Even More Indexing!

CPS 216
Advanced Database Systems

Keyword search

- Boolean searches
  - (database OR Web) AND search
- Phrase searches
  - “database search”
- Result ranking

What are the documents containing both “database” and “search”? 
Inverted lists

• Store the matrix by rows
• For each keyword, store an inverted list
  – <keyword, document-id-list>
  – <"database", [3, 7, 142, 857, …]>
  – <"search", [3, 9, 192, 512, …]>
  – It helps to sort document-id-list (why?)
• Vocabulary index on keywords
  – B+-tree or hash-based

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?
What are “all” the keywords?

- All sequences of letters?
  - … 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; not useful in search
  - Example: a, of, the, it
- Combine words with common stems
  - They can be treated as the same for the purpose of search
  - Example: database, databases

Frequency and proximity

- Frequency
  - `<keyword, {<doc-id, number-of-occurrences>,
    <doc-id, number-of-occurrences>,
    … }>`
- Proximity (and frequency)
  - `<keyword, {<doc-id, <position-of-occurrence>,
    position-of-occurrence>, …>,
    <doc-id, <position-of-occurrence>, …>,
    … }>`

Ranking Web pages using links

- Basic idea: A page is relevant if a lot of relevant pages have links pointing to it
  - Recursive definition?
    - No problem—fixed-point iteration!
- Google
  - Pre-compute the “general” ranking of all pages
  - This ranking can be use in the inverted lists
- HITS, Teoma
  - Compute the “topic-specific” ranking dynamically for pages that satisfy the search criteria
Keywords × documents

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Document 1</th>
<th>Document 2</th>
<th>Document 3</th>
<th>Document 4</th>
<th>Document 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;a&quot;</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>&quot;cat&quot;</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>&quot;database&quot;</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>&quot;dog&quot;</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>&quot;search&quot;</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Signatures

- Store the matrix by columns
- For each document, store a signature
  - If the document satisfies a search condition (e.g., contains “database”), set the corresponding bit in the signature
  - Signature too big? Compress!
    - Example: hash keywords and then set corresponding bits
      - Lossy compression can generate false positives

  \[
  \text{hash(“database”) = 0110}\quad \text{doc}, \text{contains “database”: 0110}
  \]

  \[
  \text{hash(“dog”)} = 1100\quad \text{doc}, \text{contains “dog”: 1100}
  \]

  \[
  \text{hash(“cat”) = 0010}\quad \text{doc}, \text{contains “cat” and “dog”: 1110}
  \]

Inverted lists versus signatures

- Inverted lists
  - High space overhead: could be bigger than the original documents!

- Signatures
  - Sequential scan through the signatures required
What’s next

GiST

Presenters:
Andrew Danner & Sara Sprenkle