

Efficiency – Index Pruning & Query Processing

(CS429)

Nazli Goharian

nazli@ir.iit.edu

Slides are *mostly* based on Information Retrieval Algorithms and Heuristics, Grossman, Frieder

Efficiency Techniques

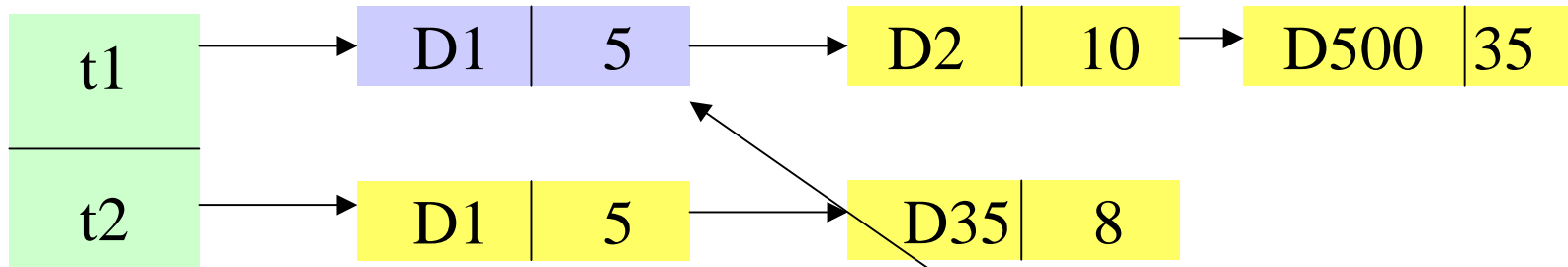
- Indexing
- Compression
 - Index Pruning (Top Doc)
 - Efficient Query Processing
 - Duplicate Document Detection

Index Pruning (Top Doc)

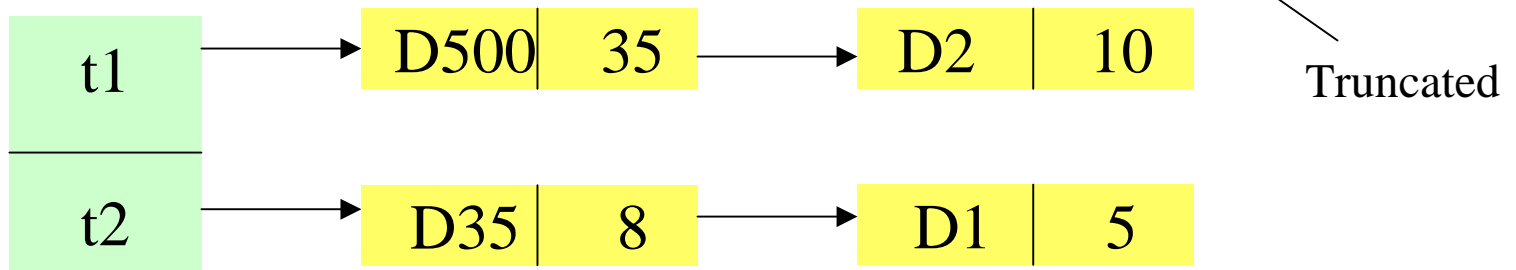
- Instead of retrieving the whole posting list, retrieve the top x documents where the documents are ranked by weight.
- Term specific pruning vs. uniform pruning
- A separate structure with sorted, truncated posting lists may be produced.
- Experimentation results: 70% of index achieves similar average precision as full index

Inverted & Pruned Indices

Inverted Index



Pruned Index ($D = 2$)



Pruned Index Summary

- Pro
 - Avoids need to retrieve the entire posting list
 - Dramatic savings on efficiency for large posting lists
- Con
 - Not feasible for Boolean queries
 - Can miss some relevant documents due to truncation

Efficient Query 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 (Query Thresholds)

Processing Least Frequent Terms First

- Process terms in “goodness” order
 - (e.g., decreasing idf values)
- Terminate processing after d documents are assigned non-zero scores

OR

- Continue processing for the above d non-zero scores with remaining query terms
 - Further option: treat remaining terms as a conjunctive (AND) condition versus traditional vector space disjunctive (OR)

