



Lecture 12:

An Introduction to Top-k Document Retrieval

Hadi Amiri

hadi@comp.nus.edu.sg

Top-k Document Retrieval

The image shows a Google search interface for the query "national university of singapore". The search bar at the top contains the text "national university of singapore" and a blue search button. Below the search bar, the word "Search" is displayed in red, followed by the text "About 9,820,000 results (0.16 seconds)". A blue box highlights the number "9,820,000".

On the left side, there is a navigation menu with categories: "Everything", "Images", "Maps", "Videos", "News", "Shopping", and "More". Below this, there is a section for "Any time" with options: "Past hour", "Past 24 hours", "Past 2 days", "Past week", "Past month", "Past year", "Custom range...", and "More search tools".

The main search results area displays several items:

- An advertisement for "NTU Admissions 2012" with the URL admissions.ntu.edu.sg/ and the text "Realise Your Aspirations with NTU."
- An advertisement for "18-Month Part-Time MBA | smu.edu.sg" with the URL www.smu.edu.sg/MBA and the text "For an Accelerated, Innovative Learning in the heart..."
- The "NUS - Home" result with the URL www.nus.edu.sg/ and a snippet: "1 day ago - A research-intensive university with an entrepreneurial... is ranked consistently as one of the world's top universities. We offer ..."
- Other results include "NUS Business School", "Library", "Faculties & Schools", and "Office of Deputy President...".

On the right side, there is a map titled "National University of Singapore" showing the location of the university in Singapore. The map includes labels for "Singapore Polytechnic", "Clementi Woods", "Queenstown", and "Ayer Rajah Expy".

A blue callout box with a pointer to the search results contains the text: "Search engines usually contain millions of relevant documents for each query, but we only examine a few docs, e.g. top-K."

Top-k Document Retrieval

Google national university of singapore

Search About 9,820,000 results (0.16 seconds)

Everything
Images
Maps
Videos
News
Shopping
More

Any time
Past hour
Past 24 hours
Past 2 days
Past week
Past month
Past year
Custom range...
More search tools

NTU Admissions 2012 - Put Your Vision Into Action At NTU.
admissions.ntu.edu.sg/
Realise Your Aspirations with NTU.

18-Month Part-Time MBA | smu.edu.sg
www.smu.edu.sg/MBA
For an Accelerated, Innovative Learning in the heart of Singapore.

NUS - Home
www.nus.edu.sg/
1 day ago - A research-intensive university with an entrepreneurial dimension, ranked consistently as one of the world's top universities. We offer ...

NUS Business School
National University of Singapore is a leading university in the ...

SoC
Graduate - Computer Science - Undergraduate - About Us - ...

Undergraduate Programmes
Bachelor/Master of Engineering or Bachelor/Master of Science or ...


Library
Top page of the library, with links to the catalogue, electronic ...

Faculties & Schools
Home > Education > Academic > Faculties & Schools ...

Office of Deputy President ...
Office of the Deputy President (Research & Technology) ...

