11-442 / 11-642 / 11-742:
Search Engines

Document Structure

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

1

Introduction

Until now, the discussion of indexes and retrieval models treated a document as a single bag of words

Today we consider more complex documents

- Documents with fields ("flat structure")
- Hierarchical documents (e.g., XML)

We also consider two sources of document structure

- Explicit markup
- Multiple text representations

2

Sources of Document Structure #1: Explicit Markup

<DOC>

<PUBLICATION> Carcinogenesis </PUBLICATION>

<DATE> December 7, 2017 </DATE>

<TITLE> Aspartame, a bittersweet pill </TITLE>

<AU> M Paolini, F Vivarelli, A Sapone, D Canistro </AU>

<ABSTRACT> For the first time, the aspartame case shows how a corporation decided to ban an artificial … </ABSTRACT>

<MeSH> Animals, Aspartame/toxicity*, … </MeSH>

<SUBSTANCES> Sweetening Agents, Aspartame </SUBSTANCES>

: : :

</DOC>

3

3

Sources of Document Structure #1: Explicit Markup

Usually explicit markup is created by the author

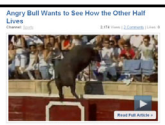
Explicit markup may be simple or complex

- "Flat" structure
  - E.g., PubMed search engine:
    - » Fields: title, author, journal, abstract, attribute/value metadata, …
  - Usually 5-50 fields per document
- Hierarchical structure
  - E.g., XML documents: Title, abstract, chapters, sections, subsections, tables, footnotes, citations, references, …
  - Often many types of elements in each document

4

4

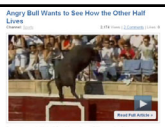
Sources of Document Structure #2: Multiple Representations



<DOC>
<TITLE> See New Viral Videos: Bull in Crowd **</TITLE>**
<BODY>
A Spanish sporting exhibition went horribly awry when a disgruntled bull leapt into the stands and began forcefully interacting with spectators. 40 onlookers were injured …
</BODY>
<URL> spike, channel, viralvideo, bull in crowd **</URL>**
<INLINK>
bull in crowd video, bull jumps into crowd, 40 people hurt, crazy video, Spanish bull fights back, bullfighting tragedy, ...
</INLINK>
</DOC>

5

Sources of Document Structure #2: Multiple Representations



Usually multiple representations are created by the search engine

- Perhaps using explicit markup provided by the author
  - E.g., Metadata keywords, title, body
- Perhaps using information from other documents
  - E.g., citation text, inlink text

Usually each representation is a simple bag-of-words

- A small'ish number of representations (e.g., 5-10)
- Not hierarchical

6

Outline

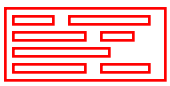
Document structure

- Index support for structure
- Fields
- Multiple representations of meaning
- Hierarchical structure ("XML documents")

7

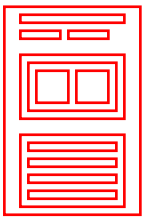
Indexing Structured Documents: Two Typical Approaches

1. Treat each region as independent of other regions



- This makes sense for flat fields
- E.g., don't mix TITLE, DATE, AUTHOR terms
- E.g., Medical records
- **Advantage:** Simple architecture

2. Treat regions as part of a hierarchy of related content



- This makes sense for XML documents & elements
- E.g., DOCUMENT ⊇ SECTION ⊇ SUBSECTION
- E.g., Scientific papers, government regulations
- **Advantage:** Flexible, may better match user needs

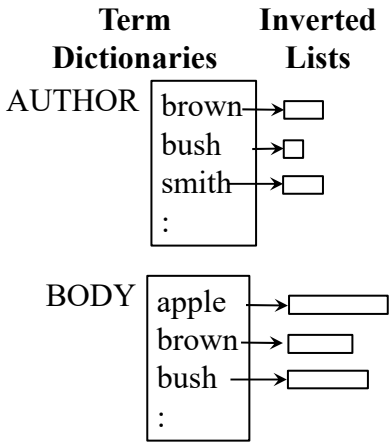
8

Storing Fields and Structure Implicitly: Separate Vocabularies

When regions are independent, a simple approach is to use separate vocabularies for each field

- Treat terms as a combination of FIELD and TERM information
  - E.g., AUTHOR::bush, BODY::bush
  - E.g., bush.AUTHOR, bush.BODY
  - E.g., "aspartame" [MeSH Terms]
- Lucene does this

