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

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

- 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"

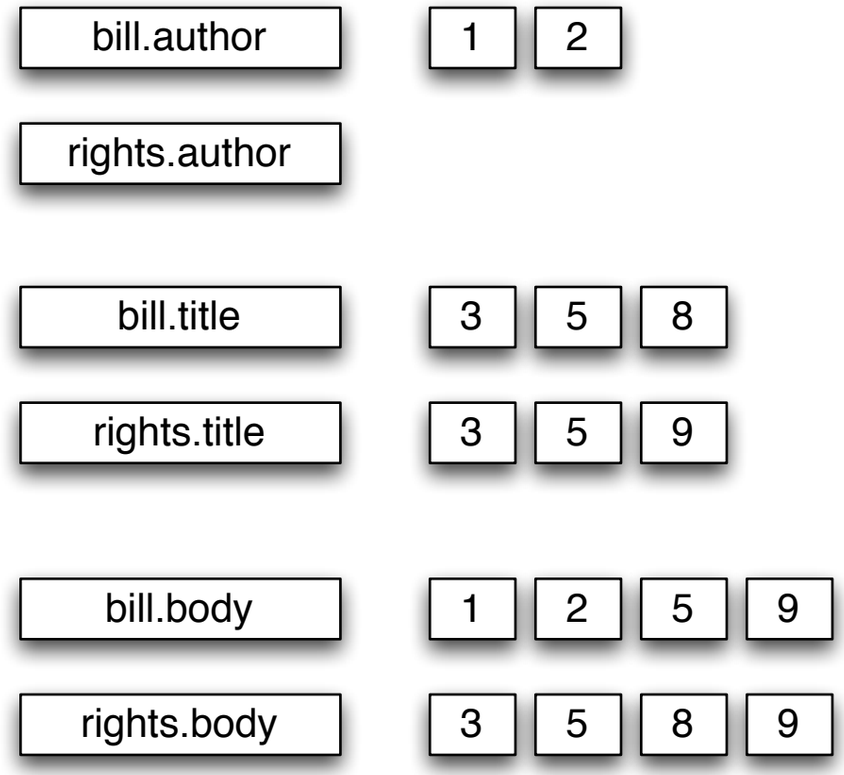
bill.author	1	2		
rights.author				
bill.title	3	5	8	
rights.title	3	5	9	
bill.body	1	2	5	9
rights.body	3	5	8	9