Search nus.edu.sg

National University of Singapore



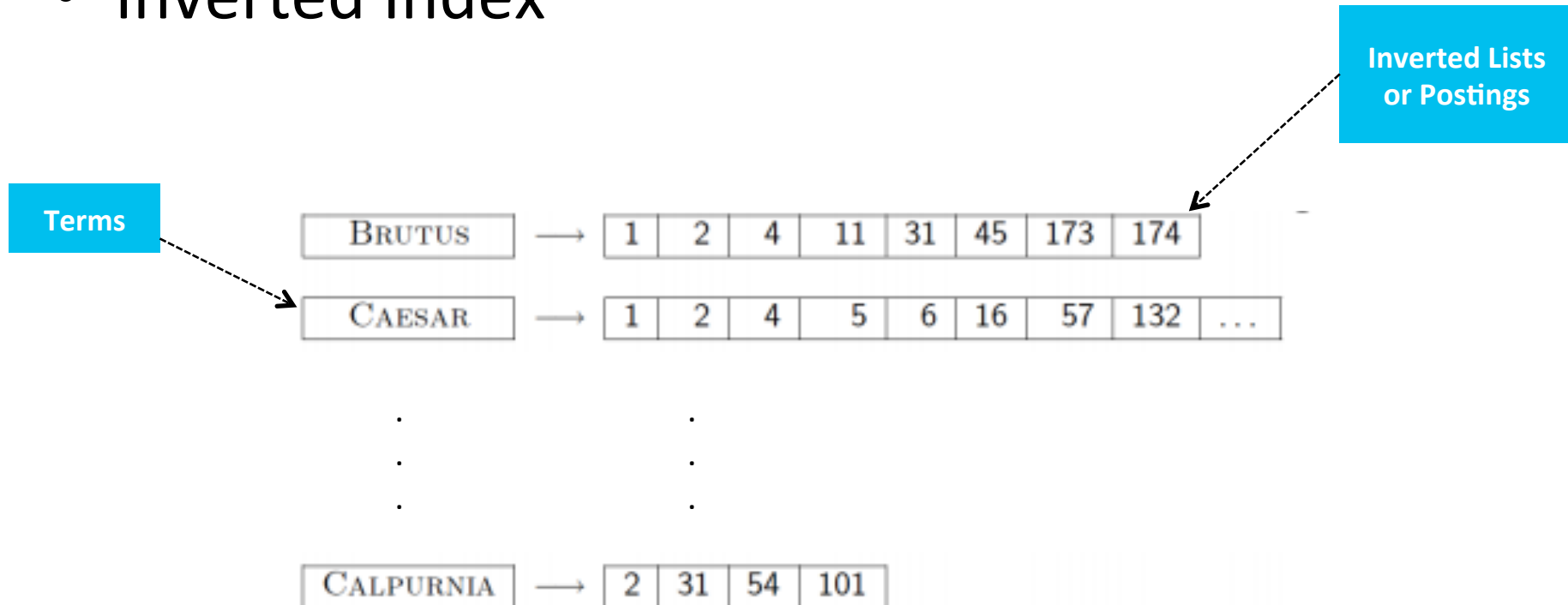
How can we retrieve documents efficiently?

Outline

- Indexing and Compression
- Index Organization
- Index Traversal
- Query Processing and Retrieval
- Top-k processing Algorithms
- Summary

Indexing and Compression

- Inverted Index



Each posting describes all places where term occurs in the collection

A simple and efficient data structure that allows us to find documents that contain a particular term.

Indexing and Compression

- Terms:
 - Phrases, e.g. apple tree
 - N-Grams, efficient apple light
 - Patterns, save apple power
- Selecting good terms can highly improve the efficiency of the systems.

Indexing and Compression

- Supplementary Information
 - Posting level:
 - docID
 - term frequency,
 - positions, plus other context such as font size etc.
 - Term level:
 - tID,
 - (Inverted) document frequency
 - Or any pre-computed impact score
- Some statistics are also stored outside the index:
 - document lengths or global scores such as Pagerank

Indexing and Compression

- Index Compression
 - The inverted lists of common query terms may consist of millions or even billions of postings.
 - To allow faster access to lists on disk, and limit the memory needed, sophisticated compression techniques are designed to reduce the size of each inverted list.

