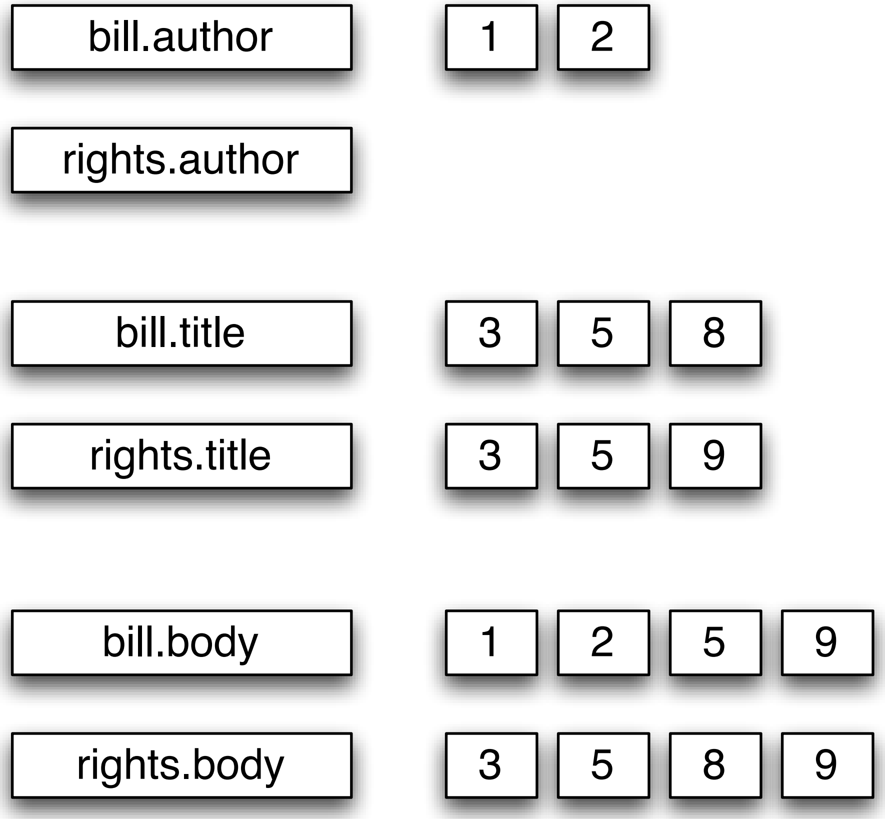
Building up our query technology

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

Querying

Zones combination index

- Encode the zone in the posting
- At query time accumulate the contributions to the total score from the various postings



