QUERYING

Introduction to Information Retrieval
CS 150
Donald J. Patterson

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

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

QUERYING

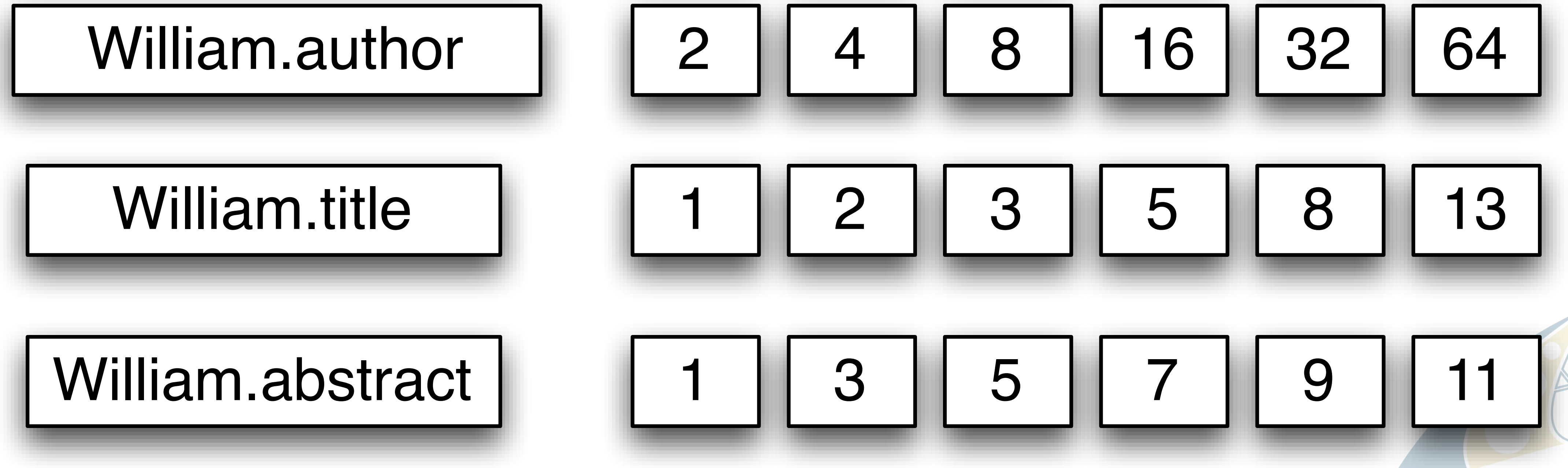
ZONES

- 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?

QUERYING

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

QUERYING

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

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

QUERYING

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

QUERYING

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.

WEIGHTED ZONE SCORING

- First generation of scoring used a linear combination of Booleans

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

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

QUERYING

WEIGHTED ZONE 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

QUERYING

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

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

QUERYING

ZONES COMBINATION INDEX

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

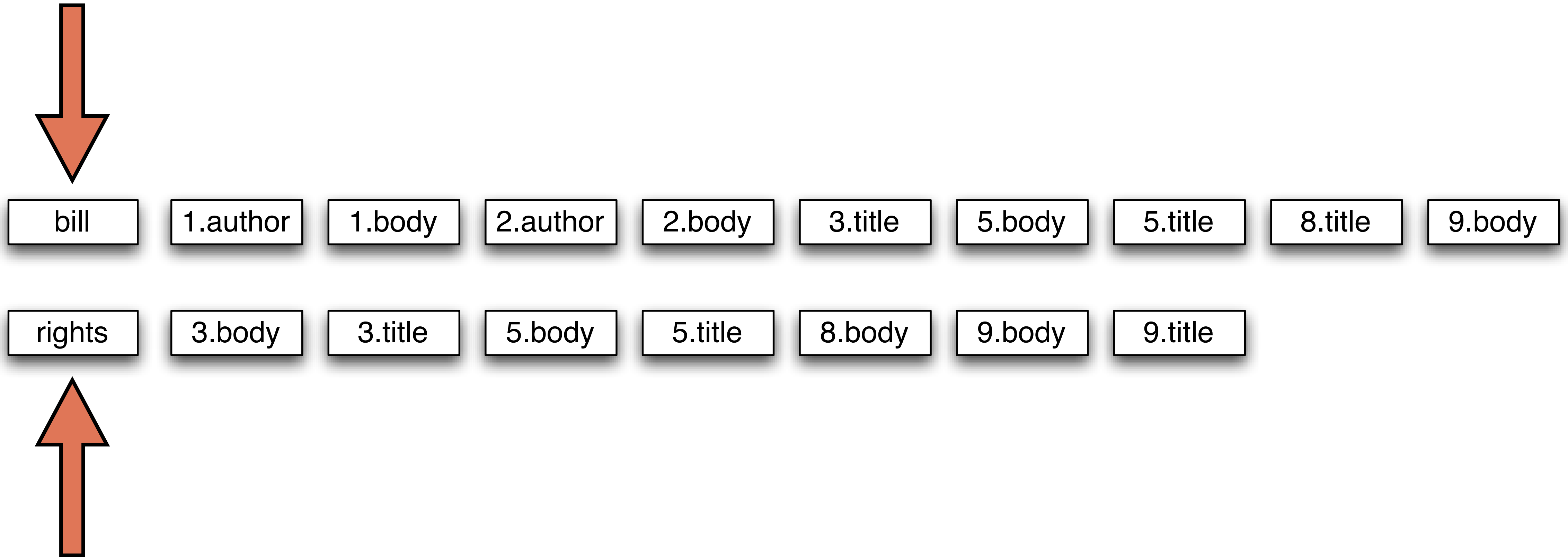
bill.author	1	2		
rights.author				
bill.title	3	5	8	
rights.title	3	5	9	
bill.body	1	2	5	9
rights.body	3	5	8	9