Outline

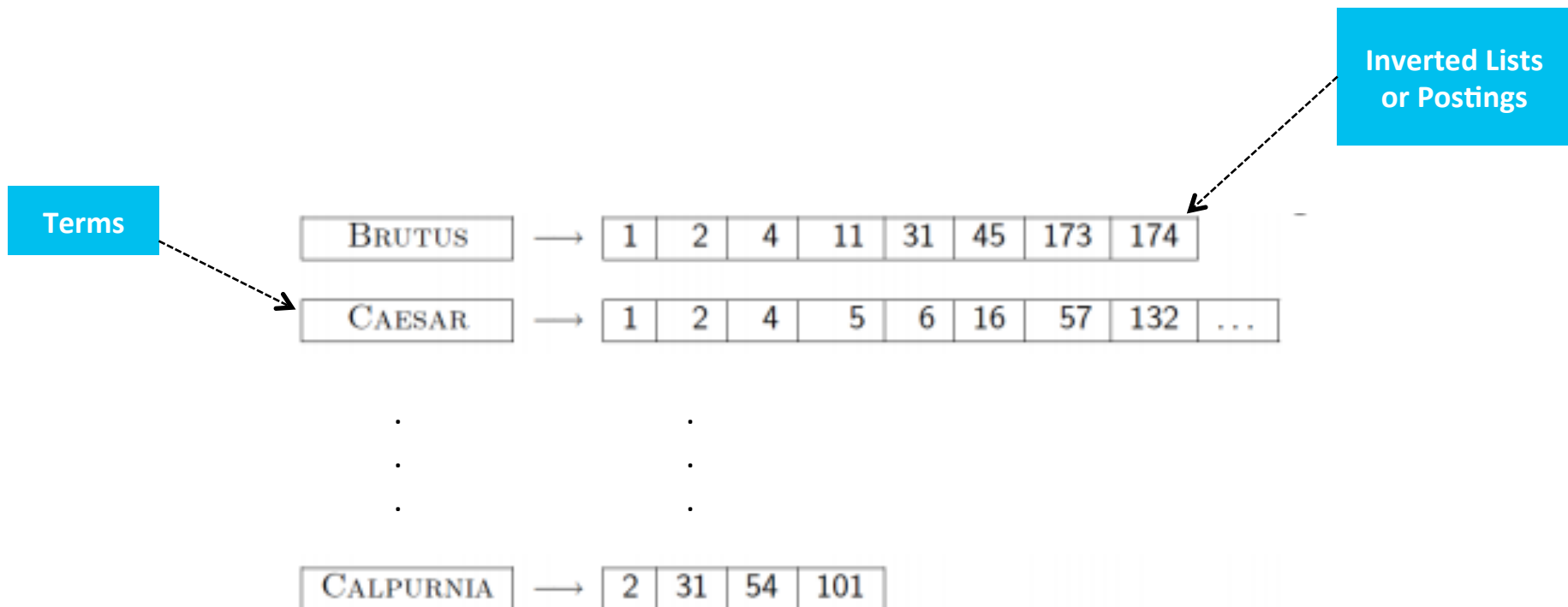
- Indexing and Compression
- Index Organization
- Index Traversal
- Query Processing and Retrieval
- Top-k processing Algorithms
- Summary

Index Organization

- How to order postings of the indexes:
 - Doc Sorted Indexes
 - Impact Sorted Indexes
 - Impact Layered Indexes

Index Organization

- Doc Sorted Indexes
 - It is the basic approach: the postings in each inverted list are sorted by document ID.



Index Organization

- Impact Sorted Indexes
 - Postings in each list are sorted by the term contribution to the score of a document (impact).
 - For example: term frequency?
 - What is a good impact factor?

Index Organization

- Impact Layered Indexes
 - Postings in each list are partitioned into a number of layers, such that all postings in layer i have a higher impact than those in layer $i+1$, and then sort the postings in each layer by docID.

Index Organization

- Both Impact-sorted and impact-layered indexes place the most promising postings close to the start of the lists.
- Problem with impact-sorted indexes is:
 - compression could suffer as docID gaps in the inverted lists may be very large.
 - In this case, an impact-layered index that uses a small number of appropriately chosen layers may provide a better alternative.
 - What if the number of distinct impact scores is small?

Outline

- Indexing and Compression
- Index Organization
- Index Traversal
- Query Processing and Retrieval
- Top-k processing Algorithms
- Summary

Index Traversal

- Term-At-A-Time (**TAAT**):
 - We first access one term and then move to the next term.
 - We have to use a temporary data structure to keep track of currently active candidates.
 - The complete score of a document is unknown until all query terms are processed.

Index Traversal

- Document-at-a-time (**DAAT**)
 - In the DAAT, all relevant inverted lists are simultaneously processed, and the score of a document is fully computed before moving to the next document.

