Document Size Distribution

Andrew Kane, University of Waterloo
arkane@cs.uwaterloo.ca

Database Group Talk - Feb. 12th 2014

(LSDS-IR 2014 workshop paper with Frank Wm. Tompa)
Outline

• Introduction to search engines.
• Distribution by document size.
• List intersection.
• Experiments:
  • Space improvement.
  • Runtime improvements.
• Applications in practice.
Indexing

- Input: (document, metadata)⁺
- Name: (docID, documents, metadata)⁺
- Convert: (docID, text, metadata)⁺
- Tokenize: (docID, tokens and offsets)⁺
- Invert: map of token to docID⁺
  - i.e. list of documents containing that token (postings list).
  - Add frequency for ranking.
  - Add offsets for phrase, proximity and ranking.
# Search Engine Query Processing

<table>
<thead>
<tr>
<th>Query</th>
<th>Lookup</th>
<th>Intersect</th>
<th>Rank top-k</th>
<th>Expand</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>terms</td>
<td>encodings</td>
<td>list</td>
<td>short list</td>
<td>metadata</td>
<td></td>
</tr>
</tbody>
</table>

```
AND
OR
Weak-AND
Phrase
Proximity
```

---

Early Termination
Pruning
Search Engine Query Processing

Partition

Query | Lookup | Intersect | Rank top-k | Expand | Result
terms | encodings | list | short list | metadata

AND
OR
Weak-AND
Phrase
Proximity

Early Termination
Pruning
Document Distribution

- How do you distribute documents to partitions?
Document Distribution

- **Random distribution** is normally used:
  - Balanced distribution of query work and index size.
  - We refer to this as rand-p

- **Document size distribution** improves performance:
  - Benefits to index size and query resource usage.
  - Balancing requires tuning of the partition cutoff points.
  - We measure size by # terms in document.
  - We refer to this as td-p
Within Partitions

- Can use any document ordering within the partitions.
  - We use random ordering for our tests to avoid bias, so we compare rand-p-rand vs. td-p-rand.
  - Using URL ordering produces similar types of improvement.
## Search Engine Query Processing

<table>
<thead>
<tr>
<th>Query</th>
<th>Lookup</th>
<th>Intersect</th>
<th>Rank top-k</th>
<th>Expand</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>terms</td>
<td>encodings</td>
<td>list</td>
<td>short list</td>
<td>metadata</td>
<td></td>
</tr>
</tbody>
</table>

- **AND**
- **OR**
- **Weak-AND**
- **Phrase**
- **Proximity**

- **Early Termination**
- **Pruning**
List Intersection

Uncompressed lists: next(), fsearch(int, method).
Compressed lists: next().
• We use simple16 compression: layout + data = 4 + 28 bits
• Skips over compressed lists: `next()`, `fsearch(int)`.
List Intersection

- Bitvector: contains(int), bitwiseAND(bitvector), convertToList().
- Hybrid: Use bitvectors if freq. > F and compressed lists for others [Culpepper and Moffat 2010].
Experiments

- Conjunctive-AND list intersection.
- Three partitions with equal number of postings.
- Sum index space and query runtime over partitions.
- Setup:
  - Using GOV2 dataset (426GB) and 5000 corpus queries (4.1 terms per query).
  - AMD Phenom II X6 1090T 3.6Ghz Processor running Ubuntu Linux 2.6.32-43-server.
• Terms-in-document count for GOV2 dataset, split by number of postings into three partitions.
After Partitioning

- Encoding as compressed lists of deltas (simple16):
  - rand-p-rand: 7.54 bits/posting.
  - td-p-rand: 6.70 bits/posting.

- Space improvement of 11.1%.
Benefits with Skips

• Skew from terms-in-document distribution:
  • In large-document partitions:
    • Increases density of postings.
    • Therefore, cache line clustering (locality of access).
  • In small-document partitions:
    • Reduces density of intermediate results.
    • Therefore, skips more effective.
Results for Skips

![Graph showing time (ms/query) vs. space (bits/posting) for S16+skips(rand–p–rand) and S16+skips(td–p–rand)]
Benefits with Bitvectors

- Bitvectors chosen independently in each partition (red).
- More bitvectors in partitions with larger documents (green).
Results for Bitvectors

![Graph showing the performance of S16+bitvectors with space (bits/posting) on the x-axis and time (ms/query) on the y-axis. The graph compares the performance of two methods: S16+bitvectors(rand−p−rand) (blue squares) and S16+bitvectors(td−p−rand) (red triangles). The graph includes data points for different space and time values.]
Ranking

• Direct improvements:
  • Delta compression and skips are often used in ranking systems.

• Expected improvements:
  • Locality of access from increased density of lists.
  • Sparse intermediate results.
  • Structures/processing that adapts to each partition.
Potential Improvements

• Within a partition:
  • Tune algorithms in each partition to fit the data in partition.

• Across partitions:
  • Run on subset of partitions to decide on subsequent processing. For example, decide on AND vs. Weak-AND processing for other partitions.
Distribution in Practice

- Use a hierarchy of distribution/ordering mechanisms in practice, for example:
  - Tier documents by global relevance (e.g., PageRank).
  - URL domain distribution (e.g., .gov) within a tier.
  - Document Size Distribution within a domain.
  - Order by URL within partition.
Conclusions

• We have shown that document size distribution improves:
  • Compression of postings lists.
  • Locality of access inside structures.
  • Performance of skips and bitvectors.

• Document size distribution is broadly applicable.
Questions/Comments
Related Ideas - Outline

- Databases:
  - Order by row size.
  - Order by usage.

- Using terms-in-document partitions:
  - Solving model constants.

- Other:
  - Replication vs. Partitioning.
  - Error identification.
Database Row Size

• Ordering by document size improves search systems.
• Reordering in databases is restricted to clustering by attributes.
• Use ordering by row size in database systems?
  • Number of non-null values.
  • Number of characters in row values.
Density vs. Usage

- Ordering by document size improves search systems.
- Document size correlates with likelihood of being in result list (at least for conjunctive-AND queries).
- For database queries, ordering rows by their usage in queries may produce similar improvements.
  - Improves locality of access and filtering of indexes.
  - Ordering by recency of update.
  - Ordering by recency of access.
Solving Model Constants

• Each document size distributed partition has different data distributions.

• Use partitions to solve system of linear equations and get performance model constants.
  • Random distribution gives singular system.
  • Normally need to isolate parameters to solve for constants.
    • Isolate by changing dataset or query workload.
    • Could be related to query mix.
Replication vs. Partitioning

- To improve throughput, should a search system add replicas or do more document partitioning?
- Assume: linear scaling of partitioning.
- For example:
  - 1x4GB partition = 1000 qps
  - 2x4GB replicas = 2000 qps
  - 2x2GB partitions = 2000 qps, but now have 2GB of memory per partition to improve throughput.
Error Identification

- Processing errors are more common with more data, and some application cannot tolerate errors (legal).

- While processing:
  - Verify data read from lists (error correcting codes).
  - Verify data decoded (encode last value in uncompressed form and compare).

- Post processing:
  - Verify intersection (result size boundary checks; signature checks).
  - Verify ranking (boundary checks).
Thank you.

Questions?

/* Comments */