Processing Least Frequent Terms First

For each term t in the query

Obtain posting list p for documents that contain t

For each document x in the list of d documents

Scan posting list p for x

if x exists

Update score for document x

Processing Least Frequent Terms First (“Goodness Options”)

- Stop processing based on I/O threshold after processing x “good” terms.
 - Good terms identified based on:
 - Based on df (*using top 25%-75% no degradation in result for TREC-4*)
 - Based on maximum estimated weight ($tf_{max} idf$)
 - Weight of a disk page in the posting list - to determine which posting list to process first

Modifying Inverted Index to Support Fast Scanning

- **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 , by checking the first doc id of two consecutive partitions.
- If doc x is not found, jump to next partition. Otherwise, scan current partition.
=> A small size d , 1000, resulted to the best CPU time for a set of TREC queries [1996].

Modifying Inverted Index to Support Fast Scanning

- Order each posting list by decreasing order of term frequency.
- Create an entry in index for each page of a given posting list. This entry indicates the maximum term frequency (tf_{max}) on each page of a posting list
- Order the pages of a posting list based on

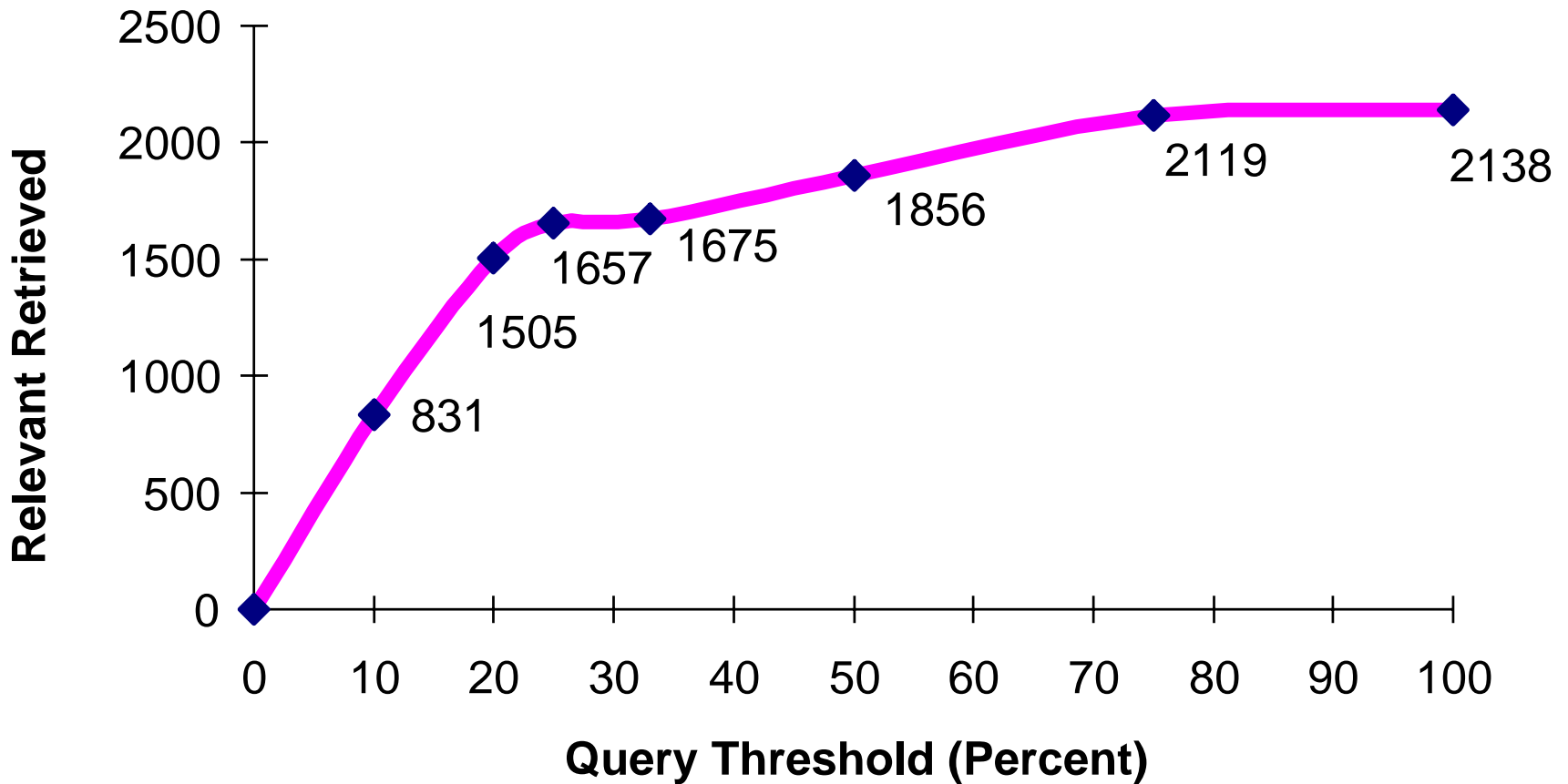
$$tf_{max} * idf * f(I)$$

Query Threshold

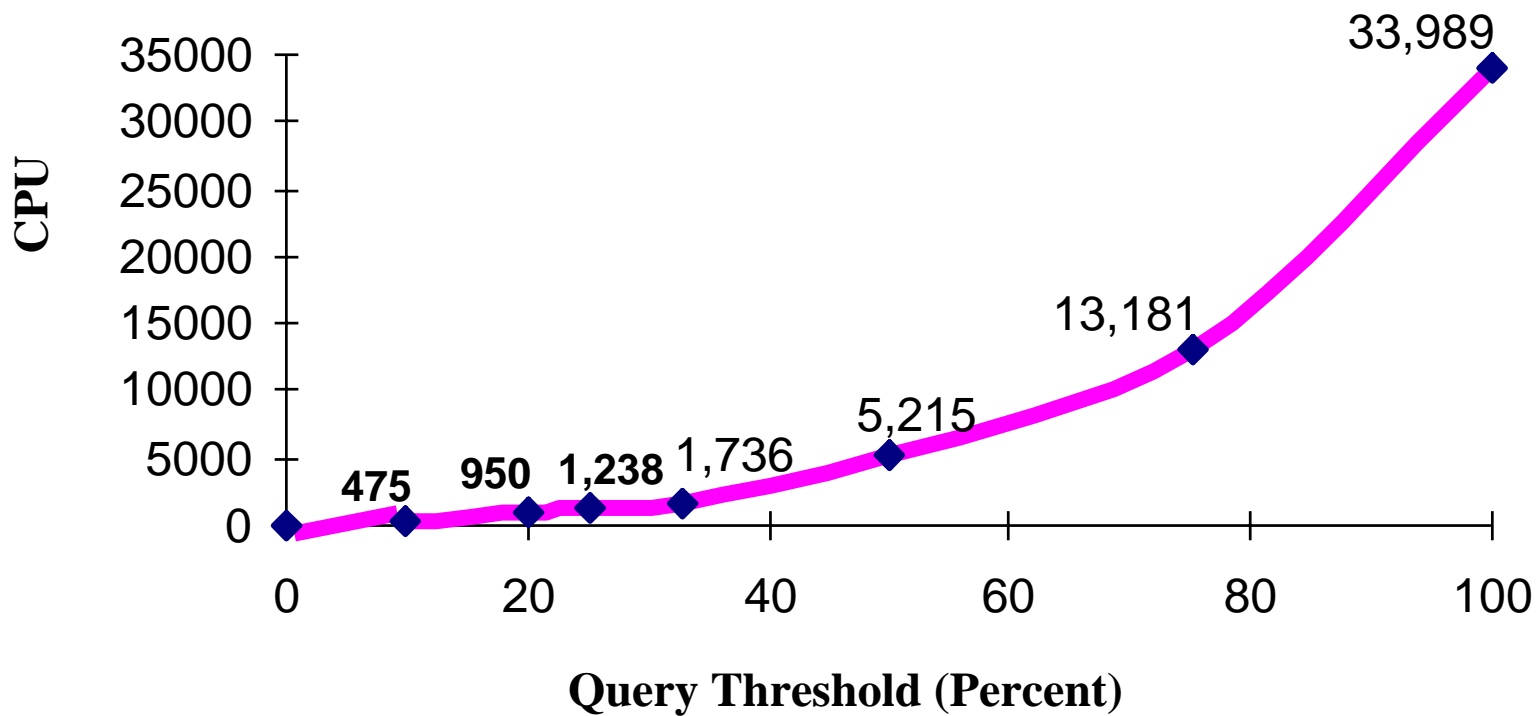
- Consider a query with terms $t_1, t_2, t_3, \dots, t_n$.
- Define a threshold as the percentage of terms taken from the original query in a newly created reduced query.