Index Traversal

- Two advantages of DAAT over TAAT:
 1. DAAT only requires a small amount of memory to store (e.g., current Top 10) results
 - while TAAT needs much more memory which is usually expensive to maintain.

Index Traversal

- Two advantages of DAAT over TAAT:
 2. DAAT methods can easily identify whether query terms satisfy some given conditions in a document (e.g., phrase).
 - while TAAT can't.
 - It needs to keep the occurrences of the first term within the document in the intermediate results, in order to verify whether the second term satisfies the constraint.

Outline

- Indexing and Compression
- Index Organization
- Index Traversal
- Query Processing and Retrieval
- Top-k processing Algorithms
- Summary

Query Processing and Retrieval

- Query types
 - Disjunctive and Conjunctive
 - the most basic form of queries are *Boolean queries*: (**apple AND orange**) OR **pear**
 - How to interpret the Free text queries like: **apple orange pear**?
 - disjunctive queries tend to be significantly more expensive than conjunctive queries as they have to evaluate many more documents.
 - Search Engines are optimized for conjunctive queries.

Query Processing and Retrieval

- More complex queries
 - Indri query language

Name	Example	Behavior
term	dog	occurrences of dog (Indri will stem and stop)
"term"	"dog"	occurrences of dog (Indri will not stem or stop)
ordered window	#odn(blue car) - <i>or</i> - #n(blue car)	blue <i>n</i> words or less before car
unordered window	#uwn(blue car)	blue within <i>n</i> words of car
synonym list	#syn(car automobile)	occurrences of car or automobile
weighted synonym	#wsyn(1.0 car 0.5 automobile)	like synonym, but only counts occurrences of automobile as 0.5 of an occurrence
any operator	#any:person	all occurrences of the person field

Query Processing and Retrieval

- Retrieval Using the Postings

$$S(d, q) = \alpha \cdot G(d) + \beta \cdot IR(d, q)$$

- $S(d, q)$: overall score for the doc d with respect to the query q ,
- $G(d)$: global or static score of d ,
- $IR(d, q)$: the query-dependent score for d with respect to q ,
- β and α are parameters satisfying $\beta + \alpha = 1$.

Query Processing and Retrieval

- Fig X from the book

Outline

- Indexing and Compression
- Index Organization
- Index Traversal
- Query Processing and Retrieval
- Top-k processing Algorithms
- Summary

Top-k processing Algorithms

- As the inverted lists for common terms could be very long, we want to process a few (e.g. $k \cdot c$) postings during query processing.
- Preliminaries
 - Mechanism
 - Exhaustive
 - Non- Exhaustive
 - Safety
 - Safe
 - Not safe
 - When does it happen?

Top-k processing Algorithms

- Mechanism
 - Exhaustive
 - The algorithm is exhaustive if it fully evaluates all documents that satisfy required conditions.
 - Non- Exhaustive
 - The algorithms that use early termination so that ignore evaluating many of the documents that satisfy required conditions.

Top-k processing Algorithms

- Safety
 - Safe: the same results as in the exhaustive approach:
 - the same set of top-k documents in the same order with the same scores
 - Not safe
 - try to return search results that are somehow similar (or of similar quality) to the exhaustive approach.

Top-k processing Algorithms

- When can the early termination happen?
 - When the minimum score in the current Top-k results is larger than the maximal possible score of the unprocessed documents
 - and sometimes also the rank order of the current Top-k results will not be changed.
 - The way we organize the inverted lists plays a vital role as it determines the number of documents to be processed.
 - Stop early
 - Skip within Lists
 - Omit Lists
 - Score Only Partially

Top-k processing Algorithms

- Stop Early
 - We stop the traversal of the index as soon as we have the top-k results.
 - In this case, we assume that postings are arranged such that the most promising documents appear early.
 - Impact-sorted and impact-layered indexes

Top-k processing Algorithms

- Skip within Lists
 - Smart pointer movement techniques are used to skip many documents that would be evaluated by an exhaustive algorithm.
 - Proper when postings in each list are sorted by docIDs, i.e. the promising documents are spread out through-out the inverted lists.

