

Query Processing and Alternative Search Structures



CS 510 Winter 2007

1

Indexing common words



What is the indexing overhead for a common term?

I.e., does leaving out stopwords help?

Consider a word such as "and"

- It will appear in most documents
- Thus most gaps are 1 or 2
- Doc-ids in its inverted list take a small number of bits

CS 510 Winter 2007

2



Comparison to rare terms

Rare term has fewer document entries,
but the gaps are much larger on
average

So not necessarily a big difference in storage
for common and rare terms

However, inverted lists for common terms
require updating more often than those
for rare terms

CS 510 Winter 2007

3



Term-position index

Recall: Entries have the form

$\langle d ; k ; p_1, p_2, \dots, p_k \rangle$

- Have w -gaps between term positions,
can use variable-length code for them
- Entries for common words do take more
space
 - Index only the first m positions?
 - Store only in a document index?

CS 510 Winter 2007

4



+/- of compression

- Reduces space and transfer cost
- Can improve caching: Can keep more inverted lists in a given amount of main memory
- Incurs processing overhead
 - Decoding on use
 - Decoding & recoding on update
 - Generally compensated by lower disk costs

CS 510 Winter 2007

5



Query evaluation

Want to arrange computation of document similarities to scan inverted lists in order

Maintain an *accumulator* A_d for a document d

- For every document in the list for some term in the query
- Probably organize them on doc-id
- Initialize to 0

CS 510 Winter 2007

6



General algorithm

```

For each t in q
  look up  $w_{q,t}$ 
  For each  $\langle d, f_{d,t} \rangle$  in list(t)
     $A_d \leftarrow A_d + w_{q,t} * w_{d,t}$ 
For each non-zero  $A_d$ 
   $S_{q,d} \leftarrow A_d / W_d$ 

```

CS 510 Winter 2007

7



Variations on the theme

May want to limit memory use or processing time of a query

- Memory use often dominated by accumulators
- Processing time often dominated by the number of disk reads
 - Might be interested in time for first page of results

Note that such techniques will modify the answers you get

CS 510 Winter 2007

8



Limiting accumulators

Note that requiring all query terms to be in the document limits the number of accumulators

$$q = \{t_1, t_2, t_3\}$$

At the end, only need accumulators for

$$\text{docs}(t_1) \cap \text{docs}(t_2) \cap \text{docs}(t_3)$$

In this case, start with t_i with smallest $\text{docs}(t_i)$

CS 510 Winter 2007

9



Hard limit on accumulators

Can put absolute bound on the number of accumulators

- Ignore documents after you have reached the max number
- Start with terms with highest $w_{q,t}$ (i.e., least frequent)

Intuition: Documents with those terms will most likely have highest similarity measure

CS 510 Winter 2007

10

Threshold on accumulator values

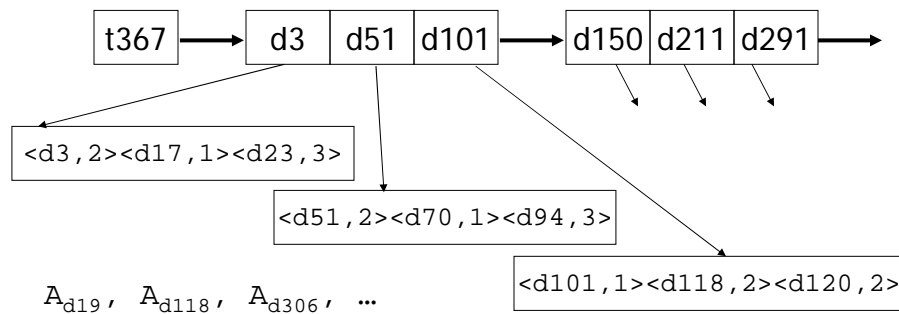
- Set a threshold S_{\min} on the value of A_d
 - Discard accumulators with less than this value
 - Do not create a new accumulator unless $w_{q,t} * w_{d,t} > S_{\min}$ for current term t
 - Start with S_{\min} low, increase as you go
 - Perhaps after each query term, set it to the 90th-percentile value

CS 510 Winter 2007

11

Avoid reading complete lists

If you are limiting accumulators, can skip reading or decoding parts of an inverted list that do not contribute to active accumulator



CS 510 Winter 2007

12



Frequency-ordered lists

- For the inverted list for a term t , put documents with highest frequency first
(In case we don't read the whole list)

`<d23,3><d94,3><d3,2><d51,2><d118,2><d120,2>
<d17,1><d70,1><d101,1>`

Don't need to repeat frequencies:

`3:<d23,d94> 2:<d3,d51,d118,d120> 1:<d17,d70,d101>`

Tradeoff: Don't repeat frequency values,
but the d-gaps are larger



Scanning multiple term lists

Rather than reading the inverted list for one query term completely, start by reading prefixes of the lists for all terms.

These are the quantities with the most to contribute to the similarity score



Impact-ordered lists

Note that the real contribution to $S_{q,t}$ from document d is based on $w_{d,t}/W_d$

So order lists on that quantity

Problem: That quantity isn't an integer

Solution: bin values in 10 to 30 ranges

1: (0.0, 0.5]

2: (0.5, 0.7]

3: (0.7, 0.85]

...