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		

QUERYING

SCORING WITH ZONES COMBINATION INDEX

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



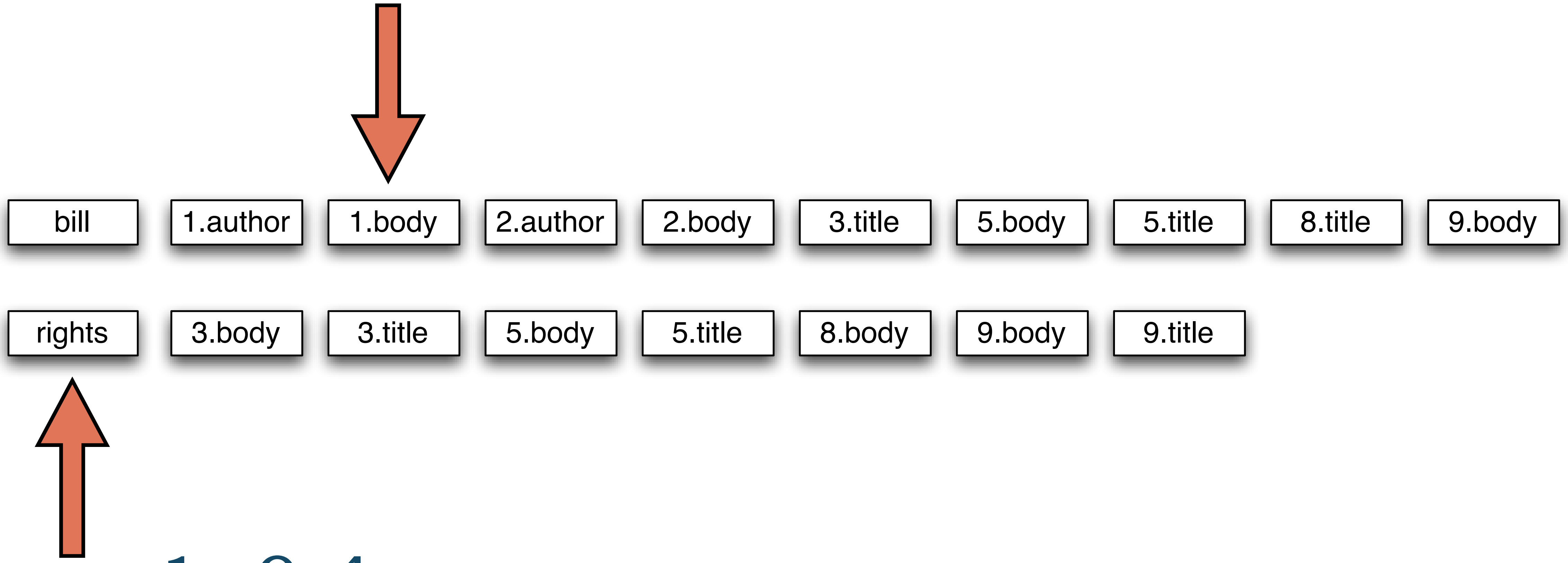
bill → 1.author → 1.body → 2.author → 2.body → 3.title → 5.body → 5.title → 8.title → 9.body

