Skewed Partial Bitvectors for List Intersection

Andrew Kane and Frank Wm. Tompa
University of Waterloo

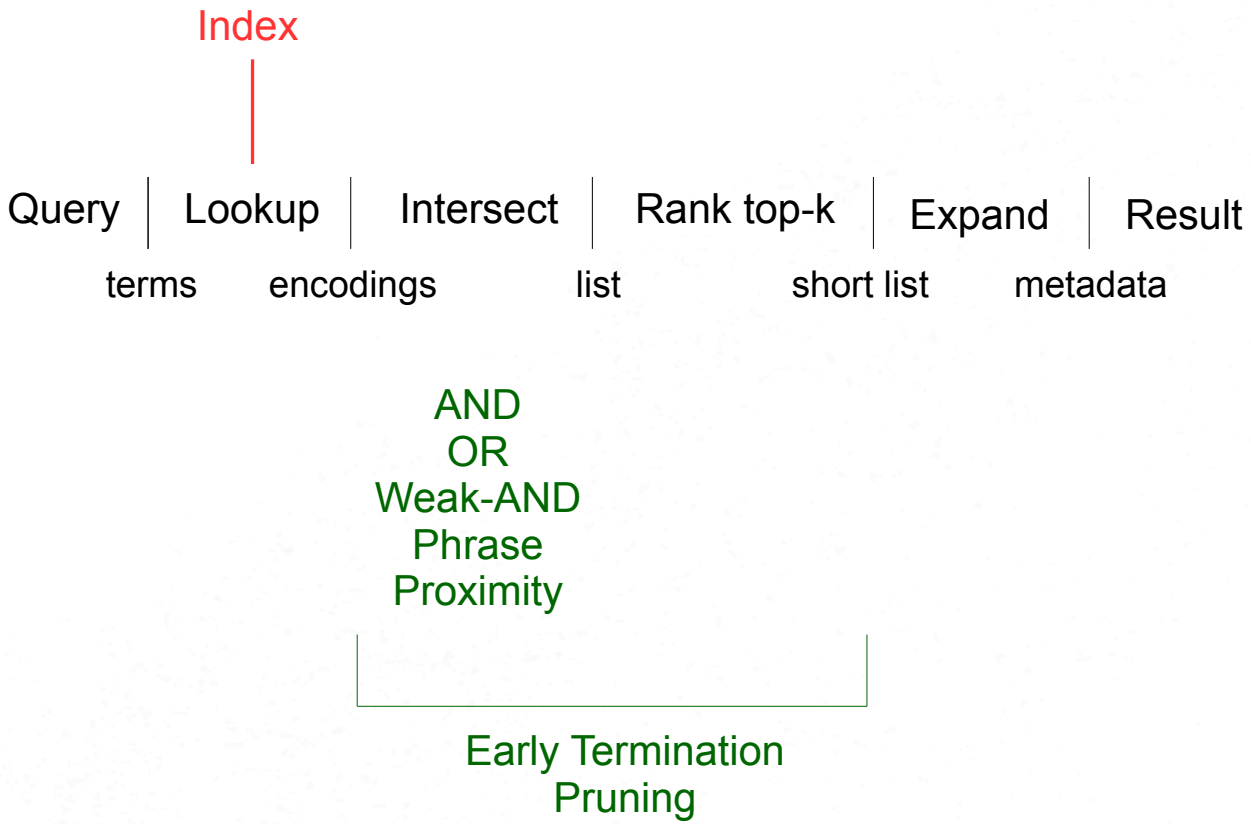
arkane@cs.uwaterloo.ca

SIGIR - July 8th 2014

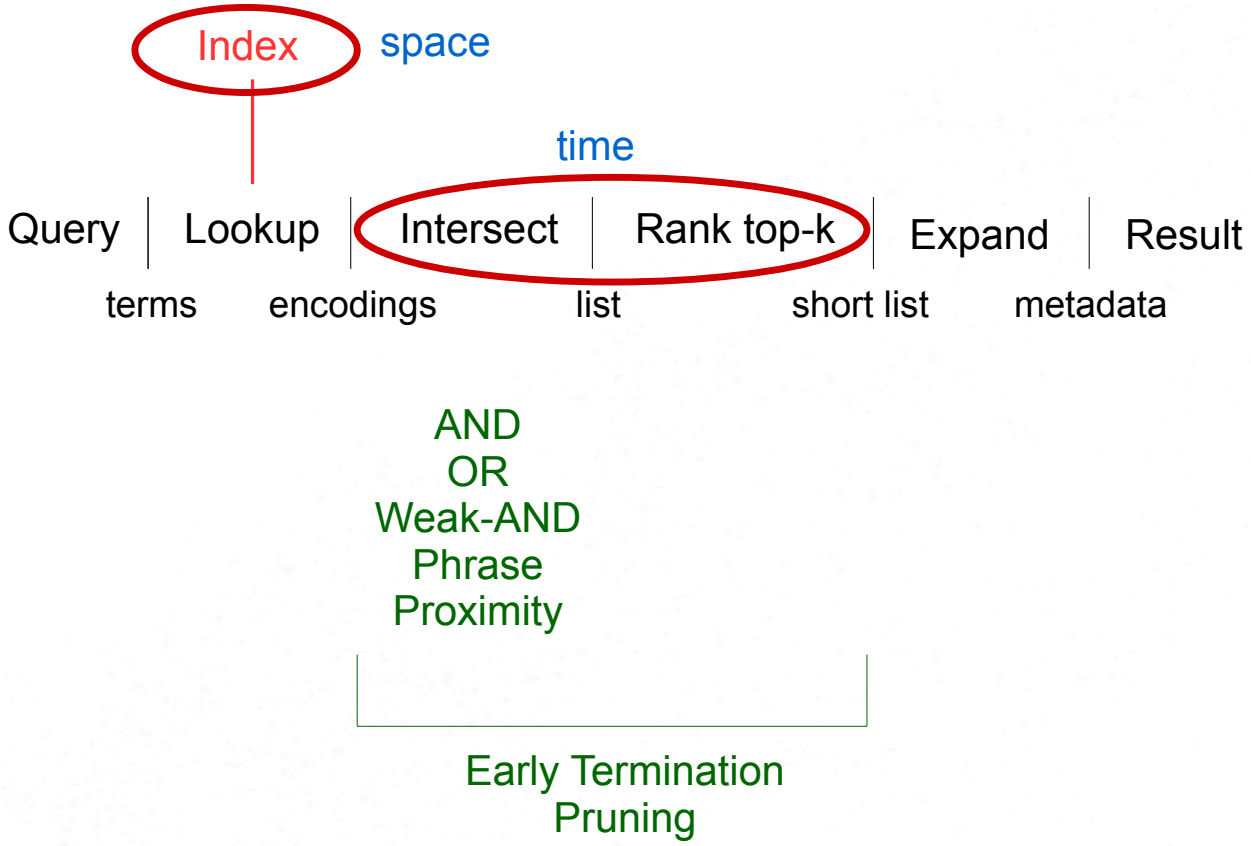
Outline

- Introduction to search engines.
- List Intersection - compression, skips, bitvectors.
- Document reordering - URL vs. terms-in-document.
- Partial bitvectors - semi-bitvectors with skewed groups.
- Conclusions.