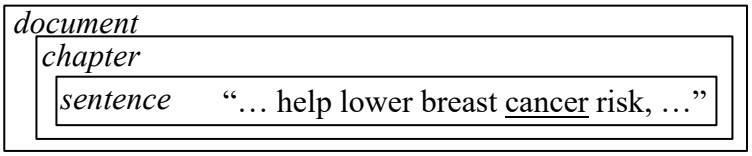
Simple, efficient, effective for shallow structure



Storing Fields and Structure Explicitly

Complex structure requires a more sophisticated approach

- Users want to specify which parts of the document are matched
  - #AND[document] (breast cancer) *Retrieve documents*
  - #AND[chapter] (breast cancer) *Retrieve chapters*
  - #AND[sentence] (breast cancer) *Retrieve sentences*
- If each element is indexed by a separate vocabulary, how many inverted lists index cancer?



  - One inverted list for every element that cancer can retrieve (i.e., 3)
  - High storage costs when structure is hierarchical and deep

Storing Fields and Structure Explicitly

Complex structure requires a more sophisticated approach

Solution

- Store term locations separately from document structure
  - Additional data structures that store document structure
- At query time, use document structure to select term locations

There are two main approaches

- Store each document's structure
  - A new type of forward index
- Store each type of structure
  - A new type of inverted list

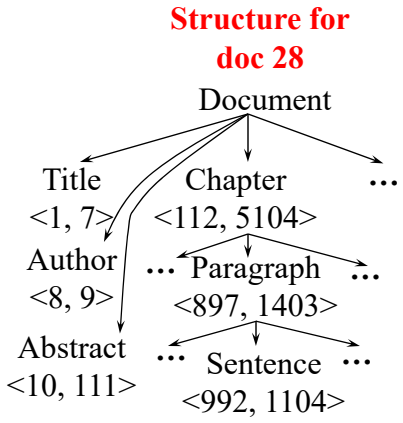
Storing Fields and Structure Explicitly: Trees

Document structure can be stored in a tree

- <begin, end> positions of each element
- Hierarchical relationship among elements
- Use trees to ignore parts of inverted lists
- This is a type of forward index
  - Given docid, get back all of its structure

Slower than using separate vocabularies

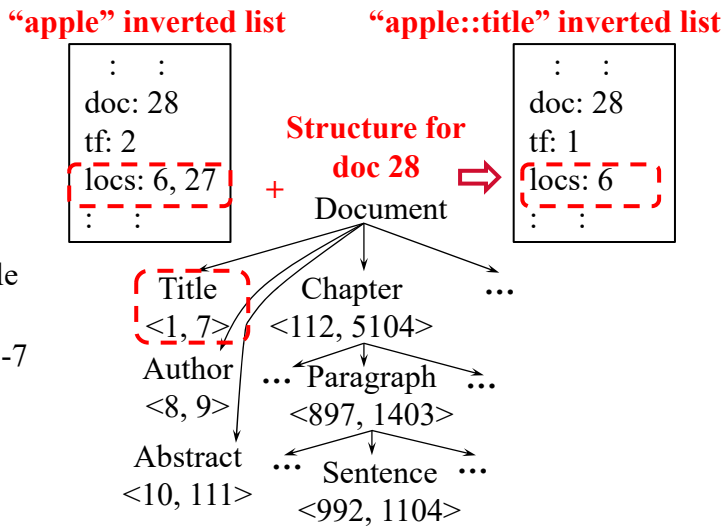
- But not a lot slower
- And a lot more flexible



Storing Fields and Structure Explicitly: Trees

Finding apple::title

- This is a QryIop operator
- Get apple inverted list
- For each document in the list
  - Get its structure tree
  - Traverse the tree to find the Title element
    - » Get Title range: locations 1-7
  - Discard inverted list locations outside the range 1-7
  - Result: new inverted list with tf=1, loc=6



13 © 2021, Jamie Callan

13

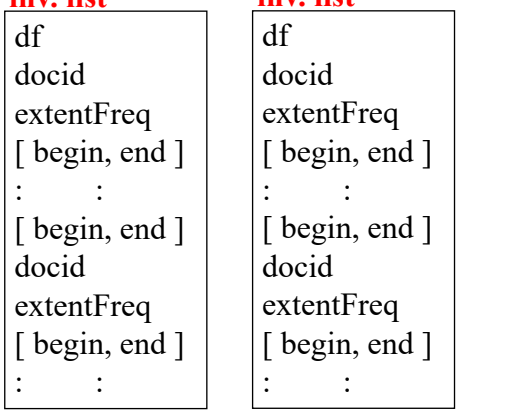
Storing Fields and Structure Explicitly: Inverted Lists

