Berkeley-DB for Text/Multimedia Retrieval

Chun Jin
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Motivation

- Recent advance in text/multimedia retrieval: good algorithms
- Scalability issue
  - Continuous data growth
  - Adding new search features
- Try: separating the scalability problem from the retrieval algorithms?
Our Goal

- Providing a library for application/system building based on Berkeley DB.
- System prototyping.
Text Retrieval

- Vector Space Model (VSM) for Text:
  - \( D = \{ t_1, t_2, \ldots, t_m \} \)
  - \( Q = \{ t_1, t_2, \ldots, t_m \} \)
  - Sim(D, Q) = \cos(D, Q)

- To Scale: Inverted Index:

  \[
  \begin{align*}
  t_1 \rightarrow & \quad \text{DID TF POS1 POS2} \quad \ldots \\
  \cdot & \quad \ldots \\
  \cdot & \quad \ldots \\
  t_2 \rightarrow & \quad \text{DID TF POS1 POS2} \quad \ldots \\
  \cdot & \quad \ldots
  \end{align*}
  \]
Image Retrieval

- Feature Space
  - $M = \{f_1, f_2, \ldots, f_m\}$
  - $Q = \{f_1, f_2, \ldots, f_m\}$
  - Distance($M$, $Q$) = $||M - Q||$

- To Scale: Quantization then Index:
  - $f_1 \rightarrow \ldots$
  - $\ldots$
  - $f_i \rightarrow \text{MID } f_{im} \ldots$
  - $|f_iQ - f_{im}| < \delta$
Retrieval Algorithm

- Get feature entries
- Compute feature-level similarities
- Compute document-query similarities

\[ Q = \{TF_{t1}, TF_{t2}, ..., TF_{tm}\} \]
Retrieval Algorithm

- Get feature entries: Berkeley DB:
  - BTree/Hash indexing
  - Storage/buffer management
- Compute feature-level similarities
- Compute document-query similarities: Join:
  - AND/OR (Inner/Outer)
  - Join methods
  - Callback to compute Step 2
Development Layers

Retrieval Application:
  Feature Extraction
  Similarity Measures

Retrieval API:
  Join Methods
  Iterator
  Inverted Index Formatter

Berkeley DB API:
  Key Indexer Lib (BTree/Hash, etc.)
  Storage Management
<table>
<thead>
<tr>
<th>Task</th>
<th>BDB</th>
<th>DBMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexing techniques</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Storage management</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Operation: Join</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Developer’s API</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SQL</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Transaction management</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Recovery management</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Merge Join

```
List MergeJoin(List left, List right, Feature Qrfeature)
    while (not left.end() and not right.end())
        lpair = left.current;
        rpair = right.current;
        if lpair.key = rpair.key
            FeatureSim v = Qrfeature.Sim(rpair.data);
            lpair.data = DocSim(lpair.data, v);
            left = left.next();
            right = right.next();
        else if lpair.key < rpair.key
            left = left.next();
        else
            right = right.next()
    return left;
```
Information Encapsulation

- BDB: index key and entry boundary
- Iterator: sub-entry boundary
- Join: docID key and the rest of data
- Similarity function: data structure
Flexibility

- Feature similarity:
  - Term positions for proximity search
  - Weighted link information
  - Meta data adjustment

- Document-Query similarity:
  - Cosine
  - Euclidean
  - Probabilistic
Ongoing Work

- Inverted Index Structure Design
- Implementing join methods and iterators
- Inverted Indexer
- Similarity functions and feature extraction
Related Work

- **Commercial Systems:**
  - Google
  - Endeca
  - Oracle Text DB/Multimedia DB
  - IBM Net Search Extender
  - Thunderstone Texis
  - YouTube

- **Research**
  - CMU
  - Stanford [Su & Widom IDEAS05]
Conclusion

- **Problem:**
  - Scalability issue on text/multimedia retrieval.

- **Idea:**
  - Separating the problem from retrieval algorithms.
  - Layered architecture.

- **Goal:**
  - Providing a library for application/system building.
  - Prototyping.
Acknowledgements

- Thanks to Minglong Shao (CMU) and Zhu Liu (AT&T) for helpful discussions.
- Thanks to Jaime Carbonell (CMU) for his continuous support and encouragement.