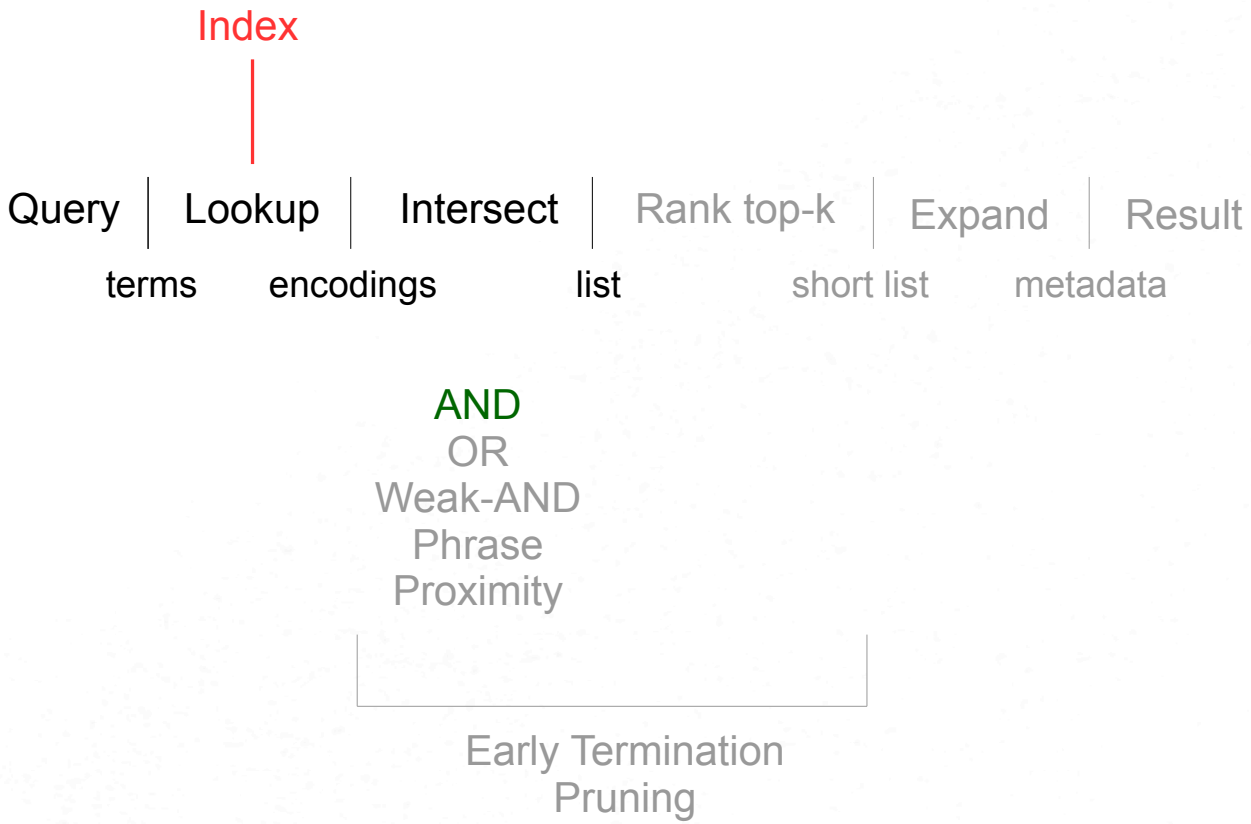
Search Engine Query Processing



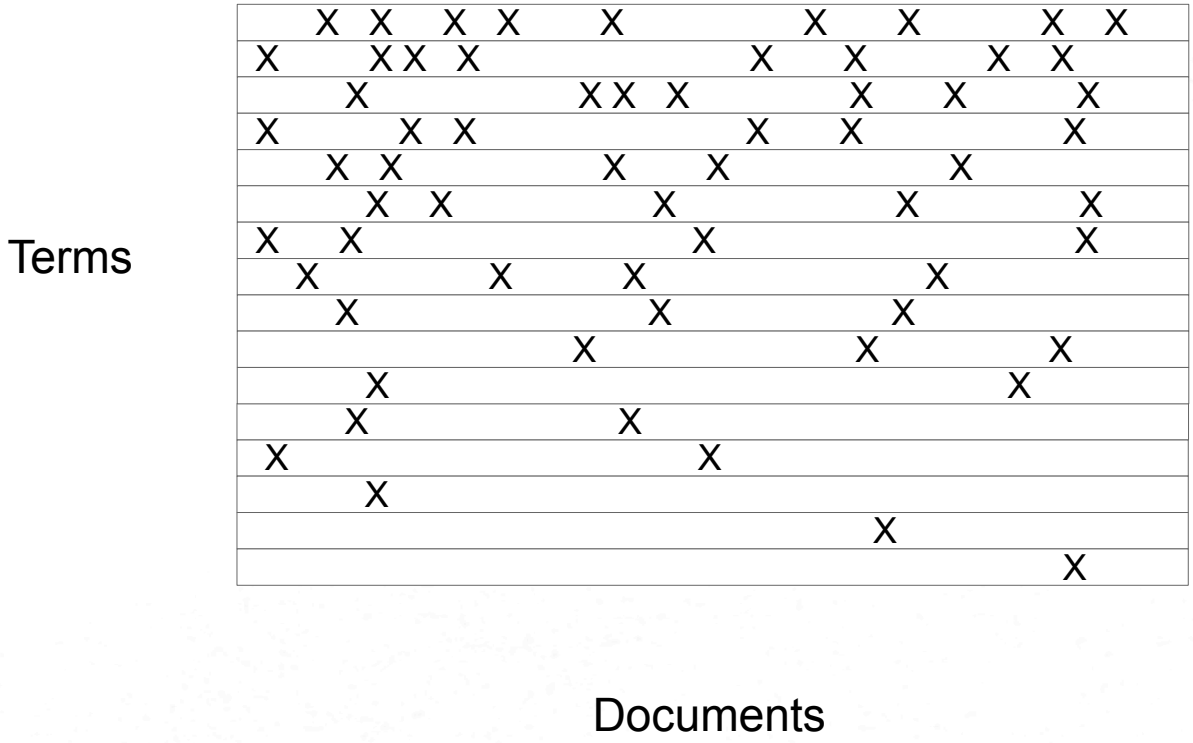
Search Engine Query Processing



