

Web Searching & Indexing

CPS 116
Introduction to Database Systems

Announcements (November 29)

- ❖ Homework #4 due today
- ❖ Course project demo Dec. 7-14
 - Each project gets a 30-minute slot with me and Yi
 - Watch for an email this weekend scheduling demo slots
- ❖ Final exam on Saturday, Dec. 15, 7-10pm
 - Again, open book, open notes
 - Focus on the second half of the course

Keyword search

The screenshot shows a search engine interface with a search bar containing the text "database AND search" and a "Search" button. Below the search bar, there are several search results displayed as overlapping cards. One card is titled "The Internet Movie Database (IMDb)..." and another is titled "CPS 216: Advanced Database Systems (Fall 2007)".

What are the documents containing both "database" and "search"?

Keywords × documents

All keywords	All documents			
	Document 1	Document 2	Document 3	Document 4
"a"	1	1	1	1
"cat"	1	1	0	0
"database"	0	0	1	0
"dog"	0	1	0	1
"search"	0	0	1	0
...

1 means keyword appears in the document; 0 means otherwise

- ❖ Inverted lists: store the matrix by rows
- ❖ Signature files: store the matrix by columns

Inverted lists

- ❖ Store the matrix by rows
- ❖ For each keyword, store an inverted list
 - $\langle \text{keyword}, \text{doc-id-list} \rangle$
 - $\langle \text{"database"}, \{3, 7, 142, 857, \dots\} \rangle$
 - $\langle \text{"search"}, \{3, 9, 192, 512, \dots\} \rangle$
 - It helps to sort *doc-id-list* (why?)
- ❖ Vocabulary index on keywords
 - B⁺-tree or hash-based
- ❖ How large is an inverted list index?

Using inverted lists

- ❖ Documents containing "database"
 - Use the vocabulary index to find the inverted list for "database"
 - Return documents in the inverted list
- ❖ Documents containing "database" AND "search"
 - Return documents in the intersection of the two inverted lists
- ❖ OR? NOT?
 - Union and difference, respectively

What are "all" the keywords?

- ❖ All sequences of letters (up to a given length)?
 - ... that actually appear in documents!
- ❖ All words in English?
- ❖ Plus all phrases?
 - Alternative: approximate phrase search by proximity
- ❖ Minus all stop words
 - They appear in nearly every document, e.g., a, of, the, it
 - Not useful in search
- ❖ Combine words with common stems
 - Example: database, databases
 - They can be treated as the same for the purpose of search

Frequency and proximity

- ❖ Frequency
 - $\langle \text{keyword}, \{ \langle \text{doc-id}, \text{number-of-occurrences} \rangle, \langle \text{doc-id}, \text{number-of-occurrences} \rangle, \dots \} \rangle$
- ❖ Proximity (and frequency)
 - $\langle \text{keyword}, \{ \langle \text{doc-id}, \langle \text{position-of-occurrence}_1, \text{position-of-occurrence}_2, \dots \rangle \rangle, \langle \text{doc-id}, \langle \text{position-of-occurrence}_1, \dots \rangle \rangle, \dots \} \rangle$
 - When doing AND, check for positions that are near

Signature files

- ❖ Store the matrix by columns and compress them
- ❖ For each document, store a w -bit signature
- ❖ Each word is hashed into a w -bit value, with only $s < w$ bits turned on
- ❖ Signature is computed by taking the bit-wise OR of the hash values of all words on the document

Docs d_i contain

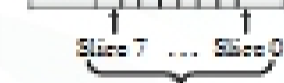
$\text{hash}(\text{"database"}) = 0110$	d_{x_1} contains "database": 0110 "database"?
$\text{hash}(\text{"dog"}) = 1100$	d_{x_2} contains "dog": 1100
$\text{hash}(\text{"cat"}) = 0010$	d_{x_3} contains "cat" and "dog": 1110