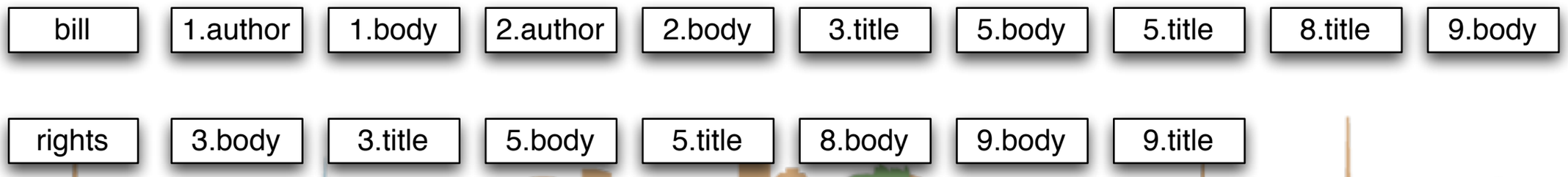
Building up our query technology

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

Zones combination index

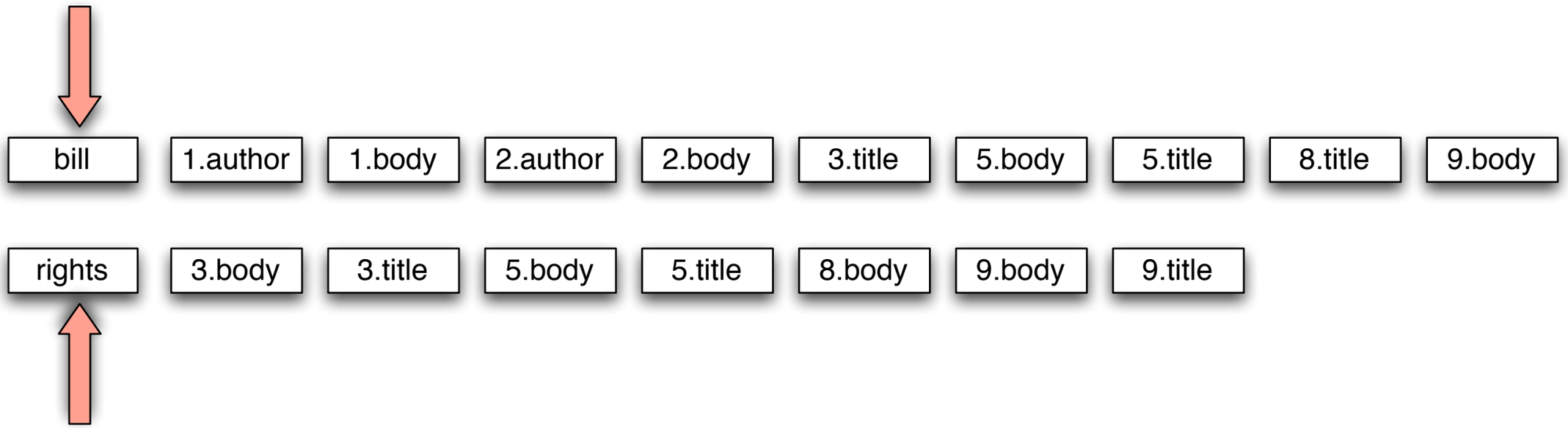
- Encode the zone in the posting
- At query time accumulate the contributions to the total score from the various postings





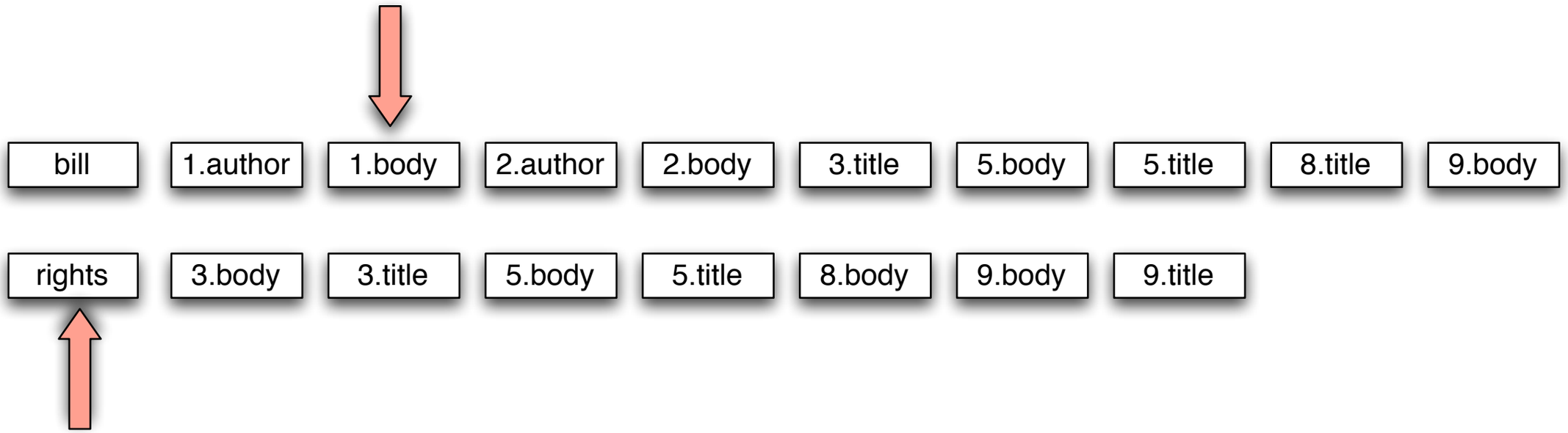
Zone scoring with zones combination index