bill.author 1 2

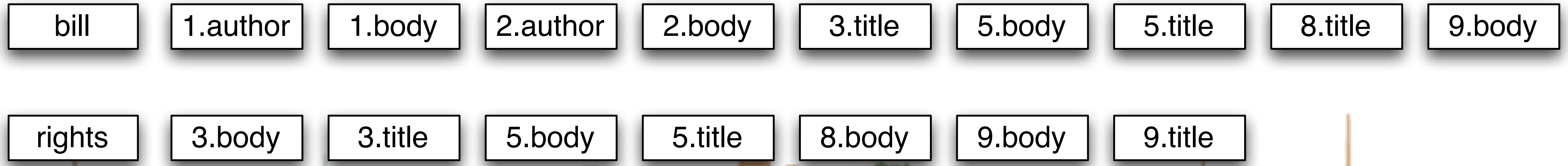
rights.author

bill.title 3 5 8

rights.title 3 5 9

bill.body 1 2 5 9

rights.body 3 5 8 9

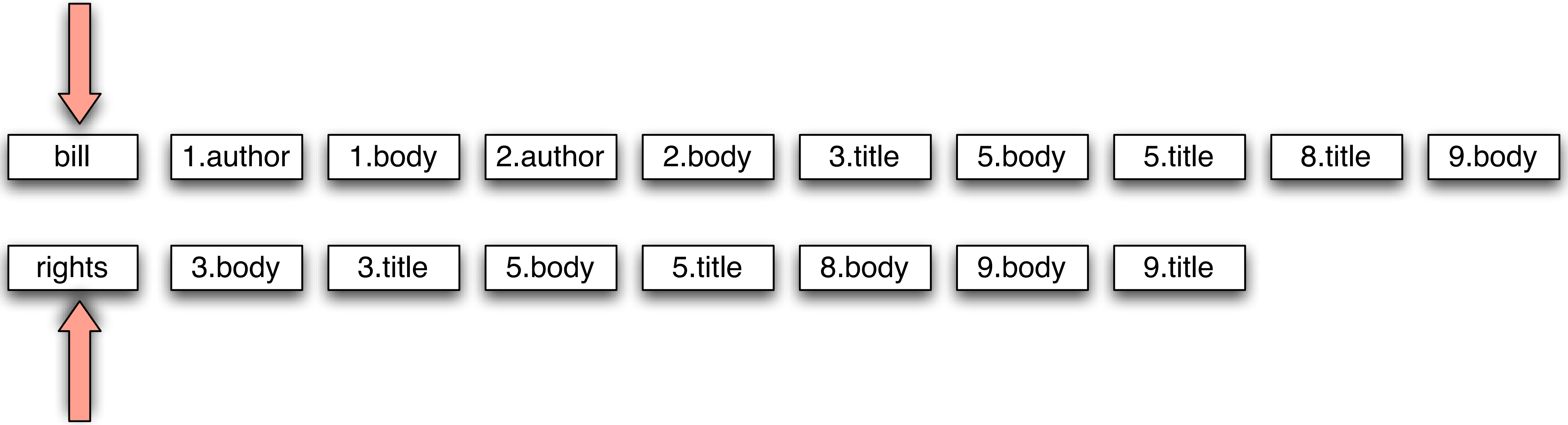


bill 1.author 1.body 2.author 2.body 3.title 5.body 5.title 8.title 9.body

rights 3.body 3.title 5.body 5.title 8.body 9.body 9.title

Zone scoring with zones combination index

"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



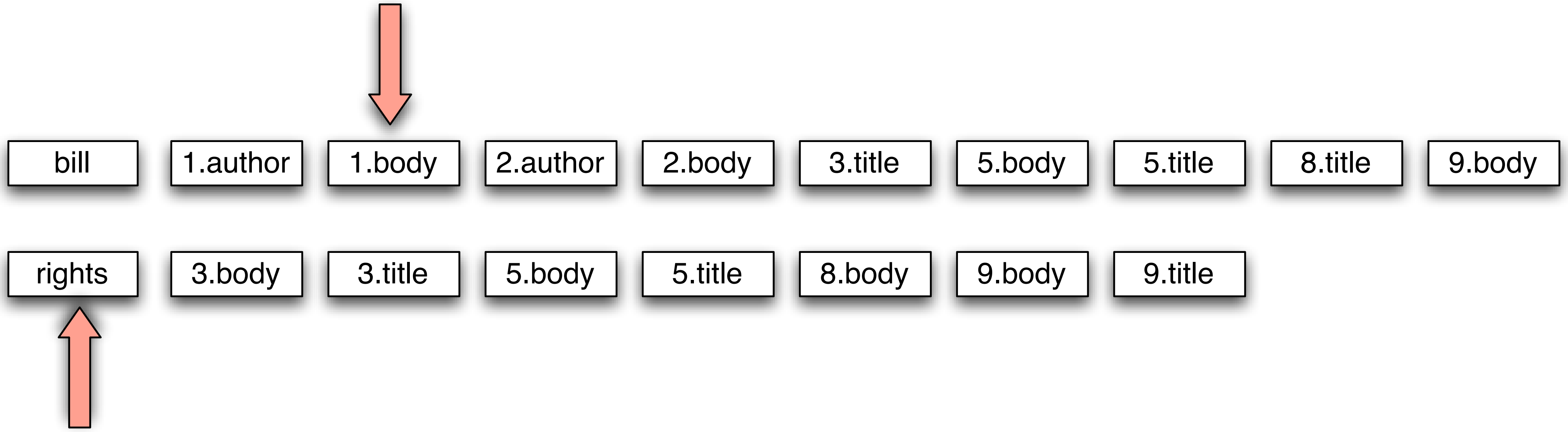
bill → 1.author → 1.body → 2.author → 2.body → 3.title → 5.body → 5.title → 8.title → 9.body