Top-k processing Algorithms

- Omit Lists
 - One or more lists for the query terms are completely ignored, if they do not affect the final results by much.

Top-k processing Algorithms

- Score Only Partially
 - We partially evaluate a document by computing only some term scores, or by computing approximate scores.

Top-k processing Algorithms

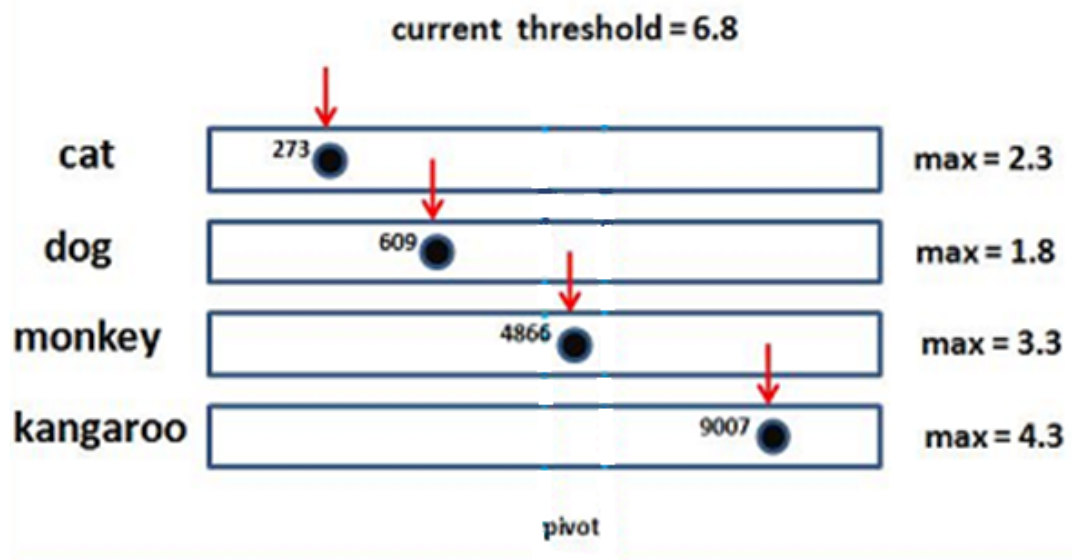
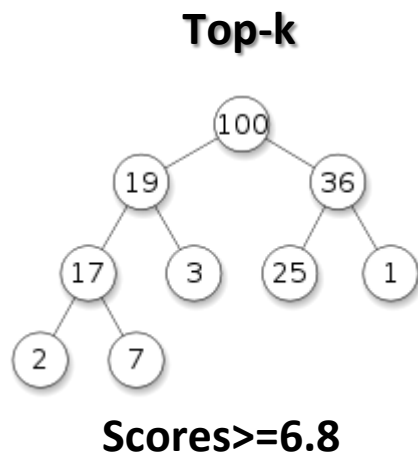
- Today's Focus is on Top-k retrieval algorithms that are optimized for
 - Disjunctive queries (Type of Query)
 - Document Sorted Indexes (Index Organization)
 - DAAT (Index Traversal)
 - Safe algorithms
- How about other requirements like
 - Positional indexing, proximity, etc??

Top-k processing Algorithms

- We introduce two major algorithms
 - Weak and or weighted and (WAND)
 - Block Max
- There are also other methods
 - Optimized Block Max
 - etc

Top-k processing Algorithms

- WAND
 - Query terms: [dog, cat, kangaroo, monkey]
 - Current docIDs pointers: 609, 273, 9007, and 4866.
 - maximum impact score for each query term
 - Threshold: 6.8



Top-k processing Algorithms

- Each list
 - has a pointer that points to a "current" posting in the list
 - has a maximum impact score.
 - could be kept in the term dictionary of the index.
- Current threshold: the lowest score in the heap that contains the top-k results found thus far.
- Pointers move forward as the query is being processed.

