Even More Indexing!

CPS 216
Advanced Database Systems

Search features

- Boolean searches
  - (database OR Web) AND search
- Phrase searches
  - “database search”
- Result ranking
  - Number of occurrences of keywords in the document
  - Proximity of keywords within the document
  - Popularity of document
  - Google, Teoma, etc., etc.

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
    - It helps to keep inverted lists sorted!
- OR? NOT?
  - Union and difference, respectively
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>, ...
    ... ]>
  - When doing AND, check for positions that are near

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>All documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>All keywords</td>
</tr>
<tr>
<td>“a”</td>
</tr>
<tr>
<td>“cat”</td>
</tr>
<tr>
<td>“database”</td>
</tr>
<tr>
<td>“dog”</td>
</tr>
<tr>
<td>“search”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Document 1</th>
<th>Document 2</th>
<th>Document 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Document 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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
        - Example: hash “database” = 0110
        - doc1 contains “database”: 0110
        - doc2 contains “dog”: 1100
        - hash(“cat”) = 0010
        - doc3 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