rights → 3.body → 3.title → 5.body → 5.title → 8.body → 9.body → 9.title

Querying

Zone scoring with zones combination index

"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



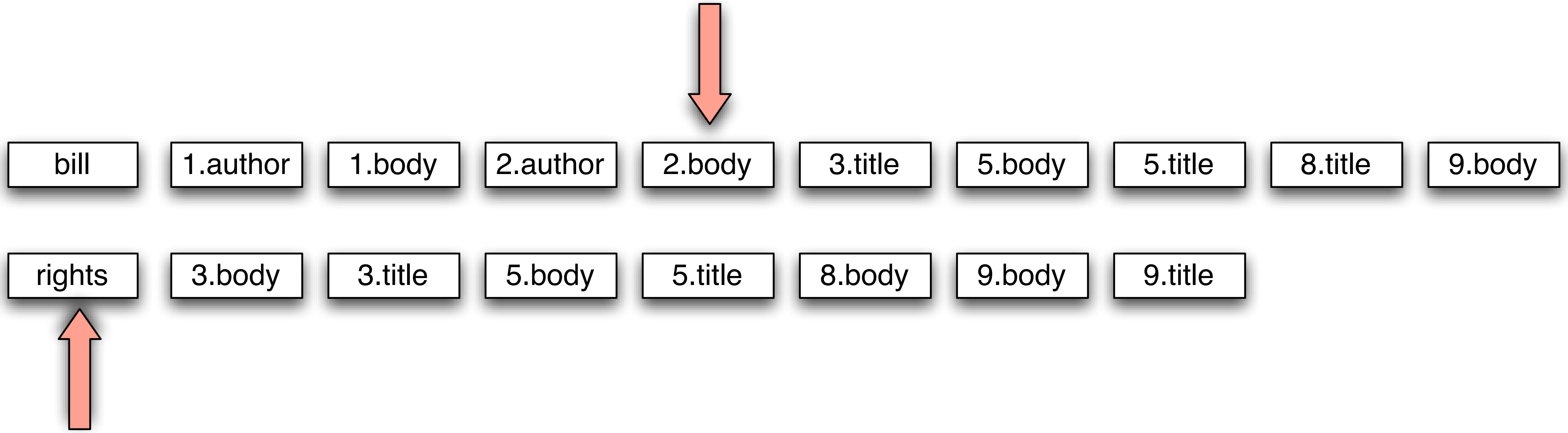
bill | 1.author | 1.body | 2.author | 2.body | 3.title | 5.body | 5.title | 8.title | 9.body

rights | 3.body | 3.title | 5.body | 5.title | 8.body | 9.body | 9.title

1: 0.4

Zone scoring with zones combination index

"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



bill → 1.author → 1.body → 2.author → 2.body → 3.title → 5.body → 5.title → 8.title → 9.body

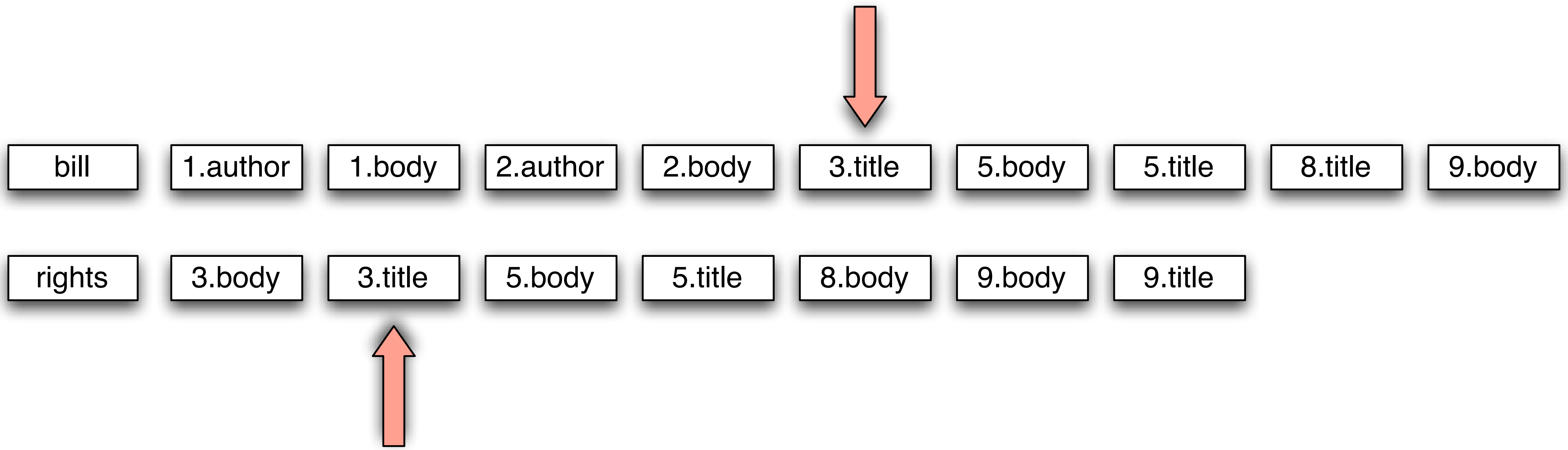
rights → 3.body → 3.title → 5.body → 5.title → 8.body → 9.body → 9.title

1: 0.4

2: 0.4

Zone scoring with zones combination index

“bill OR rights” (0.1 author), (0.3 body), (0.6 title)