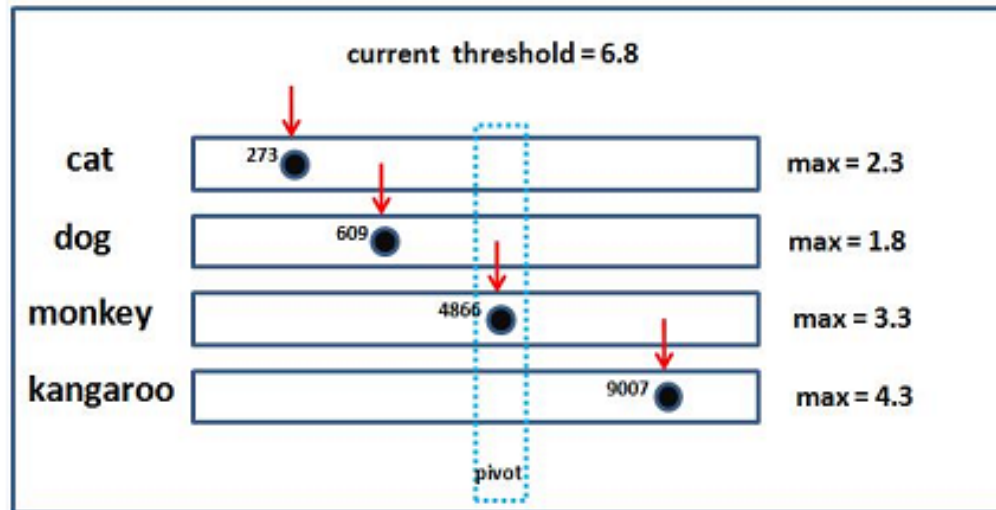
Top-k processing Algorithms

- WAND
 - Standard Document-sorted index
 - DAAT for index traversal
 - Thus, any posting to the left of the pointers have already been processed.

Top-k processing Algorithms

- WAND algorithm
 - Sort the lists from top to bottom according to the docIDs of the current pointers,
 - sum up the maximum scores of the lists from top to bottom until we reach a score no smaller than the threshold.
 - The current docID of this list is the pivot ID.
 - Claim: the smallest docID that can make it into the top-k is the pivot ID.
 - Thus, the current pointers of the previous query terms are forwarded to the first postings in their lists with docIDs greater than or equal the pivot ID.
 - If pivot ID appears in *ALL* the lists of the previous query terms
 - then we evaluate pivot ID.
 - Otherwise,
 - we sort the lists according to the current docIDs and pivot again.

WAND Example



- Sort the lists and sum up the maximum scores
 - Third list ($2.3 + 1.8 + 3.3 > 6.8$). The current docID of this list is the pivot, 4866.
- Smallest docID that can make it into the top-k is 4866.
 - Thus, move the top two pointers forward to the first postings in their lists with docIDs at least 4866.