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



Zone scoring with zones combination index

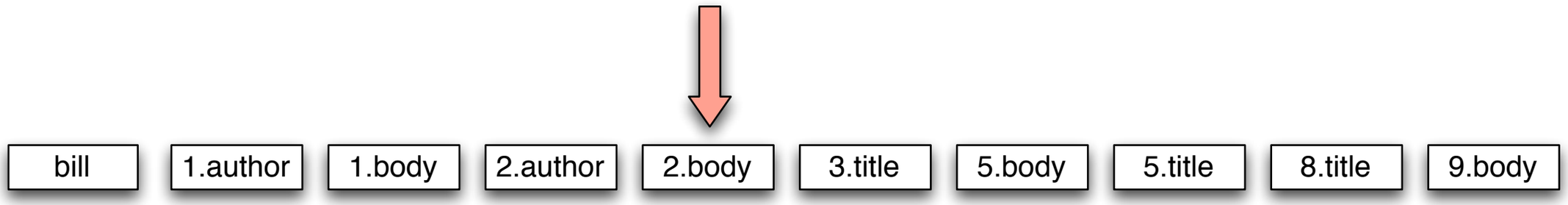
"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



1: 0.4

Zone scoring with zones combination index

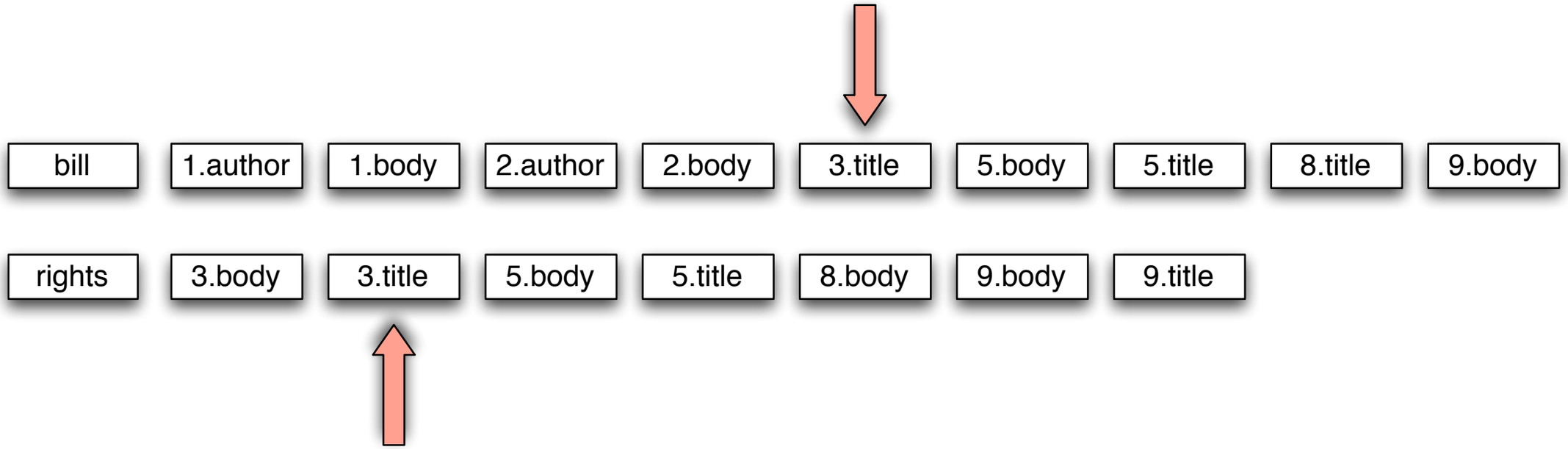
"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