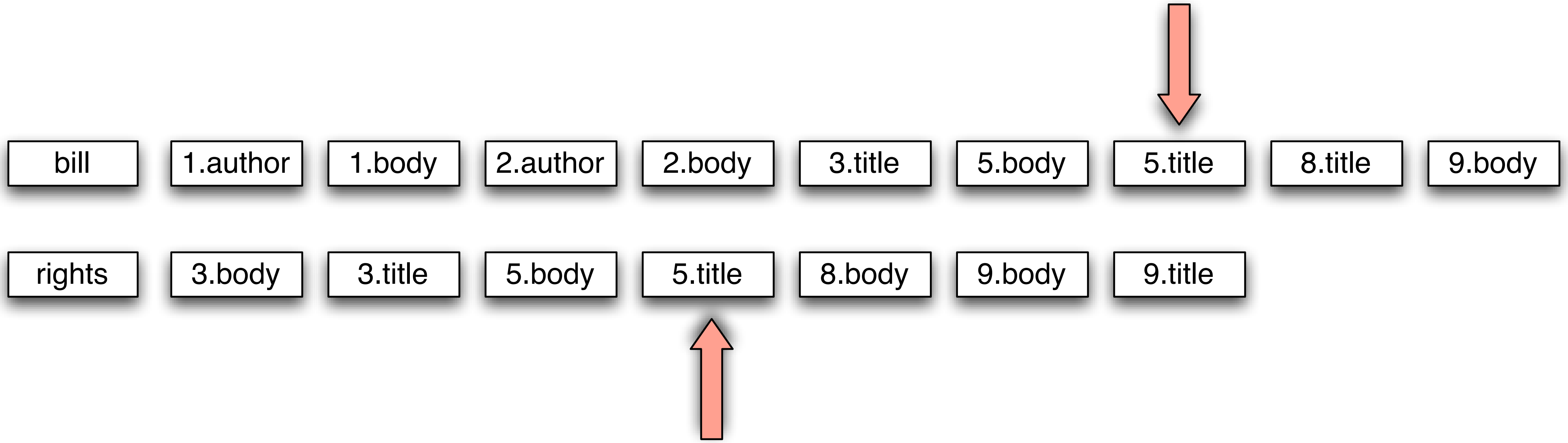
bill 1.author 1.body 2.author 2.body 3.title 5.body 5.title 8.title 9.body

rights 3.body 3.title 5.body 5.title 8.body 9.body 9.title

1: 0.4
2: 0.4
3: 0.9

Zone scoring with zones combination index

“bill OR rights” (0.1 author), (0.3 body), (0.6 title)



bill → 1.author → 1.body → 2.author → 2.body → 3.title → 5.body → 5.title → 8.title → 9.body

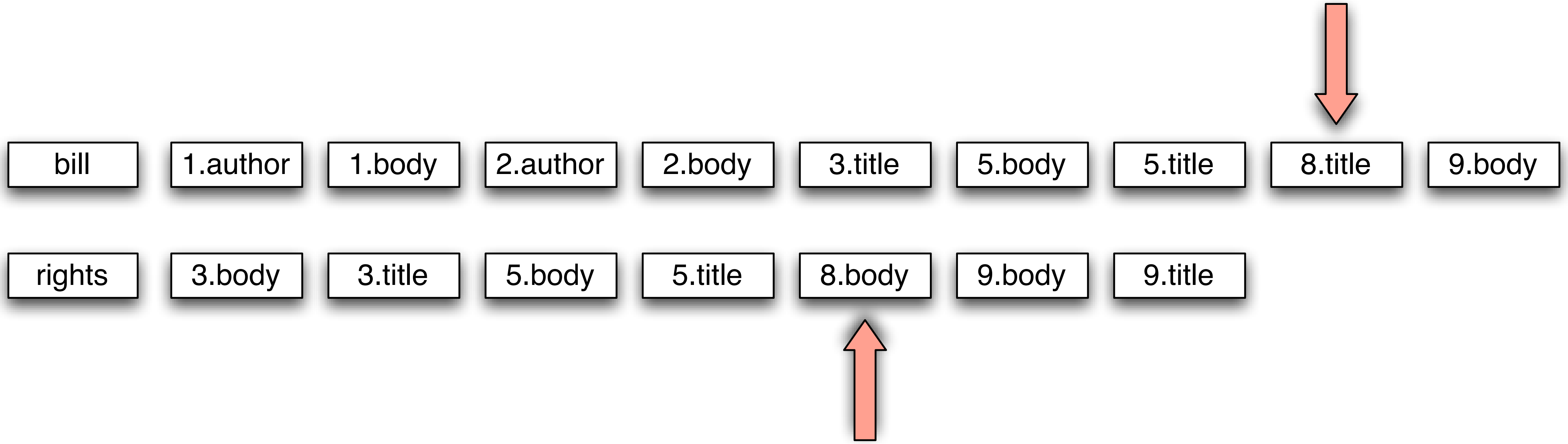
rights → 3.body → 3.title → 5.body → 5.title → 8.body → 9.body → 9.title

1: 0.4   5: 0.9
2: 0.4
3: 0.9

Querying

Zone scoring with zones combination index

“bill OR rights” (0.1 author), (0.3 body), (0.6 title)