Top-k processing Algorithms

- Maximum impact score for a query term t

$$UB_t \geq \alpha_t \max(w(t, d_1), w(t, d_2), \dots).$$

- Score of a document d for query q is

$$\text{Score}(d, q) = \sum_{t \in q \cap d} \alpha_t w(t, d)$$

Top-k processing Algorithms

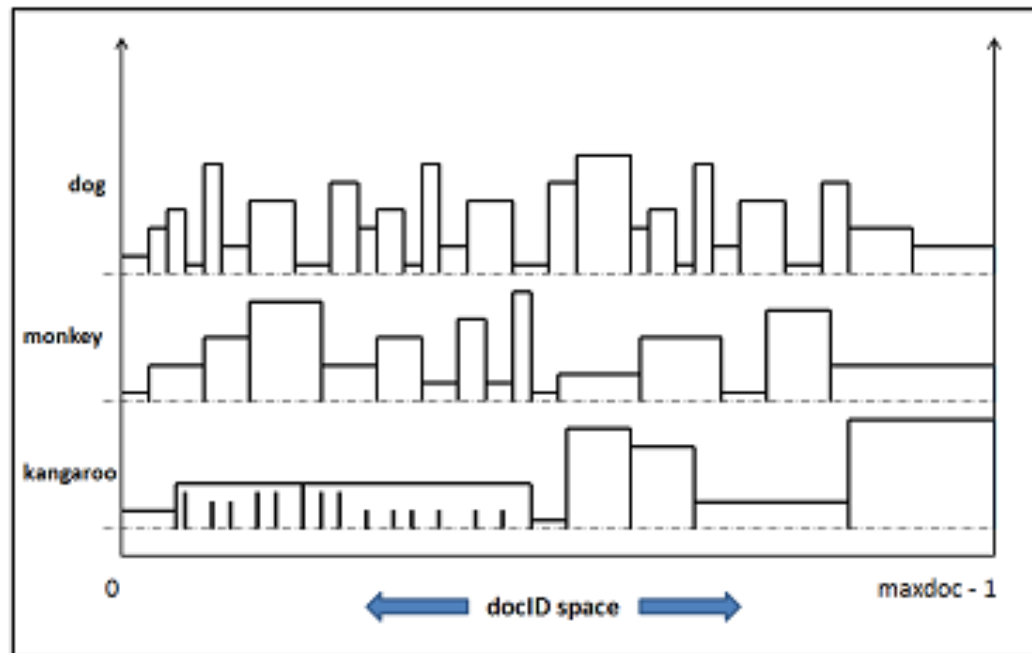
- Block Max
 - max impact scores in WAND can be much larger than the individual doc scores or the average.

Top-k processing Algorithms

- Block Max
 - Splits the inverted lists into blocks of, say, 64 or 128 docIDs such that each block can be decompressed separately.
 - Create an extra table, which stores for each block
 - The maximum (or minimum) docID,
 - Maximum impact value for each block, and
 - The block size.

Top-k processing Algorithms

- we get a piece-wise upper-bound approximation of the impact scores in the lists



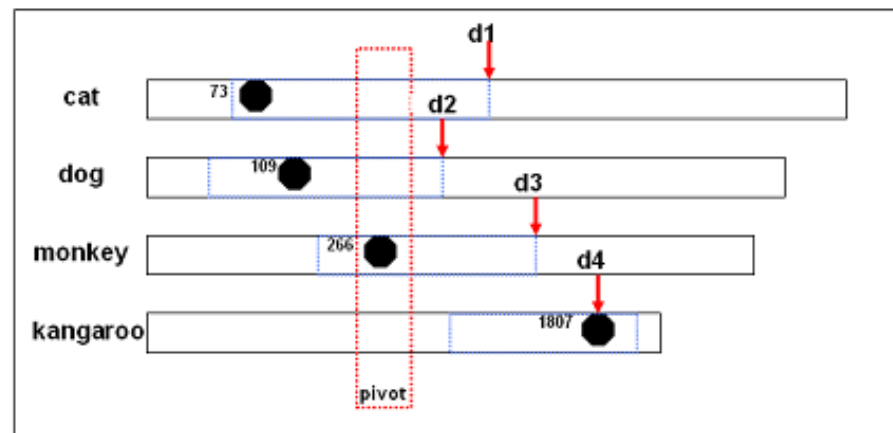
Three inverted lists where lists are piecewise upper-bounded by the maximum scores in each block. Inside each block we have various values, including many (implied) zero values, that may be retrieved by decompressing the block.

Top-k processing Algorithms

- Block Max algorithm
 - Sort the lists from top to bottom according to the docIDs of the current postings,
 - Find the pivot (as we did in WAND)
 - Use the global maximum scores to determine a candidate pivot, as in WAND,
 - Use the block maximum scores to check if the candidate pivot is a real pivot
 - If it is, similar to WAND,
 - move the current pointers of the previous query terms to the first postings in their lists with docIDs greater than or equal the pivot ID.
 - Else
 - Move to $d = \min(d_i)$ where d_i are the maximum docIDs in the current blocks.