Search Engine Query Processing

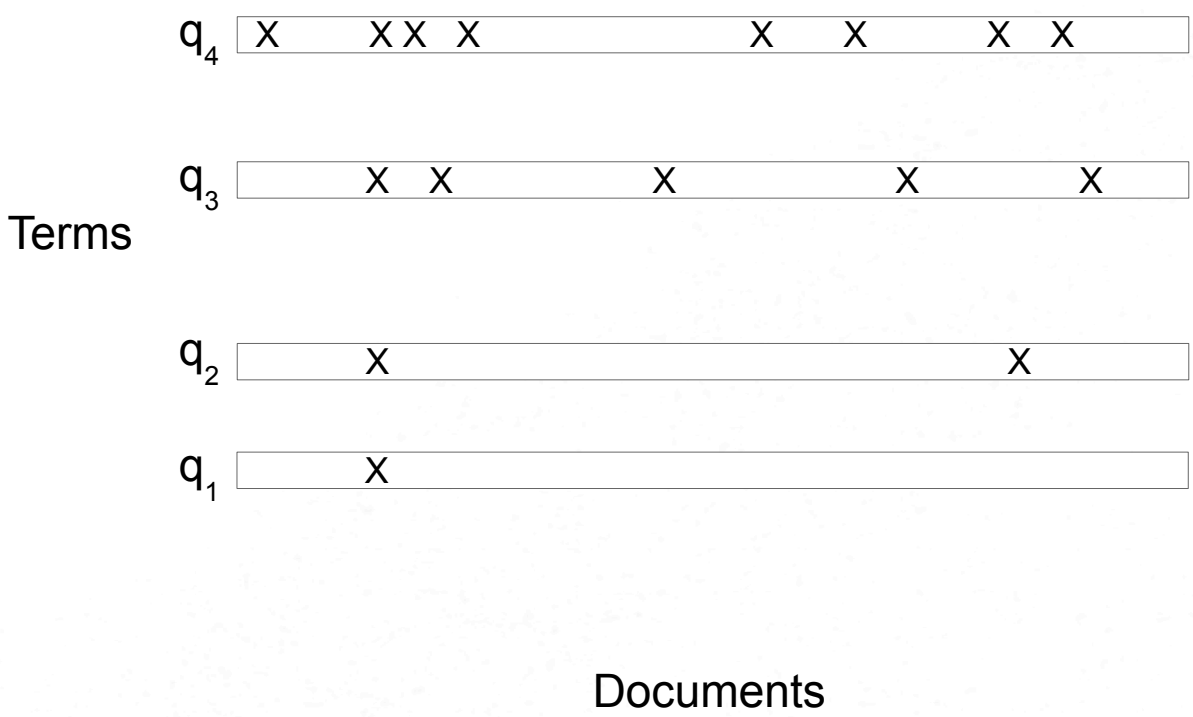


Data for List Intersection



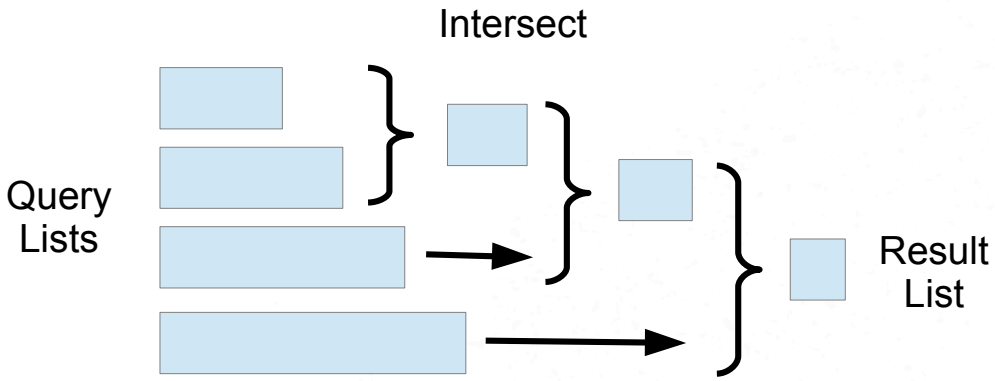