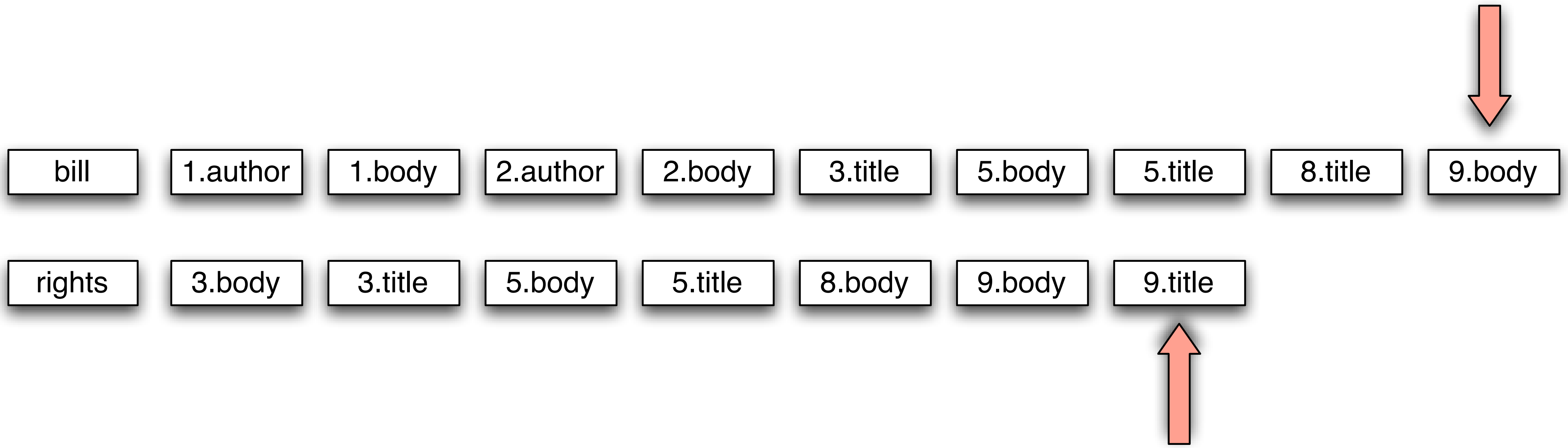
bill → 1.author, 1.body, 2.author, 2.body, 3.title, 5.body, 5.title, 8.title, 9.body

rights → 3.body, 3.title, 5.body, 5.title, 8.body, 9.body, 9.title

1: 0.4 5: 0.9
2: 0.4 8: 0.9
3: 0.9

Zone scoring with zones combination index

"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



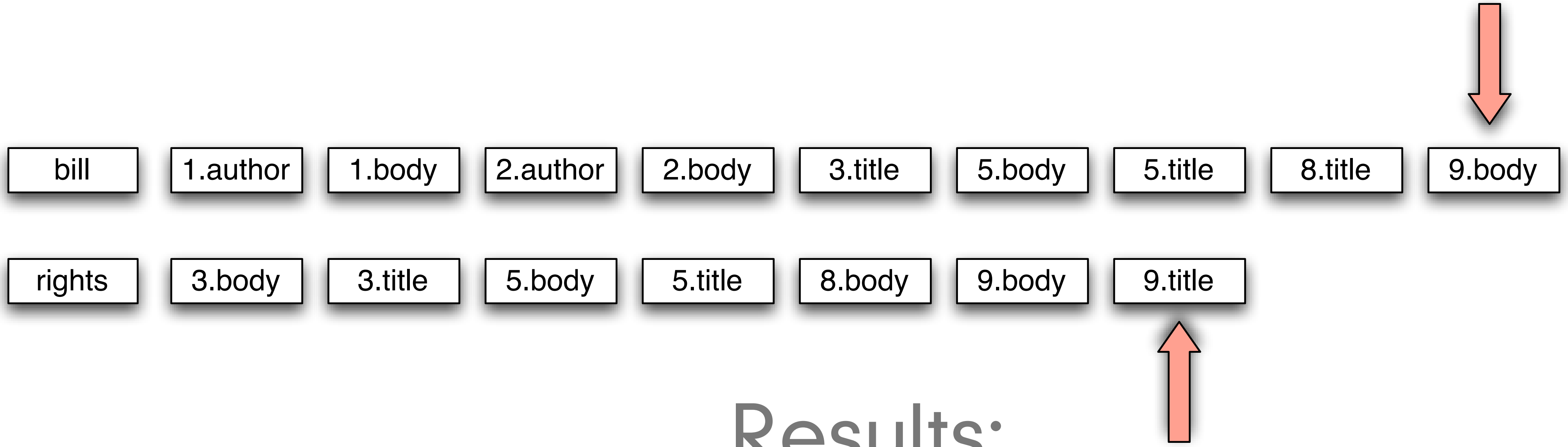
bill	1.author	1.body	2.author	2.body	3.title	5.body	5.title	8.title	9.body
rights	3.body	3.title	5.body	5.title	8.body	9.body	9.title		

