Web Searching & Indexing

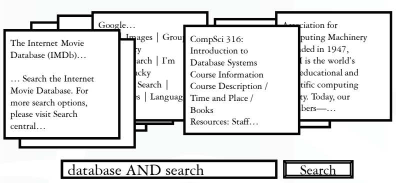
CompSci 316
Introduction to Database Systems

Announcements (Thu. Dec. 6)

- ❖ Homework #4 sample solution will be emailed by this weekend
- ❖ Course evaluation is online and <u>due by tomorrow</u>
- ❖ Project demo of Mobile Pay today from Michael, Kevin, Derek
- ❖ Final exam 2-5pm Dec. 12
  - Open book, open notes; focus on the second half
  - Extended office hours
    - Tue.: 12pm-2pm; Wed.: 10-11:30am, 1-2pm

Outline

- ❖ Keyword search



- ❖ Ranking search results
  - A single search may return many pages
  - A user will not look at all result pages
  - Complete result may be unnecessary

Keywords × documents

All documents

All keywords	Document 1	Document 2	Document 3	…	Document *n*
"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
  - ⟨*keyword*, *doc-id-list*⟩
  - ⟨"database", {3, 7, 142, 857, …}⟩
  - ⟨"search", {3, 9, 192, 512, …}⟩
  - It helps to sort *doc-id-list* (why?)
- ❖ Vocabulary index on keywords
  - B^+-tree, hash-based, or trie (later)

- ❖ 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?

7

- 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

8

- Frequency
  - ⟨*keyword*, { ⟨*doc-id*, *number-of-occurrences*⟩, ⟨*doc-id*, *number-of-occurrences*⟩, … }⟩
- Proximity (and frequency)
  - ⟨*keyword*, { ⟨*doc-id*, ⟨$position\text{-}of\text{-}occurrence_1$, $position\text{-}of\text{-}occurrence_2$, …⟩, ⟨*doc-id*, ⟨$position\text{-}of\text{-}occurrnece_1$, …⟩⟩, … }⟩
  - When doing AND, check for positions that are near

Signature files

9

- 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

hash("database") = 0110
hash("dog") = 1100
hash("cat") = 0010

doc_1 contains "database": 0110
doc_2 contains "dog": 1100
doc_3 contains "cat" and "dog": 1110

Does doc_3 contain "database"?

☞ Some false positives; no false negatives

Bit-sliced signature files

10

- 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 n 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	1	0
4	0	1	1	0	1	1	0	0
…	…							
N	0	0	0	0	1	0	1	0

Slice 7 … Slice 0

Bit-sliced signature files

Starting to look like an inverted list again!

Inverted lists versus signatures

11

- 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

12

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

- Terms $\{t_1, \dots, t_n\}$ and documents $D = \{d_1, d_2, \dots\}$
- IDF (Inverse Document Frequency) of t_i:
  - $\mathrm{idf}_i = -\log((\#\text{ of docs in } D \text{ containing } t_i)/|D|)$
- TF (Term Frequency) of t_i in d_j:
  - $\mathrm{tf}_{i,j} = \frac{\#\text{ of times } t_i \text{ appears in } d_j}{\#\text{ of term appearances in } d_j}$
- TF-IDF weight vector of d_j:
  - $\mathbf{w}_j = \langle \mathrm{tf}_{1,j}\mathrm{idf}_1, \dots, \mathrm{tf}_{n,j}\mathrm{idf}_n \rangle$
- Textual similarity between two docs d_j and d_k can be measured by the normalized dot product of these vectors, i.e.:
  $\frac{\mathbf{w}_j \cdot \mathbf{w}_k}{\|\mathbf{w}_j\|_2 \cdot \|\mathbf{w}_k\|_2} = \left(\sum_i \mathrm{tf}_{i,j}\mathrm{tf}_{i,k}\mathrm{idf}_i^2\right) \Big/ \left(\sqrt{\sum_i \mathrm{tf}_{i,j}^2\,\mathrm{idf}_i^2} \cdot \sqrt{\sum_i \mathrm{tf}_{i,k}^2\mathrm{idf}_i^2}\right)$
  - One “doc” could be the query text

Why weigh 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

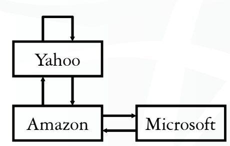
17

- 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
  - $F(p)$: set of pages that page p points to
  - $B(p)$: set of pages that point to p
  - $\mathrm{PageRank}(p) = \sum_{q \in B(p)} \mathrm{PageRank}(q)/|F(q)|$
  - ☞Each page gets a boost from every page pointing to it
  - ☞Each page distributes its importance evenly to pages it points to

Calculating naïve PageRank

18

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



$$\begin{bmatrix} y \\ m \\ a \end{bmatrix} = \begin{bmatrix} 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 \\ 0.5 & 1 & 0 \end{bmatrix} \begin{bmatrix} y \\ m \\ a \end{bmatrix}$$

$$\begin{bmatrix} y \\ m \\ a \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0.5 \\ 1.5 \end{bmatrix}, \begin{bmatrix} 1.25 \\ 0.75 \\ 1 \end{bmatrix}, \begin{bmatrix} 1.125 \\ 0.5 \\ 1.375 \end{bmatrix}, \begin{bmatrix} 1.25 \\ 0.6875 \\ 1.0625 \end{bmatrix}, \dots, \begin{bmatrix} 1.2 \\ 0.6 \\ 1.2 \end{bmatrix}$$

19

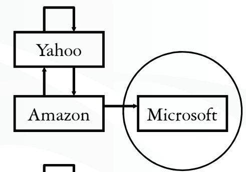
Random surfer model

- A random surfer
  - Starts with a random page
  - Randomly selects a link on the page to visit next
  - Never uses the “back” button
- $\text{PageRank}(p)$ measures the probability that a random surfer visits page p

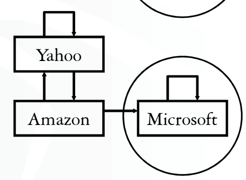
20

Problems with the naïve PageRank

- Dead end: a page with no outgoing links
  - A dead end causes all importance to “leak” eventually out of the Web



- Spider trap: a group of pages with no links out of the group
  - A spider trap will eventually accumulate all importance of the Web



21

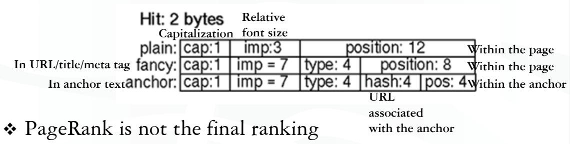
Practical PageRank

- d: decay factor
- $\text{PageRank}(p) = \left(d \cdot \sum_{q \in B(p)} \text{PageRank}(q)/|F(q)|\right) + (1 - d)$
- Intuition in the random surfer model
  - A surfer occasionally gets bored and jump to a random page on the Web instead of following a random link on the current page

22

Google (1998)

- Inverted lists in practice contain a lot of context information



- PageRank is not the final ranking
  - Type-weight: depends on the type of the occurrence
    - For example, large font weights more than small font
  - Count-weight: depends on the number of occurrences
    - Increases linearly first but then tapers off
  - For multiple search terms, nearby occurrences are matched together and a proximity measure is computed
    - Closer proximity weights more

23

Summary

- Index documents for substring queries
  - Inverted lists, signature files—index “words”
  - Other approaches (not covered): suffix tree, Pat tree, suffix array—index all suffixes
- Web search and information retrieval go beyond substring queries
  - TF-IDF, PageRank, …