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

QUERYING

SCORING WITH ZONES COMBINATION INDEX

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



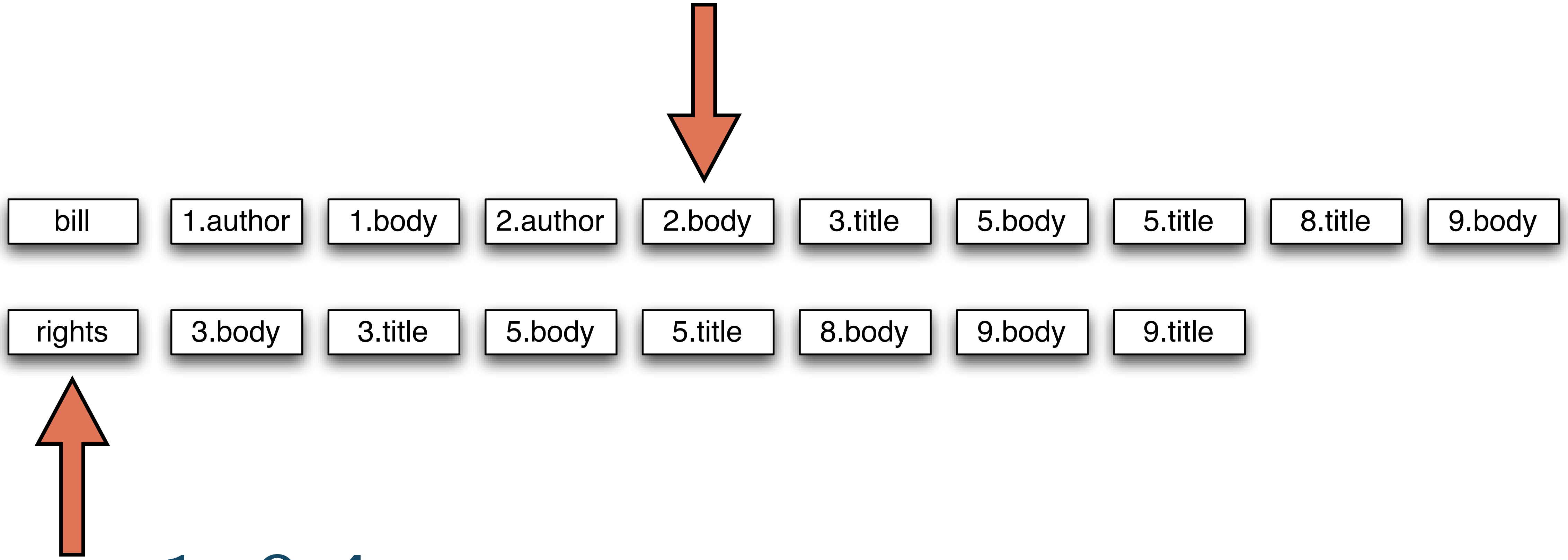
bill 1.author 1.body 2.author 2.body 3.title 5.body 5.title 8.title 9.body

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

1: 0.4

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

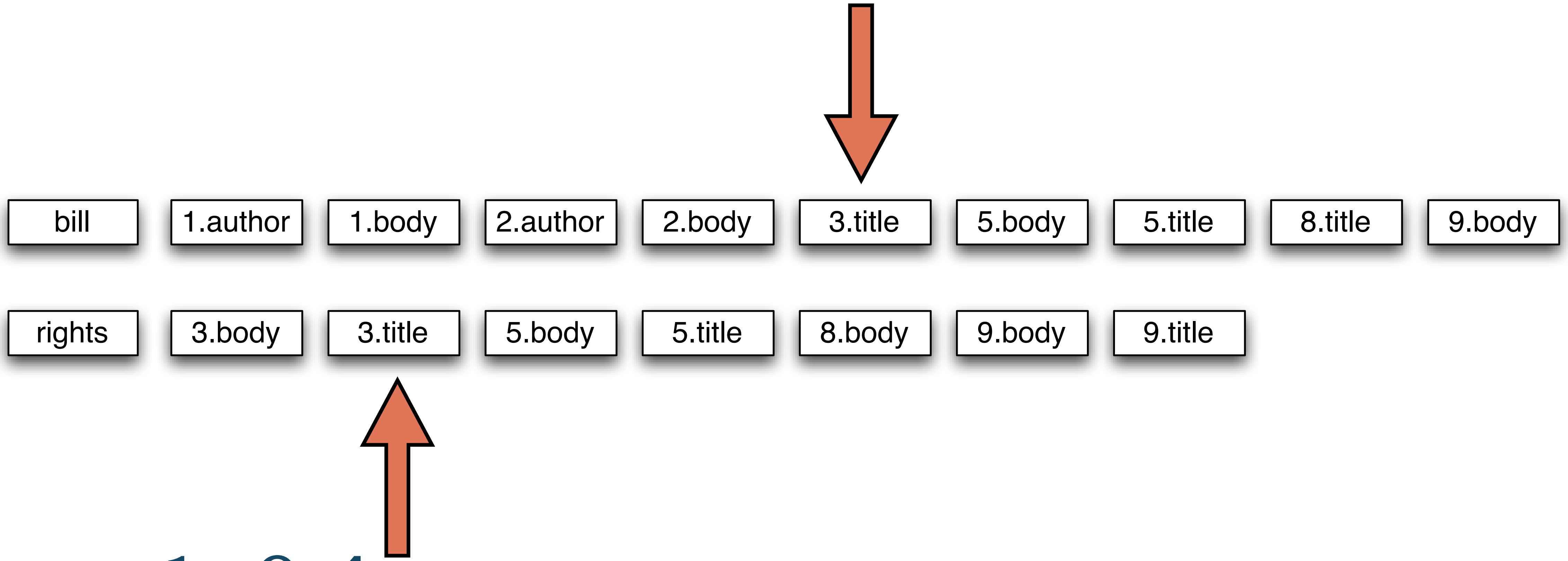
rights → 3.body, 3.title, 5.body, 5.title, 8.body, 9.body, 9.title