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

Relevant Retrieved for Varying Query Thresholds



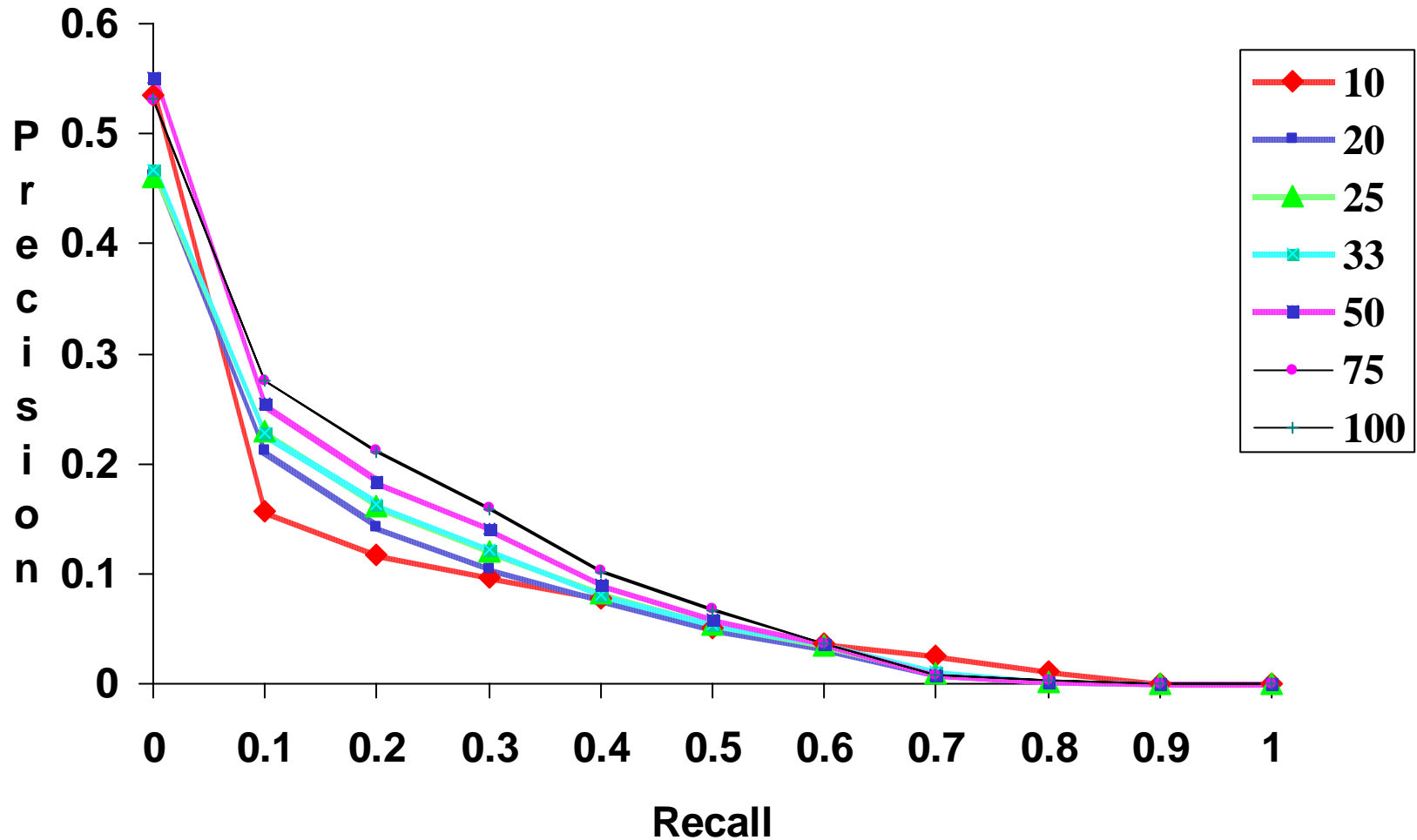
Run Time as a Function of Query Thresholds



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

Precision/Recall



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.