CS 510 Winter 2007

15



Importance-ordered lists

If your ranking function is dominated by something besides document-query similarity, use that quantity to order lists

For example, PageRank

CS 510 Winter 2007

16



Issue with ordered lists

Makes operations for Boolean queries difficult

Can't use merge-based algorithms: order is different in each list

But perhaps for importance-ordering, you could correlate doc-ids with importance

CS 510 Winter 2007

17



Fielded search

Some query interfaces can limit search terms to particular parts of document

- Consider journal papers, with search in
Title Authors Journal Abstract
- Can create "qualified" terms
title:Java abstract:Java
"Java" by itself is the paper body

CS 510 Winter 2007

18



Considerations for fielded search

- Suppose you don't care where the term is? How do you treat `title:Java`, `abstract:Java` and `Java` as the same term?
- Should `title:Java` use the W_d for the whole document?
- Give different weights to different fields?

CS 510 Winter 2007

19



Suffix tree

Treat each position in a document as representing the entire suffix of the document starting at that position

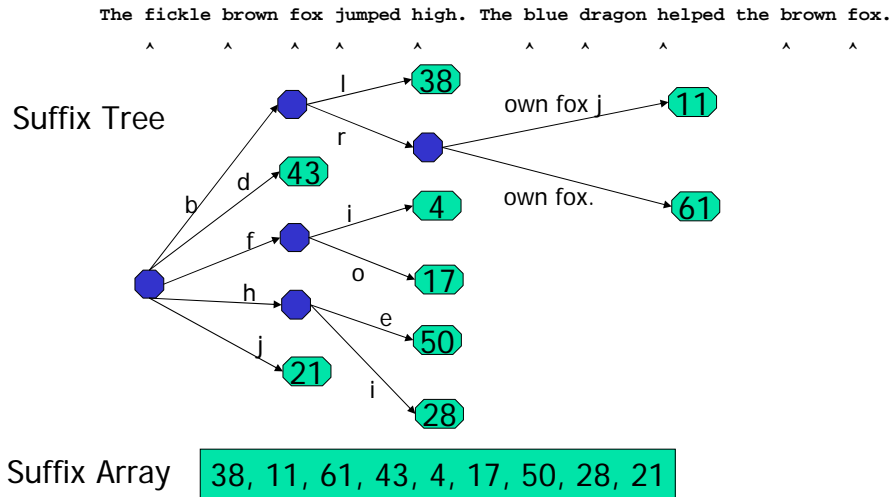
Build a search tree on those suffixes

- Can include every position, or just positions where a term starts
- Can combine multiple documents into one

CS 510 Winter 2007

20

Suffix tree; suffix array



CS 510 Winter 2007

21

Search

- Single word
 - Binary search through the suffix array
 - Add super-index to improve performance
- Multiple word/Proximity
 - Combine multiple single word searches
- Phrase
 - Binary search on the phrase!
- Pattern (`dest*`; `version?.?`)
 - Occurrences are localized in a subtree

CS 510 Winter 2007

22



However ...

Suffix trees are large structures – not much factoring of common info

Might get some of the value by representing vocabulary in a suffix array



Signature files

Will generate a *signature* for each document using a fixed number of bits B

Could be per segment of document for long documents

Each (indexed) term has a signature that sets V of the bits



Example term signatures

$B = 15, V = 3$

blue	00100	01000	01000
red	01000	10000	00001
brown	00010	00100	10000
cow	00001	01000	00100
fox	10000	00010	10000
cat	00010	00001	00010
big	10000	00010	01000
small	01000	00100	01000

CS 510 Winter 2007

25



Document signature

Compute the "or" of term signatures

d1: small blue cat

blue	00100	01000	01000
cat	00010	00001	00010
small	<u>01000</u>	<u>00100</u>	<u>01000</u>
sig1	01110	01101	01010

CS 510 Winter 2007

26

More document signatures

```

d2: big brown cow
    brown 00010 00100 10000
    cow   00001 01000 00100
    big   10000 00010 01000
    sig2  10011 01110 11100

d3: red fox
    red   01000 10000 00001
    fox   10000 00010 10000
    sig3  11000 10010 10001
    
```

CS 510 Winter 2007

27

Searching

To process a query

- Form its signature
- Find document signatures with at least those bits set

CS 510 Winter 2007

28



Search example 1

Signature file

```
sig1  01110 01101 01010
sig2  10011 01110 11100
sig3  11000 10010 10001
```

q1: small cat

```
cat    00010 00001 00010
small  01000 00100 01000
       01010 00101 01010
```



Search example 2

Signature file

```
sig1  01110 01101 01010
sig2  10011 01110 11100
sig3  11000 10010 10001
```

q2: brown cat

```
brown  00010 00100 10000
cat    00010 00001 00010
       00010 00101 10010
```



Search example 3

Signature file

```
sig1  01110 01101 01010
sig2  10011 01110 11100
sig3  11000 10010 10001
```

q3: brown fox

```
brown 00010 00100 10000
fox   10000 00010 10000
      10010 01010 10000
```

CS 510 Winter 2007

31



False drop

Query q3 is an example of a *false drop*
(also called a “false positive”)

Thus, need to post process matching documents

- Documents with more terms get more bits set, hence more likely to be false drops
- Might be good for use with a controlled vocabulary

Can we have false negatives?

CS 510 Winter 2007

32