Element boundaries can be stored in inverted lists

- The search engine only accesses the structured needed for this query
- Efficient for elements that aren't in every document

Slower than using separate vocabularies

- But not a lot slower
- And a lot more flexible



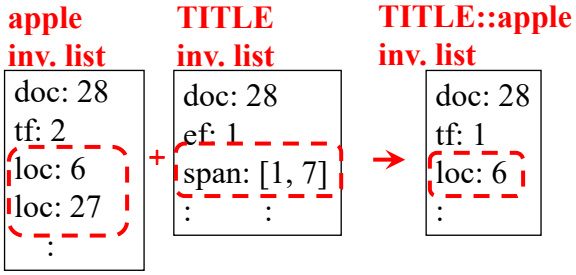
14 © 2021, Jamie Callan

14

Storing Fields and Structure Explicitly: Inverted Lists

Finding apple::title

- This is a QryIop operator
- Get apple inverted list
- Get TITLE inverted list
- Discard inverted list locations outside of title boundaries
- Result: new inverted list with tf=1, loc=6



Document Structure: Summary

Two sources of document structure

- Explicit structure: Usually created by the author
- Multiple representations: Usually created by the search engine

Three ways of storing document structure

- Store structure as part of term information (e.g., field::term)
- Store structure separately from term information
  - A tree of element locations for each document *forward index*
  - Inverted lists for each type of element *inverted index*

All 3 storage methods can handle both sources of structure

- But some are better choices than others

Outline

Document structure

- Index support for structure
- Fields
- Multiple representations of meaning
- Hierarchical structure ("XML documents")

17

Fields

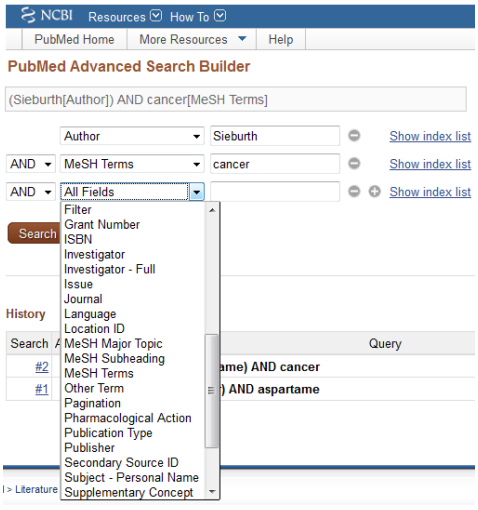
"Advanced search" interfaces allow people to construct complex queries

#FIELD (Author, Sieburth) AND #FIELD (MeSH, cancer) AND …

The user is assumed to be knowledgeable

- About query structure
- About document structure

Most effective for experts

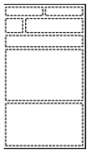


18

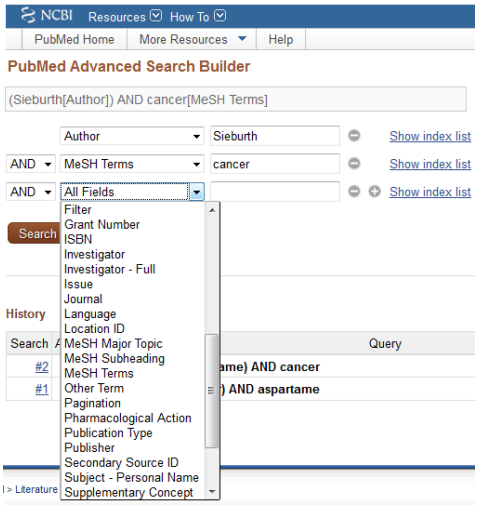
Fields:
Independent Representations

Usually "flat" document structure

- Independent fields
- No hierarchical structure
- Probably implemented as independent vocabularies
  - Author::Sieburth
  - This is what your homework systems do



The most common form of document structure



19 © 2021, Jamie Callan

19

Outline

Document structure

- Index support for structure
- Fields
- Multiple representations of meaning
- Hierarchical structure ("XML documents")