1: 0.4
2: 0.4

Zone scoring with zones combination index

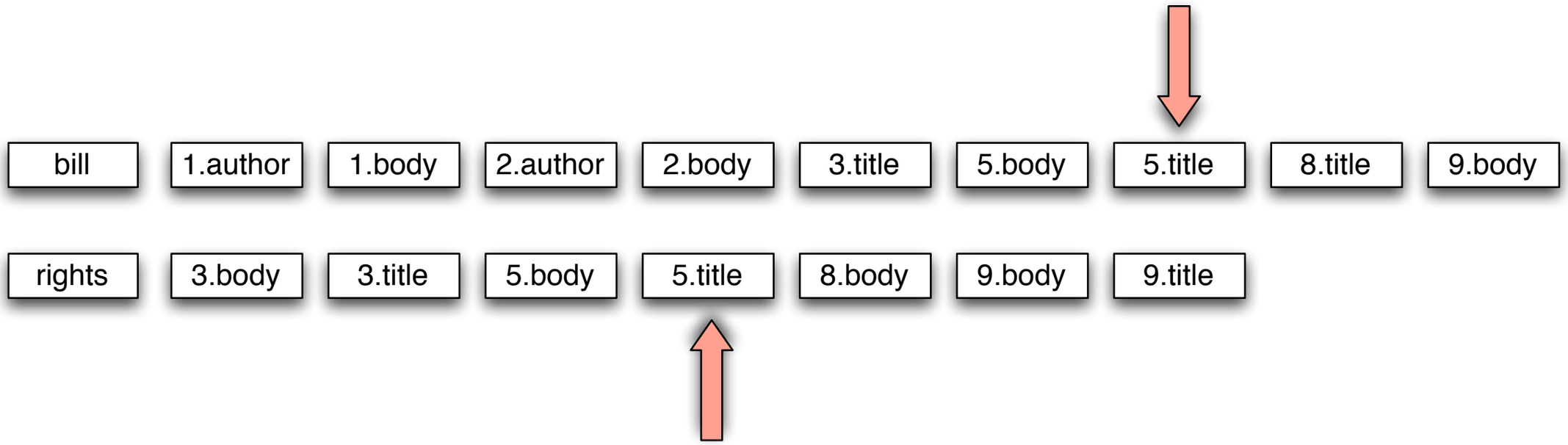
"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



1: 0.4
2: 0.4
3: 0.9

Zone scoring with zones combination index

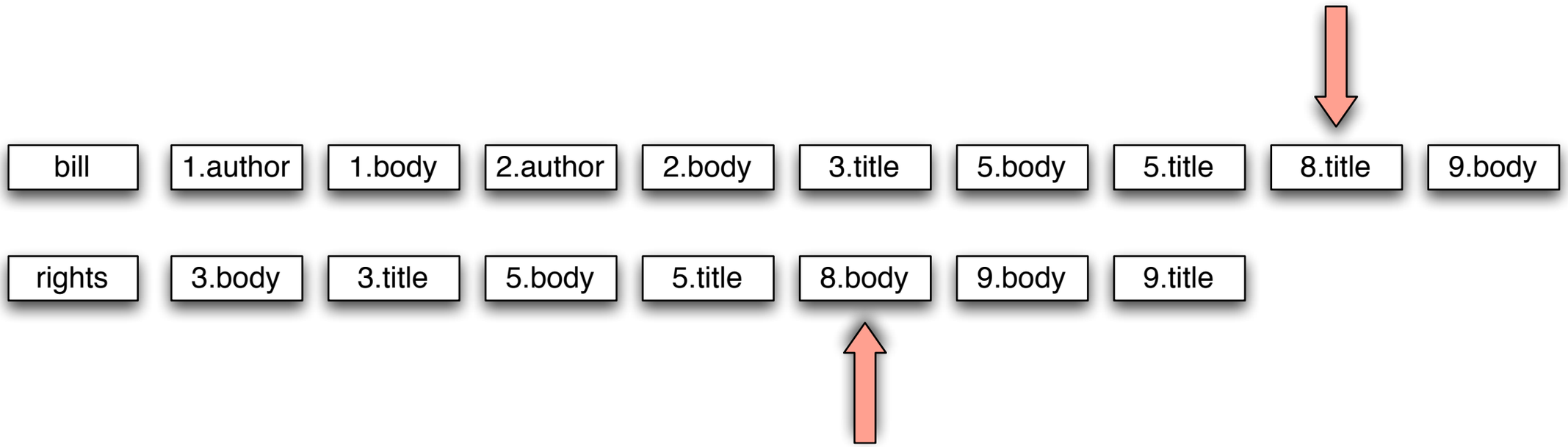
"bill OR rights" (0.1 author), (0.3 body), (0.6 title)