1: 0.4
2: 0.4

QUERYING

SCORING WITH ZONES COMBINATION INDEX

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



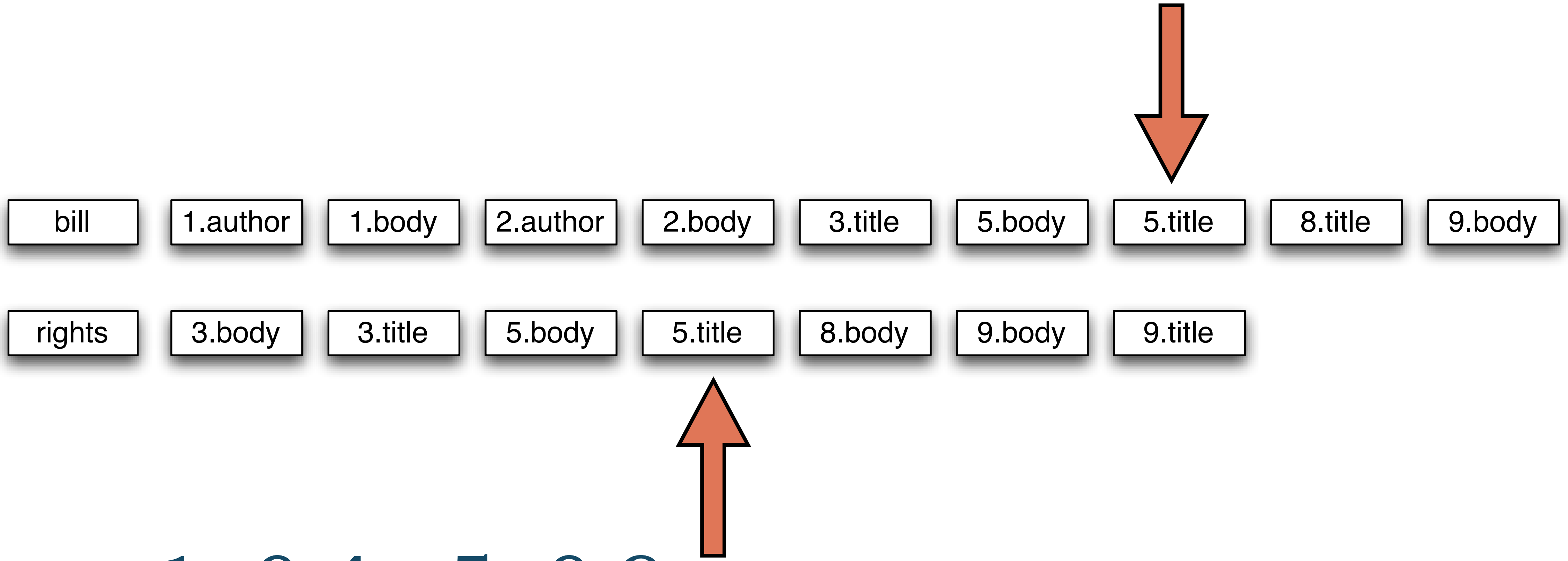
bill → 1.author, 1.body, 2.author, 2.body, 3.title, 5.body, 5.title, 8.title, 9.body

