Efficiency – Index Pruning & Query Processing

(COSC 488)
Nazli Goharian
nazli@cs.georgetown.edu

Efficiency Techniques

• Indexing
• Compression
  ➢ Index Pruning (Top Doc)
  ➢ Efficient Query Processing
  ➢ Duplicate Document Detection
Index Pruning
(Top Doc)

- Instead of retrieving the whole posting list, retrieve the top $x$ documents
- Documents are ordered by a weight ($tf$) in a PL
- Term specific pruning vs. uniform pruning
- A separate structure with sorted, truncated posting lists may be produced.

Experimentation results: 70% of index achieves similar average precision as full index

Inverted & Pruned Indices

**Inverted Index**

- $t1$:
  - D1 | 5
  - D2 | 10
  - D500 | 35
- $t2$:
  - D1 | 5
  - D35 | 8

**Pruned Index (D = 2)**

- $t1$:
  - D500 | 35
  - D2 | 10
  - Truncated
- $t2$:
  - D35 | 8
  - D1 | 5
Pruned Index Summary

• Pro
  – Avoids need to retrieve the entire posting list
  – Dramatic savings on efficiency for large posting lists

• Con
  – Not feasible for Conjunctive queries

Efficient Query Processing via Partial Processing

• Improving query run-time by partial result set retrieval

• May or may not need modification to inverted index
  – Process least frequent terms first
  – Process least frequent terms only (Query Thresholds)
Processing Least Frequent Terms First

• Process terms in “goodness” order
  – e.g., \( idf \cdot qtf \cdot t_f \cdot max \cdot idf \), …
  – using top 25%-75% no degradation in some Trec benchmark datasets

• Terminate processing after \( d \) documents are assigned non-zero scores, OR

• Continue processing for the above \( d \) non-zero scores with remaining query terms. Options:
  – Treat remaining terms as a conjunctive (AND) condition
    • Organize the index such that to support conjunctive processing
  – Traditional vector space disjunctive (OR)

Modifying Inverted Index to Support Fast Scanning

An approach:
• Assumption: Posting list is ordered based on doc id.
  – Partition the posting and add pointers to each partition from the previous partition.
  – Find the partition of document \( x \) from list \( d \) (see last slide), by checking the first doc id of two consecutive partitions.
  – If doc \( x \) is not found, jump to next partition. Otherwise, scan current partition.
Query Threshold

- Consider a query with terms $t_1$, $t_2$, $t_3$, ..., $t_n$.
- Define a threshold as the percentage of terms taken from the original query in a newly created reduced query.

<table>
<thead>
<tr>
<th>term1</th>
<th>term2</th>
</tr>
</thead>
<tbody>
<tr>
<td>term3</td>
<td>term4</td>
</tr>
<tr>
<td>term5</td>
<td>term6</td>
</tr>
<tr>
<td>term7</td>
<td>term8</td>
</tr>
<tr>
<td>term9</td>
<td>term10</td>
</tr>
</tbody>
</table>

threshold = 20%
threshold = 50%
threshold = 80%

Relevant Retrieved for Varying Query Thresholds

<table>
<thead>
<tr>
<th>Query Threshold (Percent)</th>
<th>Relevant Retrieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>831</td>
</tr>
<tr>
<td>20</td>
<td>1505</td>
</tr>
<tr>
<td>30</td>
<td>1657</td>
</tr>
<tr>
<td>40</td>
<td>1675</td>
</tr>
<tr>
<td>50</td>
<td>1856</td>
</tr>
<tr>
<td>60</td>
<td>2119</td>
</tr>
<tr>
<td>70</td>
<td>2138</td>
</tr>
<tr>
<td>80</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Run Time as a Function of Query Thresholds

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Relevant Retrieved</th>
<th>CPU Cycles</th>
<th>New Relevant Docs per Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>831</td>
<td>475</td>
<td>1.75</td>
</tr>
<tr>
<td>20</td>
<td>1505</td>
<td>950</td>
<td>1.58</td>
</tr>
<tr>
<td>25</td>
<td>1601</td>
<td>1238</td>
<td>1.29</td>
</tr>
<tr>
<td>33</td>
<td>1657</td>
<td>1736</td>
<td>0.95</td>
</tr>
<tr>
<td>50</td>
<td>1856</td>
<td>5215</td>
<td>0.36</td>
</tr>
<tr>
<td>75</td>
<td>2119</td>
<td>13181</td>
<td>0.16</td>
</tr>
<tr>
<td>100</td>
<td>2138</td>
<td>33989</td>
<td>0.06</td>
</tr>
</tbody>
</table>
Query Threshold Summary

• **Pro**
  – Avoids large posting lists.
    • Dramatic savings on efficiency when large posting list is not retrieved.
  – Effectiveness does not degrade (as long as we do not threshold too much) because we are omitting only those terms with long posting lists.

• **Con**
  – Still can have some very long posting lists.
  – May miss some good documents, affecting Recall.