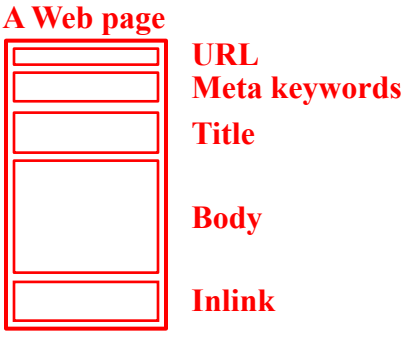
20 © 2021, Jamie Callan

20

Multiple (Related) Representations

Fields can be used to provide different representations of the same information

- Varied ways of representing the document content
- More opportunities for query terms to match the document
- Often implemented as independent vocabularies
  - TITLE::apple
  - This is what your homework systems do



How do retrieval models handle multiple representations?

Multiple (Related) Representations: The Vector Space

Use a separate vector space for each representation

- Weight each according to its reliability

$$\begin{aligned} Sim(q, d) = & \; w_{title} \times Sim(q, d_{title}) + \\ & w_{body} \times Sim(q, d_{body}) + \\ & w_{inlink} \times Sim(q, d_{inlink}) + \\ & w_{url} \times Sim(q, d_{url}) + \end{aligned}$$

Easy to manage, easy to extend

- Lucene does this

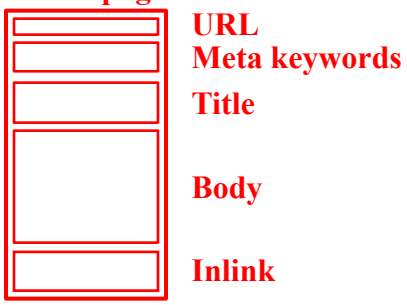
Note the similarity to your HW2 systems

- A #WSUM operator combining queries for title, body, …

Multiple (Related) Representations: BM25

Assume all fields represent the same document content

- Some fields are better evidence
  - E.g., inlink text is better evidence than body text
- Some fields are more verbose
  - E.g., body text vs. title text



Thus, evidence from each field should be weighted differently

23

Multiple (Related) Representations: BM25

How should BM25 be adapted to handle multiple fields?

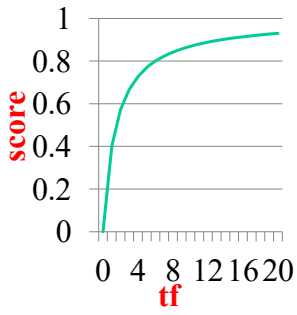
- Treat the document as |F| bags of words?
- Match the query to each field, add the scores?

This disrupts BM25's saturation assumption

- Appearing one time in |F| fields has more impact than appearing |F| times in one field

  5 fields × 1 occurrence = 5 × 0.4 = 2.0

  1 field × 5 occurrences = 0.77
- Is this the behavior that you want?



24

Multiple (Related) Representations: BM25F

Solution: Develop a composite, weighted representation

- A combined bag of words for the entire document

$$tf_t = \sum_{f \in F} w_f \, tf_{t,f}$$

$$doclen = \sum_{f \in F} w_f \, \text{doclen}_f$$

F: The set of fields

Intution: If title text is 5× more useful than body text, then replicate title text 5×

- Then just use standard BM25

Where is length normalization done?

- At the field level, or at the document level?
- Doing it at the field level provides greater control

Multiple (Related) Representations: BM25F

$$BM25F(q,d) = \sum_{t \in q} \left(\log \frac{N - df_t + 0.5}{df_t + 0.5} \right) \frac{tf_t}{k_1 + tf_t}$$

RSJ Weight

tf weight

$$tf_t = \sum_{f \in F} w_f \frac{tf_{t,f,d}}{(1 - b_f) + b_f \frac{length_{f,d}}{avglength_f}}$$

- Each field f has different parameters
  - w_f : Importance or value of field f
  - b_f : How field length normalization is done for field f

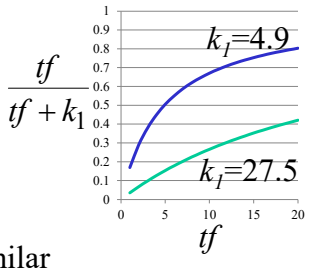
(Robertson & Zaragoza, 2007)

Multiple (Related) Representations: BM25F

Example weights (TREC 2003 Web Track)