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

1: 0.4
2: 0.4
3: 0.9

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

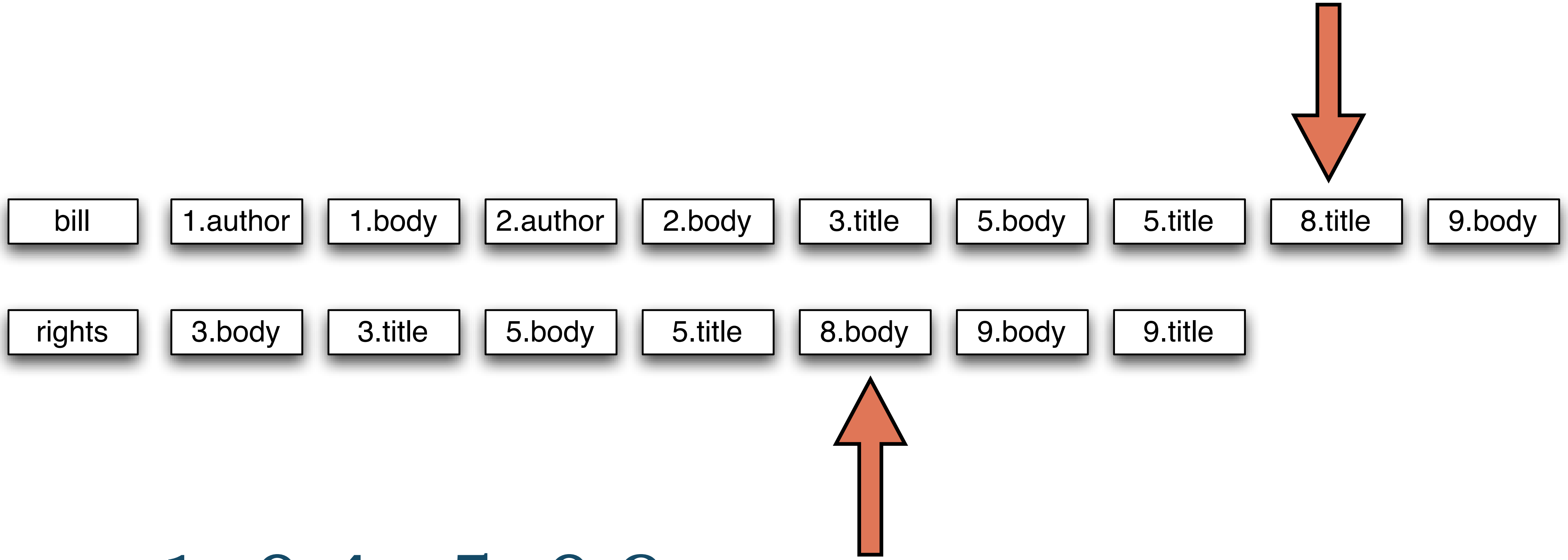
rights → 3.body → 3.title → 5.body → 5.title → 8.body → 9.body → 9.title

1: 0.4
2: 0.4
3: 0.9

5: 0.9

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

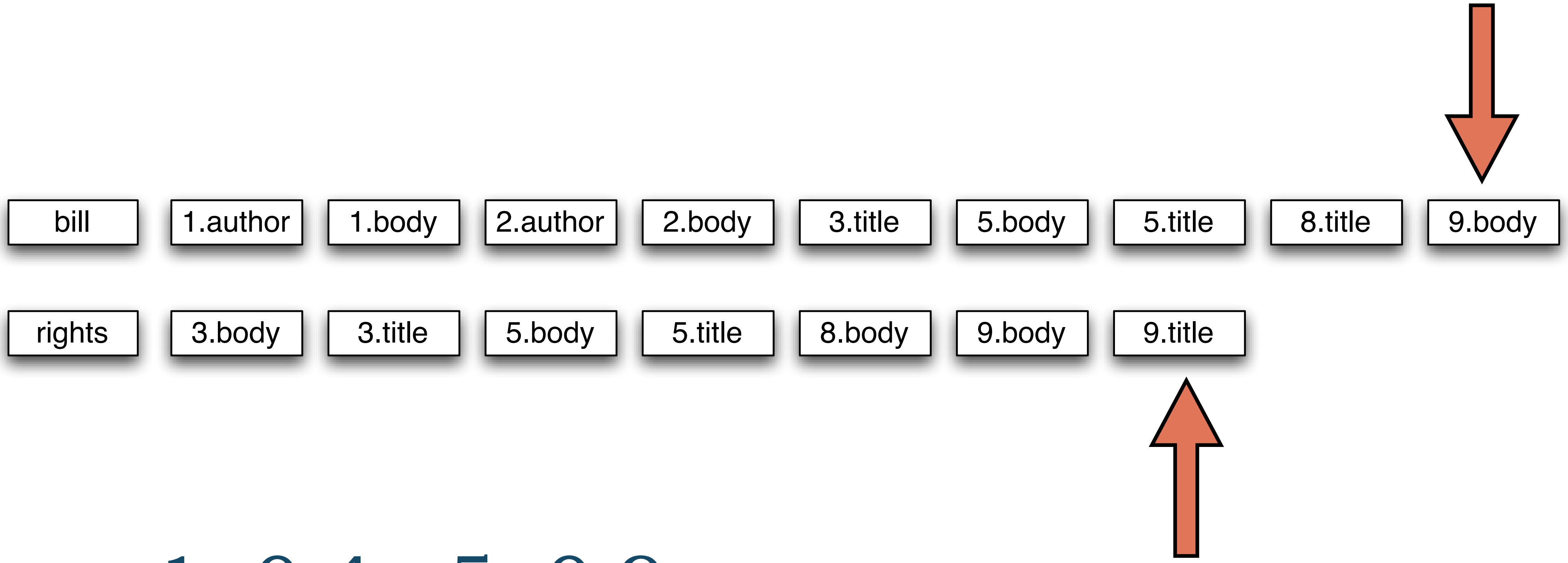
rights → 3.body, 3.title, 5.body, 5.title, 8.body, 9.body, 9.title