1: 0.4 5: 0.9
2: 0.4 8: 0.9
3: 0.9 9: 0.9

Querying

Zone scoring with zones combination index

"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



bill → 1.author → 1.body → 2.author → 2.body → 3.title → 5.body → 5.title → 8.title → 9.body

rights → 3.body → 3.title → 5.body → 5.title → 8.body → 9.body → 9.title

1: 0.4
2: 0.4
3: 0.9
5: 0.9
8: 0.9
9: 0.9

Results:
9,8,5,3,2,1

Zone scoring with zones combination index

- As we walk, we accumulate scores linearly
- Note: getting “bill” and “rights” in the title field didn’t cause us to score any higher
  - Should it?
- Where do the weights come from?
  - Machine learning
    - Given a corpus, test queries and “gold standard” relevance scores, compute weights which come as close as possible to “gold standard”

Full text queries

- Previous example was for “bill OR rights”
- Average user is likely to type “bill rights” or “bill of rights”
  - How do we interpret such a query?
  - No Boolean operators
  - Some query terms might not be in the document
  - Some query terms might not be in a zone

Full text queries

- To use zone combinations for free text queries, we need:
  - A way of scoring = Score(full-text-query, zone)
  - Zero query terms in zone -> zero score
  - More query terms in a zone -> higher score
  - Scores don’t have to be boolean (0 or 1) anymore
- Let’s look at the alternatives...

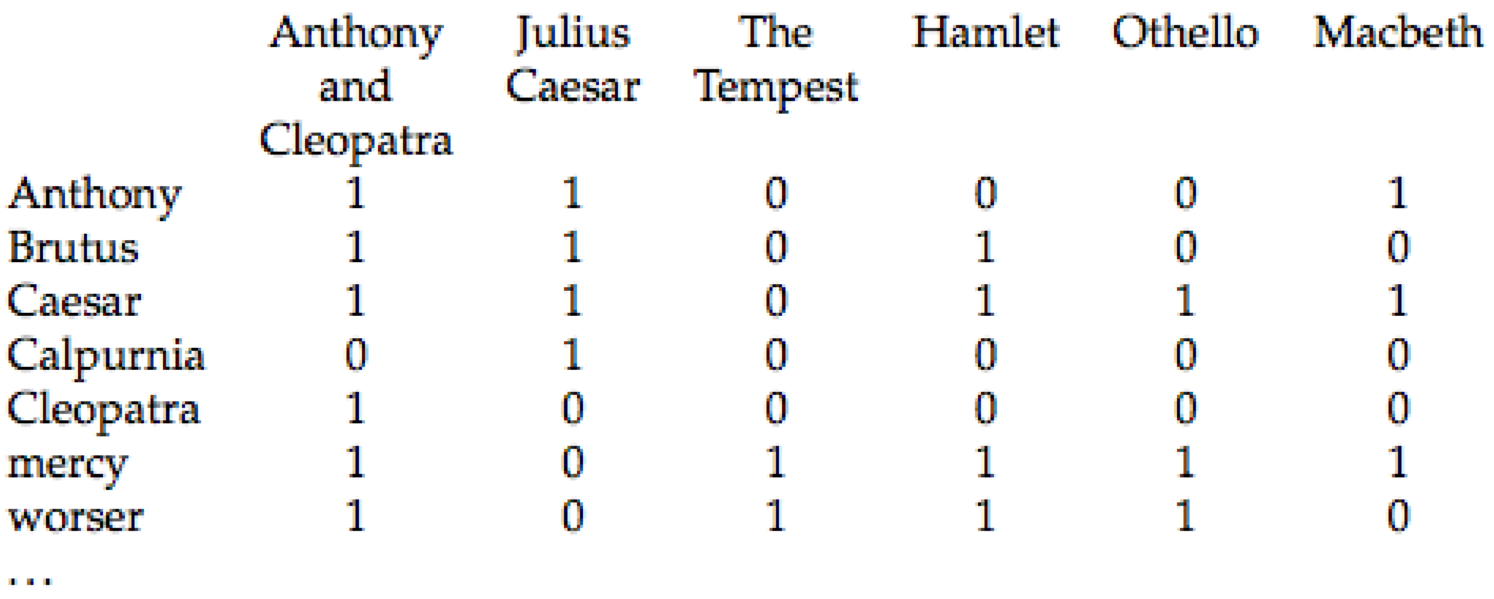
Querying

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
  - Term Frequency Matrices

Incidence Matrices

- Recall how a document, d, (or a zone) is a (0,1) column vector
  - A query, q, is also a column vector. How so?