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

Zone scoring with zones combination index

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



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

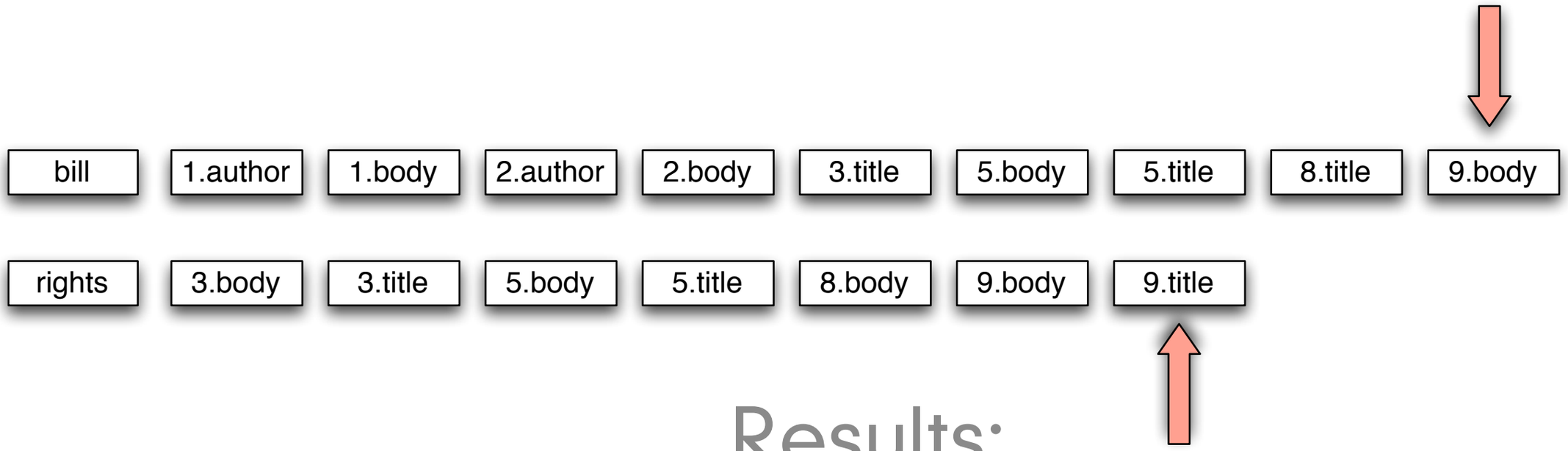
Zone scoring with zones combination index