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

QUERYING

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

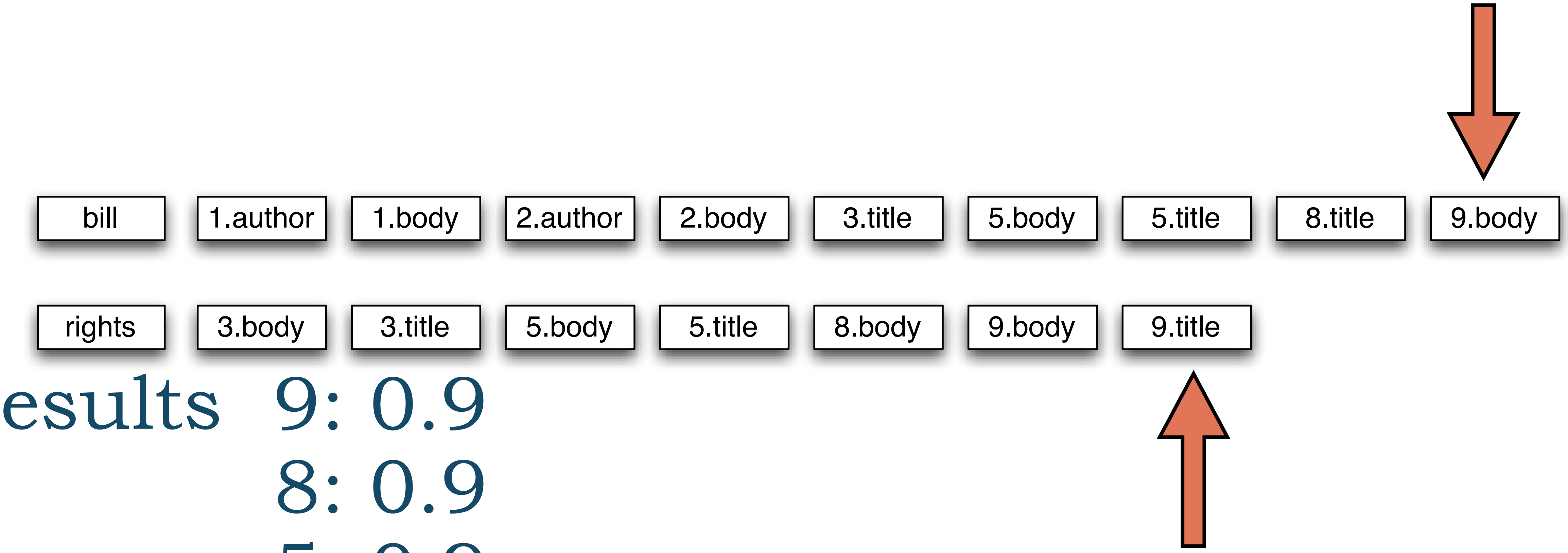
rights → 3.body → 3.title → 5.body → 5.title → 8.body → 9.body → 9.title

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

QUERYING

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

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

QUERYING

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”

