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

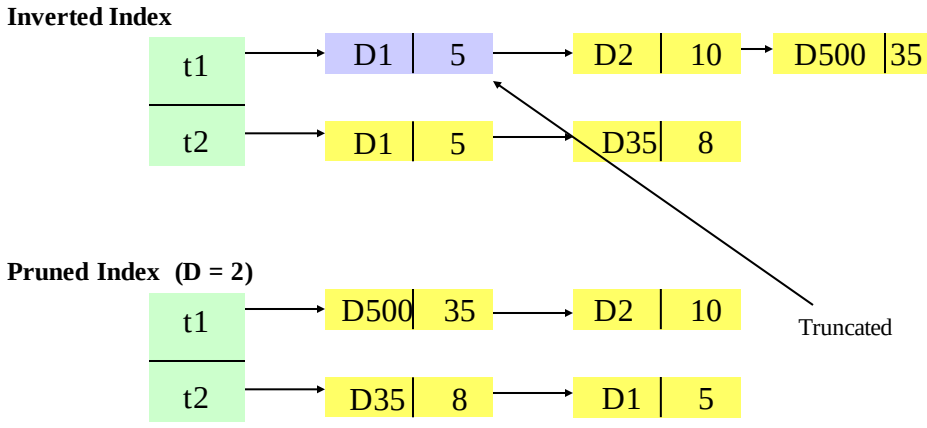
Index Pruning (Top Doc)

- 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

3

Inverted & Pruned Indices



4

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

5

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)

6

Processing Least Frequent Terms First

- Process terms in "goodness" order
  - e.g., *idf* , *qtf*, tf_{max} .*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)

7

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.

8

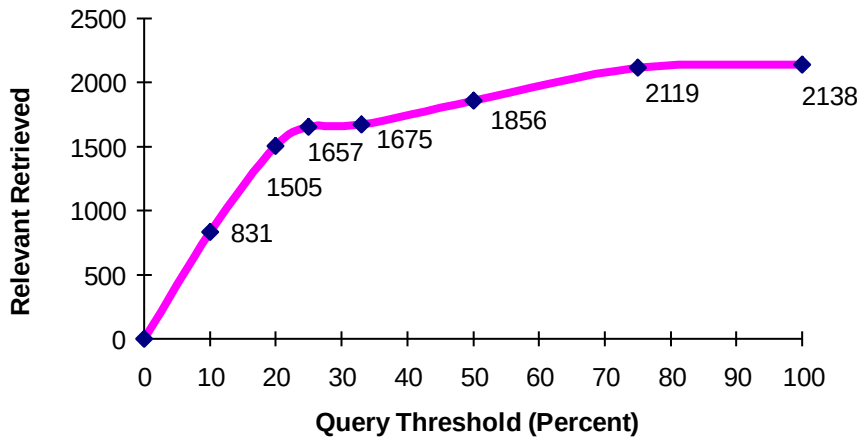
Query Threshold

- 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.

term1 term2	**threshold = 20%**
term3 term4 term5	**threshold = 50%**
term6 term7 term8	**threshold = 80%**
term9 term10	

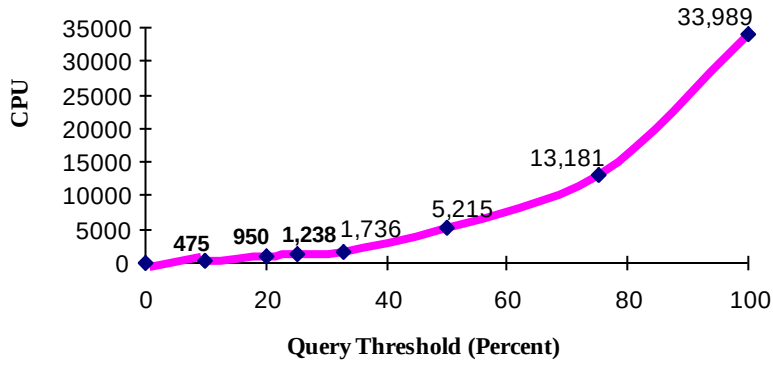
9

Relevant Retrieved for Varying Query Thresholds



10

Run Time as a Function of Query Thresholds



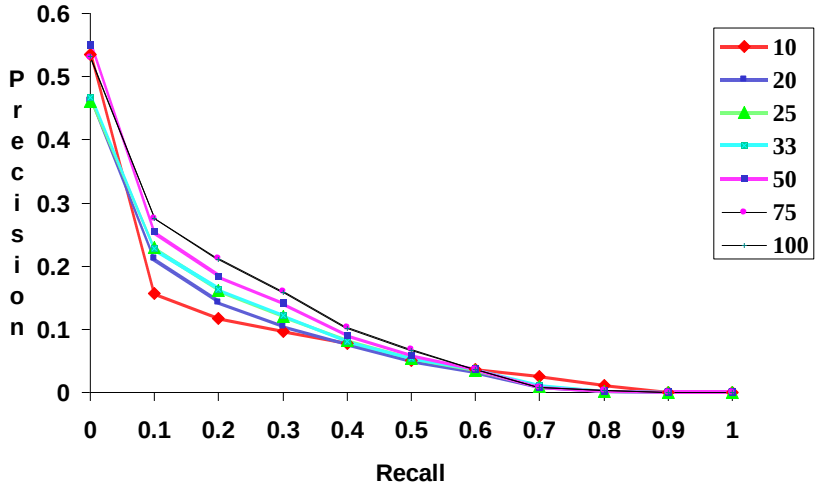
11

Relevant New Documents Per CPU Cycle

Threshold	Relevant Retrieved	CPU Cycles	New Relevant Docs per Cycle
10	831	475	1.75
20	1505	950	1.58
25	1601	1238	1.29
33	1657	1736	0.95
50	1856	5215	0.36
75	2119	13181	0.16
100	2138	33989	0.06

12

Precision/Recall



13

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.

14