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

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

Zone scoring with zones combination index

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

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

	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

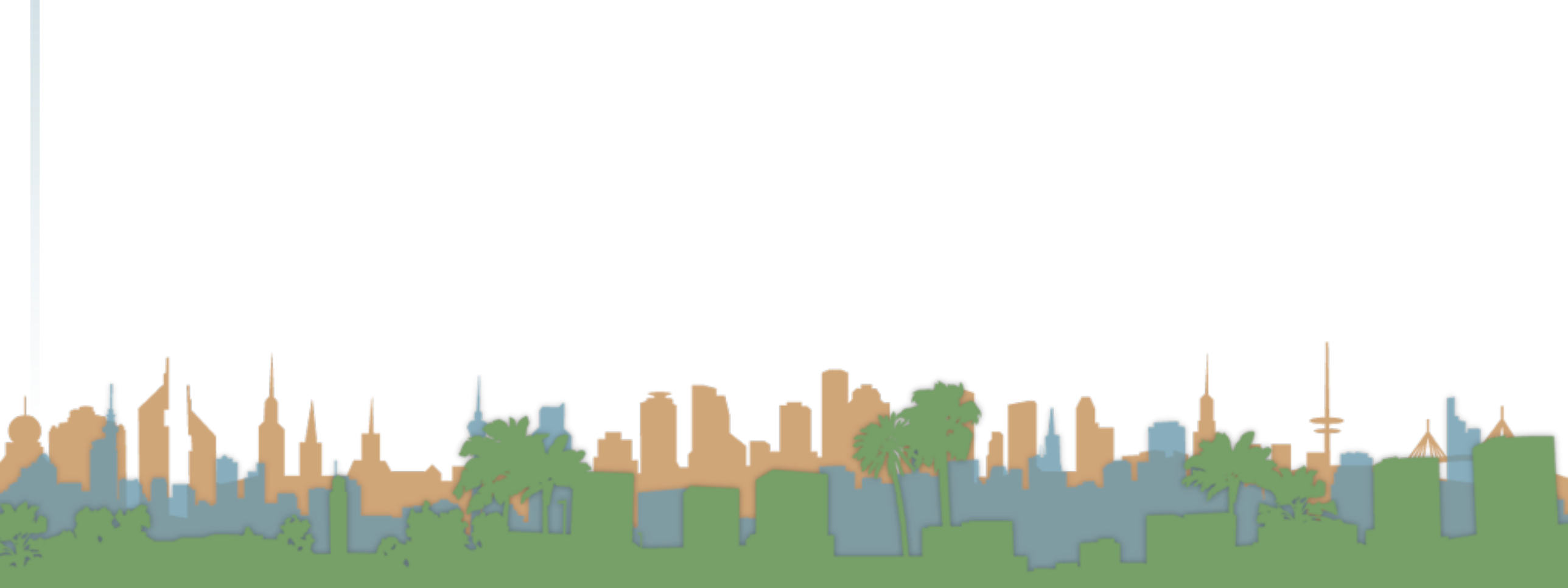
Term Frequency Matrix

- 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 occurences of a word in a document?
  - 2 vs. 100 occurences of a word in a document?
- Answer is unclear:
  - More is better, but not proportionally
  - An alternative to raw tf:

$$\begin{array}{ll} \text{WTF}(t,d) & \\ 1 & \textbf{if } tf_{t,d} = 0 \\ 2 & \quad \textbf{then } return(0) \\ 3 & \quad \textbf{else } return(1 + log(tf_{t,d})) \end{array}$$

Weighting Term Frequency - WTF

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

$$\mathrm{WTF}(t,d)$$
$$\begin{array}{ll} 1 & \textbf{if } tf_{t,d} = 0 \\ 2 & \quad \textbf{then } return(0) \\ 3 & \quad \textbf{else } return(1 + log(tf_{t,d})) \end{array}$$

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

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

Weighting Term Frequency - WTF

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

$$\begin{aligned}
Score_{WTF}(\text{"bill rights"}, declarationOfIndependence) &= \\
WTF(\text{"bill"}, declarationOfIndependence) &+ \\
WTF(\text{"rights"}, declarationOfIndependence) &= \\
0 + 1 + log(3) &= 1.48 \\
Score_{WTF}(\text{"bill rights"}, constitution) &= \\
WTF(\text{"bill"}, constitution) &+ \\
WTF(\text{"rights"}, constitution) &= \\
1 + log(10) + 1 + log(1) &= 3
\end{aligned}$$

Weighting Term Frequency - WTF

- Can be zone combined:

$$\begin{aligned} Score \quad = \quad & 0.6(Score_{WTF}(''instant\ oatmeal\ health", d.title) + \\ & 0.3(Score_{WTF}(''instant\ oatmeal\ health", d.body) + \\ & 0.1(Score_{WTF}(''instant\ oatmeal\ health", d.abstract) \end{aligned}$$

- Note that you get 0 if there are no query terms in the document.
  - Is that really what you want?
  - We will eventually address this

Unsatisfied with term weighting

- Which of these tells you more about a document?
  - 10 occurrences of “mole”
  - 10 occurrences of “man”
  - 10 occurrences of “the”
- It would be nice if common words had less impact
  - How do we decide what is common?
- Let’s use corpus-wide statistics

Corpus-wide statistics

- Collection Frequency, cf
  - Define: The total number of occurences of the term in the entire corpus
- Document Frequency, df
  - Define: The total number of documents which contain the term in the corpus

Corpus-wide statistics

$Word$	$Collection\ Frequency$	$Document\ Frequency$
$insurance$	10440	3997
try	10422	8760

- This suggests that df is better at discriminating between documents
- How do we use df?