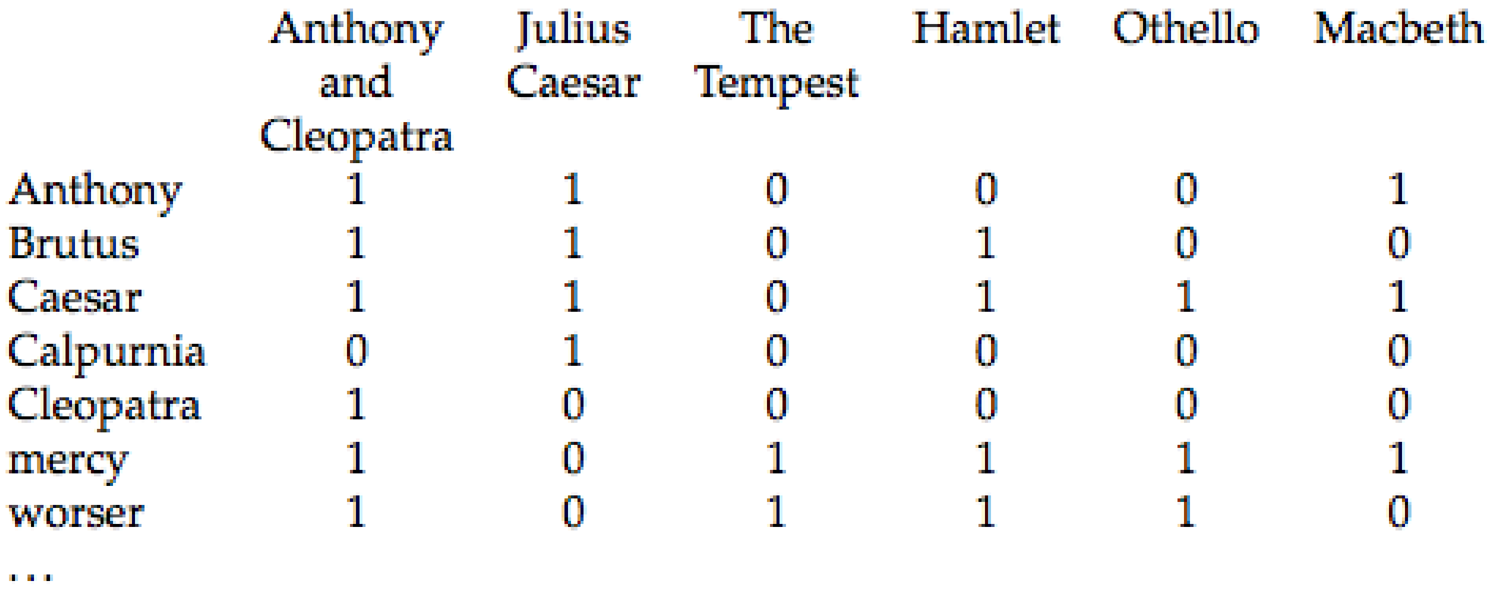


	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Anthony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0
...						

Incidence Matrices

- Using this formalism, score can be overlap measure:

$$|q \cap D|$$



	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
...						

Incidence Matrices

- Example:
    - Query “ides of march”
    - Shakespeare’s “Julius Caesar” has a score of 3
    - Plays that contain “march” and “of” score 2
    - Plays that contain “of” score 1
- Algorithm:
    - Bitwise-And between q and matrix, D
    - Column summation
    - Sort

Incidence Matrices

- What is wrong with the overlap measure?
- It doesn’t consider:
  - Term frequency in a document
  - Term scarcity in corpus
    - “ides” is much rarer than “of”
  - Length of a document
  - Length of queries

Toward better scoring

- Overlap Measure

$$|q \cap d|$$

- Normalizing queries
  - Jaccard Coefficient

$$\frac{|q \cap d|}{|q \cup d|}$$

    - Score is number of words that overlap divided by total number of words
    - What documents would score best?
  - Cosine Measure