QUERYING

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

QUERYING

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

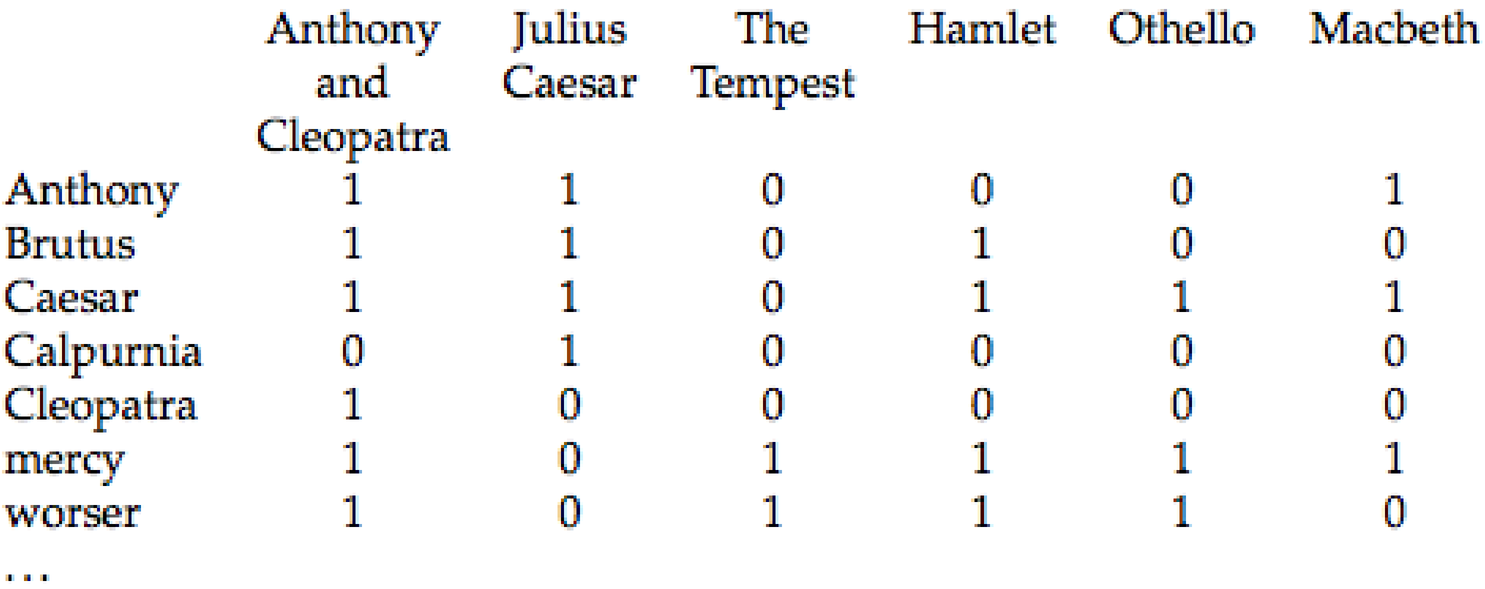
BUILDING UP OUR QUERY TECHNOLOGY

- “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

QUERYING

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?

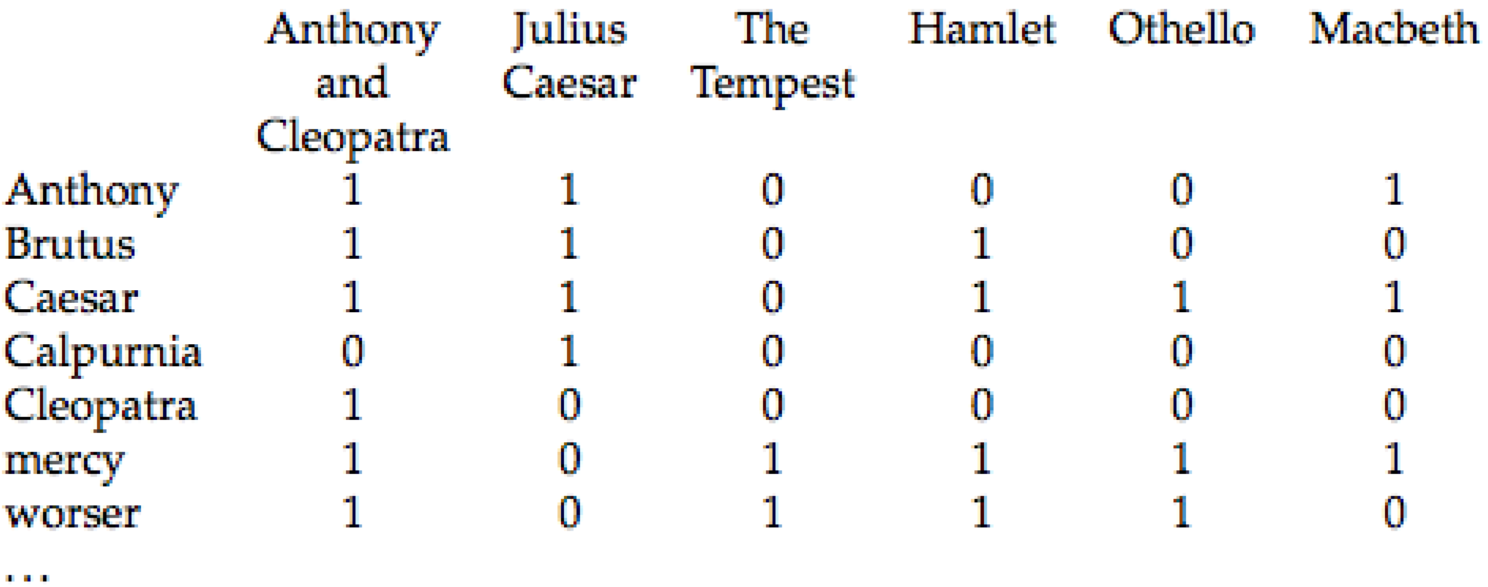


	Anthony	Julius	The	Hamlet	Othello	Macbeth
	and	Caesar	Tempest			
	Cleopatra					
Anthony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0
...						

INCIDENCE MATRICES

- Using this formalism, score can be an 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
...						

QUERYING

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

QUERYING

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

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

  - Cosine Measure
    - Will the same documents score well?