Block Max Example

- Block Max
 - Query terms: [dog, cat, kangaroo, monkey]
 - Current docIDs: 73, 109, 266, and 1807.
 - docID 266 is pivot and it fails to make it into top k results.



we enable better skipping by choosing $\min(d_1; d_2; d_3; d_4)$ as the next possible candidate, instead of $266 + 1$

Top-k processing Algorithms

- WAND and Block Max Performance

TREC 2005						
	avg	2	3	4	5	> 5
exhaustive OR	369.3	62.1	238.9	515.2	778.3	1501.4
WAND	64.4	23.5	43.7	73.4	98.9	265.9
SC	63.5	14.2	37.5	119.7	172.9	316.9
BMW	21.2	3.5	12.7	25.2	39	104
exhaustive AND	6.86	6.4	7.3	9.2	4.7	5.9

Average query processing time in ms for different numbers of query terms on the TREC 2005 query logs.

Exhaustive OR, WAND, SC, and BMW (Block Max) are for disjunctive queries, while Exhaustive AND is for conjunctive queries

Top-k processing Algorithms

- WAND and Block Max Performance

TREC 2006						
	avg	2	3	4	5	> 5
exhaustive OR	225.7	60	159.2	261.4	376	646.4
WAND	77.6	23.0	42.5	89.9	141.2	251.6
SC	64.3	12.2	36.7	75.6	117.2	226.3
BMW	27.9	4.07	11.52	33.6	54.5	114.2
exhaustive AND	11.4	10.3	10.8	14.0	15.4	15.2

Average query processing time in ms for different numbers of query terms on the TREC 2006 query logs.

Top-k processing Algorithms

- WAND and Block Max Performance

	evaluated docs	decoded ints
exhaustive OR	3815676	9356032
WAND	178391	6274432
SC	–	965248
BMW	21921	2642752
exhaustive AND	20026	1939584

The average number of evaluated docIDs and decoded instances for different methods on the TREC 2006 query log.

Outline

- Indexing and Compression
- Query Processing and Retrieval
- Index Organization
- Index Traversal
- Top-k processing Algorithms
- Summary

Summary

- Index Organization:
 - Doc Sorted Indexes
 - Impact Sorted Indexes
 - Impact Layered Indexes
- Index Traversal: TAAT, DAAT
- Query Processing and Retrieval
 - different types of queries. term-dependent and term-independent score
- Top-k processing Algorithms
 - Exhaustive and Non- Exhaustive
 - Safe and Not safe
 - WAND
 - skip the postings based on a pivot document and the global maximum scores of lists
 - Block Max
 - Improve upon WAND by splitting the docIDs into blocks. Then, skip based on the piecewise upper-bounded by the maximum scores in each block

Sample References

1. Dongdong Shan, Shuai Ding, Jing He, Hongfei Yan, and Xiaoming Li. 2012. Optimized top-k processing with global page scores on block-max indexes. In *Proceedings of the fifth ACM international conference on Web search and data mining (WSDM '12)*.
2. Shuai Ding and Torsten Suel. 2011. Faster top-k document retrieval using block-max indexes. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR '11)*.
3. Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. 2003. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the twelfth international conference on Information and knowledge management (CIKM '03)*.
4. Jonassen, S. and Bratsberg, S. 2011. Efficient Compressed Inverted Index Skipping for Disjunctive Text-Queries. *Advances in Information Retrieval* . (2011), 530-542
5. Tonello, N., Macdonald, C. and Ounis, I. 2010. Efficient dynamic pruning with proximity support. *Large-Scale Distributed Systems for Information Retrieval* (2010), 33-37.
6. Andrei Z. Broder, David Carmel, Michael Herscovici, Aya Soffer, and Jason Zien. Efficient query evaluation using a two-level retrieval process. In *Proceedings of the 12th ACM Conference on Information and Knowledge Management*, 2003.
7. T. Strohman and W. Bruce Croft. Efficient document retrieval in main memory. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007.