$$\frac{|q \cap d|}{\sqrt{|q||d|}}$$

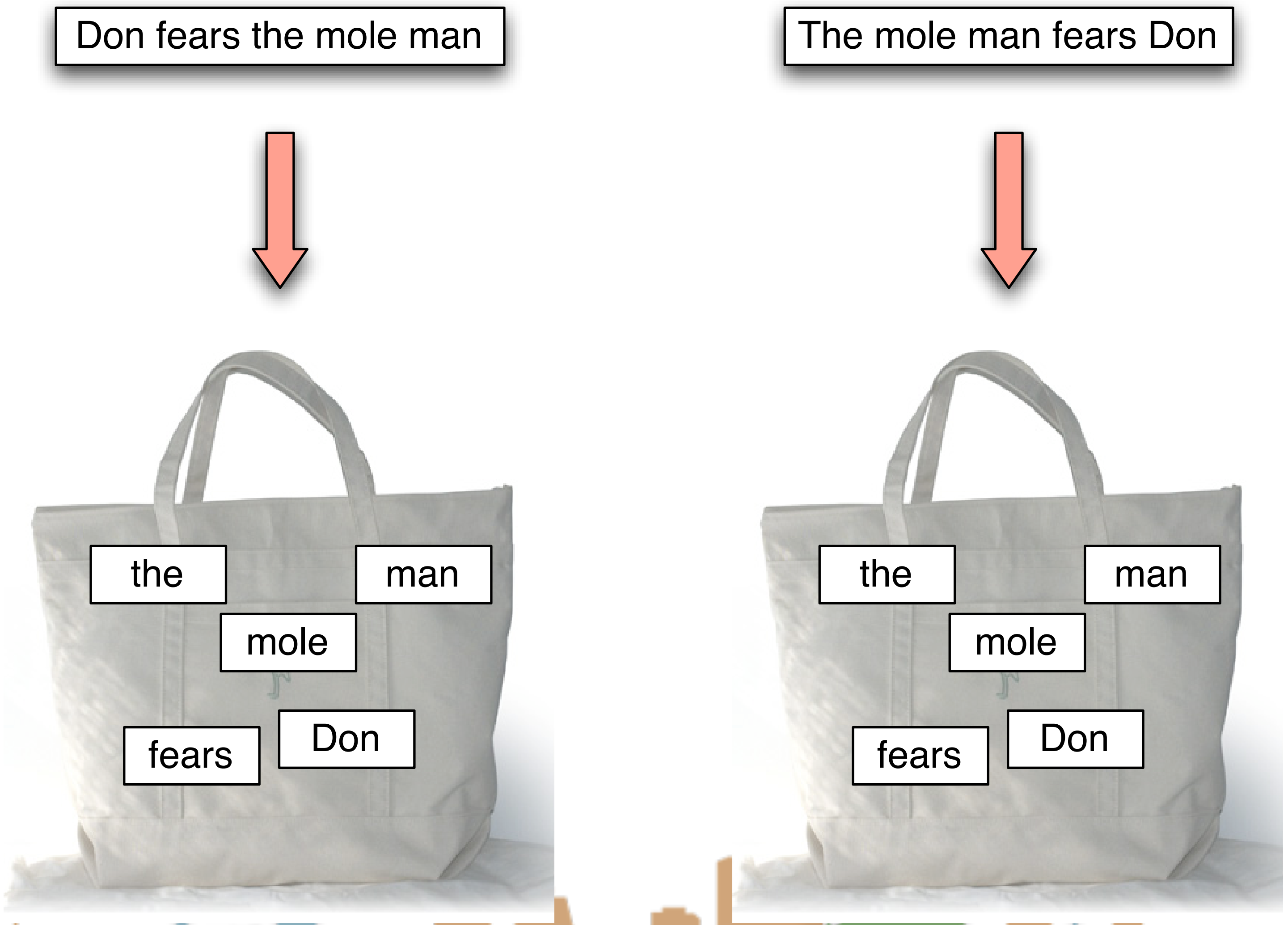
    - Will the same documents score well?

Toward Better Scoring

- Scores so far capture position (zone) and overlap
- Next step: a document which talks about a topic should be a better match
  - Even when there is a single term in the query
  - Document is relevant if the term occurs a lot
  - This brings us to term weighting

Bag of Words Model

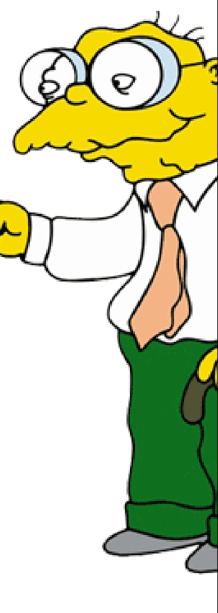
- "Don fears the mole man" equals "The mole man fears Don"
- The incidence matrix for both looks the same



Don fears the mole man

The mole man fears Don

$$\begin{array}{ccc} & d_1 & d_2 \\ Don & 1 & 1 \\ fears & 1 & 1 \\ man & 1 & 1 \\ mole & 1 & 1 \\ mule & 0 & 0 \\ the & 1 & 1 \\ zoo & 0 & 0 \end{array}$$



Term Frequency Matrix

- Bag of words
- Document is vector with integer elements

	Antony and Cleopatra	*Julius Caesar*	*The Tempest*	*Hamlet*	*Othello*	*Macbeth*
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Term Frequency - tf

- Long documents are favored because they are more likely to contain query terms
- Reduce the impact by normalizing by document length
- Is raw term frequency the right number?

Weighting Term Frequency - WTF

- What is the relative importance of
  - 0 vs. 1 occurrence of a word in a document?
  - 1 vs. 2 occurrences of a word in a document?
  - 2 vs. 100 occurrences of a word in a document?
- Answer is unclear:
  - More is better, but not proportionally
  - An alternative to raw tf:

```
$\mathrm{WTF}(t, d)$
if $tf_{t,d} = 0$
    then $return(0)$
    else $return(1 + log(tf_{t,d}))$
```