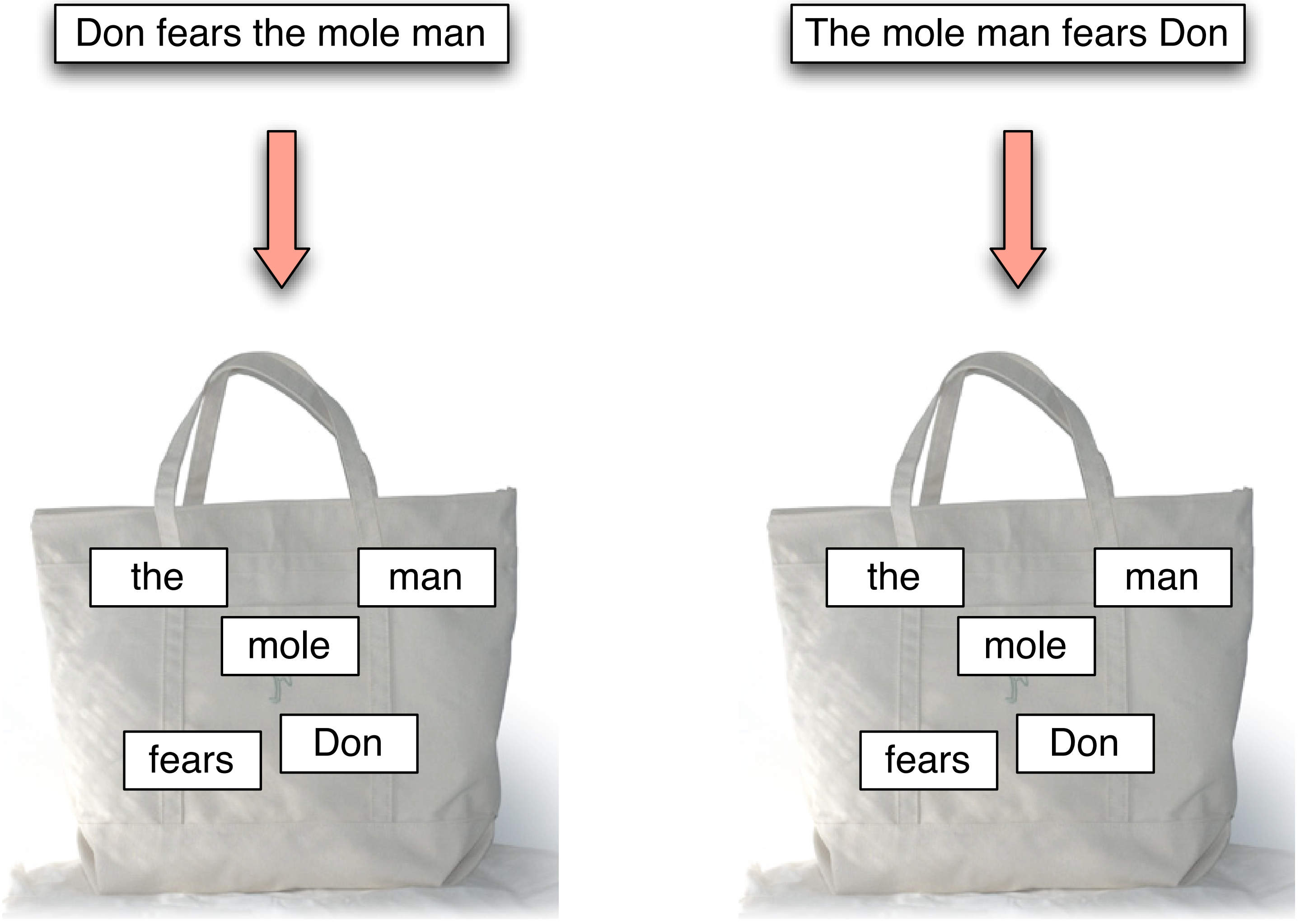
$$\frac{|q \cap d|}{\sqrt{|q||d|}}$$

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



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

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

QUERYING

TERM FREQUENCY

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

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 $return(1 + log(tf_{t,d}))$
```

QUERYING

WEIGHTING TERM FREQUENCY - WTF

- The score for query, q, is
    - Sum over terms, t

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

$$Score_{WTF}(q, d) = \sum_{t \in q} (WTF(t, d))$$

What is the score of “bill rights” in the declaration of independence?

http://www.archives.gov/exhibits/charters/declaration_transcript.html

QUERYING

WEIGHTING TERM FREQUENCY - WTF

- The score for query, q, is
    - Sum over terms, t

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

$$Score_{WTF}(q, d) = \sum_{t \in q} (WTF(t, d))$$

$$\begin{aligned}
Score_{WTF}(\text{"}bill\ rights\text{"}, declarationOfIndependence) &= \\
WTF(\text{"}bill\text{"}, declarationOfIndependence) &+ \\
WTF(\text{"}rights\text{"}, declarationOfIndependence) &= \\
0 + 1 + log(3) &= 1.48
\end{aligned}$$

http://www.archives.gov/exhibits/charters/declaration_transcript.html

QUERYING

WEIGHTING TERM FREQUENCY - WTF

$$Score_{WTF}(q,d) = \sum_{t \in q} (WTF(t,d))$$

$$\begin{aligned}
Score_{WTF}(''bill\ rights'', declarationOfIndependence) &= \\
WTF(''bill'', declarationOfIndependence) &+ \\
WTF(''rights'', declarationOfIndependence) &= \\
0 + 1 + log(3) &= 1.48
\end{aligned}$$

$$\begin{aligned}
Score_{WTF}(''bill\ rights'', constitution) &= \\
WTF(''bill'', constitution) &+ \\
WTF(''rights'', constitution) &= \\
1 + log(10) + 1 + log(1) &= 3
\end{aligned}$$

WESTMONT INSPIRED

COMPUTING LAB