- ❖ Some false positives; no false negatives

Bit-sliced signature files

- ❖ Motivation
 - To check if a document contains a word, we only need to check the bits that are set in the word's hash value
 - So why bother retrieving all w bits of the signature?
- ❖ Instead of storing w signature files, store w bit slices
- ❖ Only check the slices that correspond to the set bits in the word's hash value
- ❖ Start from the sparse slices

doc	Signature
1	0 0 0 0 1 0 0 0
2	0 0 0 0 1 0 0 0
3	0 0 0 1 1 0 0 0
4	0 1 1 0 1 1 0 0
...	...
\mathcal{D}	0 0 0 0 1 0 0 0



 Bit-sliced signature files

 Starting to look like an inverted list again!

Inverted lists versus signatures

- ❖ Inverted lists better for most purposes (TODS, 1998)
- ❖ Problems of signature files
 - False positives
 - Hard to use because s , w , and the hash function need tuning to work well
 - Long documents will likely have mostly 1's in signatures
 - Common words will create mostly 1's for their slices
 - Difficult to extend with features such as frequency, proximity
- ❖ Saving grace of signature files
 - Sizes are tunable
 - Good for lots of search terms
 - Good for computing similarity of documents

Ranking result pages

- ❖ A single search may return many pages
 - A user will not look at all result pages
 - Complete result may be unnecessary
 - ❖ Result pages need to be ranked
- ❖ Possible ranking criteria
 - Based on content
 - Number of occurrences of the search terms
 - Similarity to the query text
 - Based on link structure
 - Backlink count
 - PageRank
 - And more...

Textual similarity

13

- ❖ Vocabulary: $\{w_1, \dots, w_n\}$
- ❖ IDF (Inverse Document Frequency): $\{f_1, \dots, f_n\}$
 - $f_i = 1 /$ the number of times w_i appears on the Web
- ❖ Significance of words on page p : $\{p_1 f_1, \dots, p_n f_n\}$
 - p_i is the number of times w_i appears on p
- ❖ Textual similarity between two pages p and q is defined to be $\{p_1 f_1, \dots, p_n f_n\} \cdot \{q_1 f_1, \dots, q_n f_n\} = p_1 q_1 f_1^2 + \dots + p_n q_n f_n^2$
 - q could be the query text

Why weight significance by IDF?

14

- ❖ Without IDF weighting, the similarity measure would be dominated by the stop words
- ❖ "the" occurs frequently on the Web, so its occurrence on a particular page should be considered less significant
- ❖ "engine" occurs infrequently on the Web, so its occurrence on a particular page should be considered more significant

Problems with content-based ranking

15

- ❖ Many pages containing search terms may be of poor quality or irrelevant
 - Example: a page with just a line "search engine"
- ❖ Many high-quality or relevant pages do not even contain the search terms
 - Example: Google homepage
- ❖ Page containing more occurrences of the search terms are ranked higher; spamming is easy
 - Example: a page with line "search engine" repeated many times

Backlink

16

- ❖ A page with more backlinks is ranked higher
- ❖ Intuition: Each backlink is a "vote" for the page's importance
- ❖ Based on local link structure; still easy to spam
 - Create lots of pages that point to a particular page

Google's PageRank

17

- ❖ Main idea: Pages pointed by high-ranking pages are ranked higher
 - Definition is recursive by design
 - Based on global link structure; hard to spam
- ❖ Naïve PageRank
 - $N(p)$: number of outgoing links from page p
 - $B(p)$: set of pages that point to p
 - $\text{PageRank}(p) = \sum_{q \in B(p)} (\text{PageRank}(q) / N(q))$
 - ☞ Each page p gets a boost of its importance from each page that points to p
 - ☞ Each page q evenly distributes its importance to all pages that q points to

Calculating naïve PageRank

18

- ❖ Initially, set all PageRank's to 1; then evaluate $\text{PageRank}(p) \leftarrow \sum_{q \in B(p)} (\text{PageRank}(q) / N(q))$ repeatedly until the values converge (i.e. a fixed point is reached)