Parameter	Topic Distillation	Named Page Finding
k_1	27.50	4.90
b_{title}	0.95	0.60
b_{body}	0.70	0.50
b_{anchor}	0.60	0.60
w_{title}	38.40	13.50
w_{body}	1.00	1.00
w_{anchor}	35.00	11.50
	Many relevant documents	1-2 relevant documents

similar ratios of weights



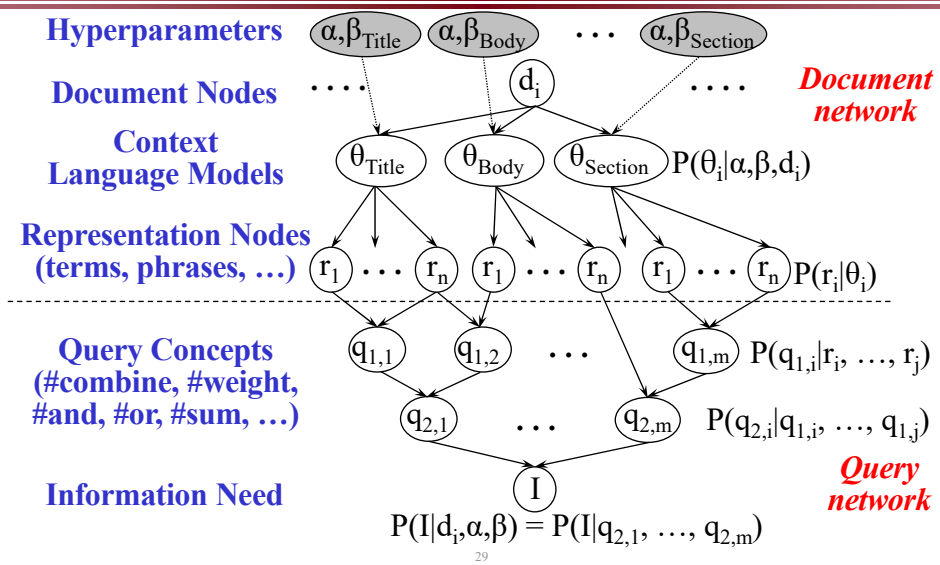
(Robertson & Zaragoza, 2007)

Multiple (Related) Representations: BM25F

Distinctive characteristics of BM25F

- One vocabulary covers all fields
  - A term has a global idf, not a field-specific idf
  - This makes sense when all fields represent the same document content
  - It might not make sense when fields have distinct content
    - » E.g., PubMed, …
- Field-specific tuning
  - But effect of constants isn't easy to understand

Multiple (Related) Representations: Indri



Multiple (Related) Representations: Indri

Bayesian inference networks offer two options for combining multiple representations

- #AND or #WAND $p_{wand}(q \mid d) = \prod_{q_i \in q} p(q_i \mid d)^{\frac{w_i}{w}}, \quad w = \sum w_i$
- #SUM or #WSUM $p_{wsum}(q \mid d) = \sum_{q_i \in q} \frac{w_i}{w} p(q_i \mid d), \quad w = \sum w_i$

The user query determines how information is combined

Which is best for combining evidence from multiple representations?

- Use #AND and #WAND to combine independent probabilities
- Use #SUM and #WSUM for different ways of estimating the same probability

Multiple (Related) Representations : Indri

User query The Time Traveler's Wife Search

A search engine might transform it into something like this

#and (
  #wsum(0.1 time.url 0.2 time.title 0.3 time.inlink 0.4 time.body)
  #wsum(0.1 traveler.url 0.2 traveler.title 0.3 traveler.inlink 0.4 traveler.body)
  #wsum(0.1 wife.url 0.2 wife.title 0.3 wife.inlink 0.4 wife.body))

31

Multiple (Related) Representations : Summary

Three retrieval models, three different approaches

- **Vector space**
  - Representations express the same meaning in different ways
  - Weighted average of scores to get a final ranking
- **Okapi BM25F**
  - Representations express the same meaning in different ways
  - Combine representations to get a better representation
- **Indri**
  - Representations may be related or independent
  - Each representation provides an estimate of p(t|d)
  - Multiple ways to combine the estimates

All three approaches work well

32

Outline

Document structure

- Index support for structure
- Fields
- Multiple representations of meaning
- Hierarchical structure (“XML documents”)

33