- Inverted index using document level postings lists.

List Intersection



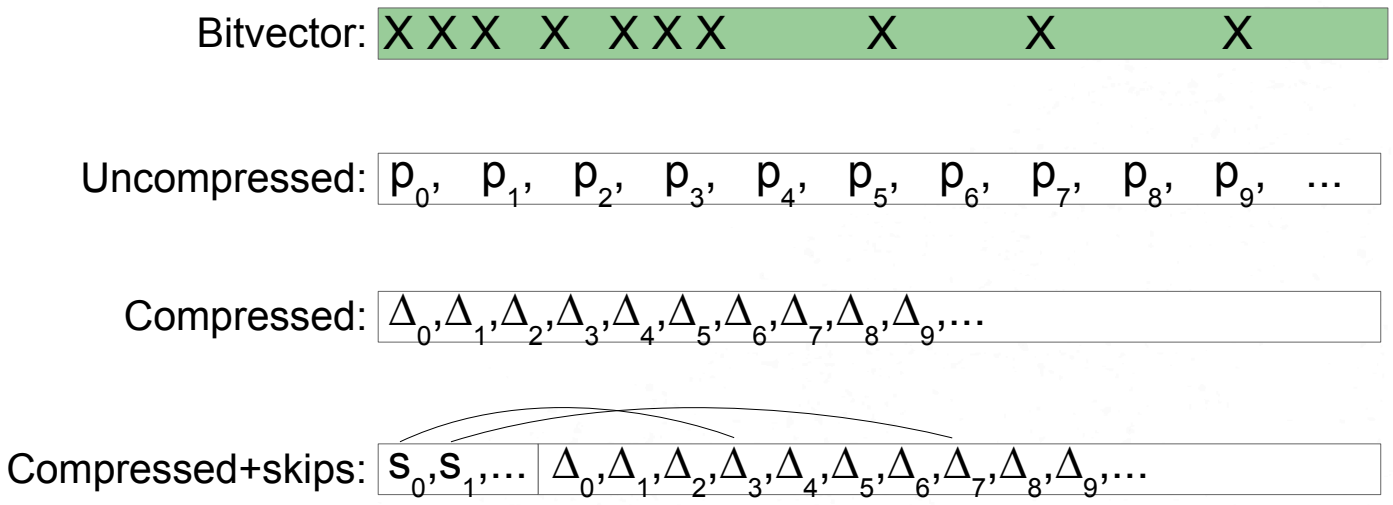
- To intersect, find the query term lists.

List Intersection



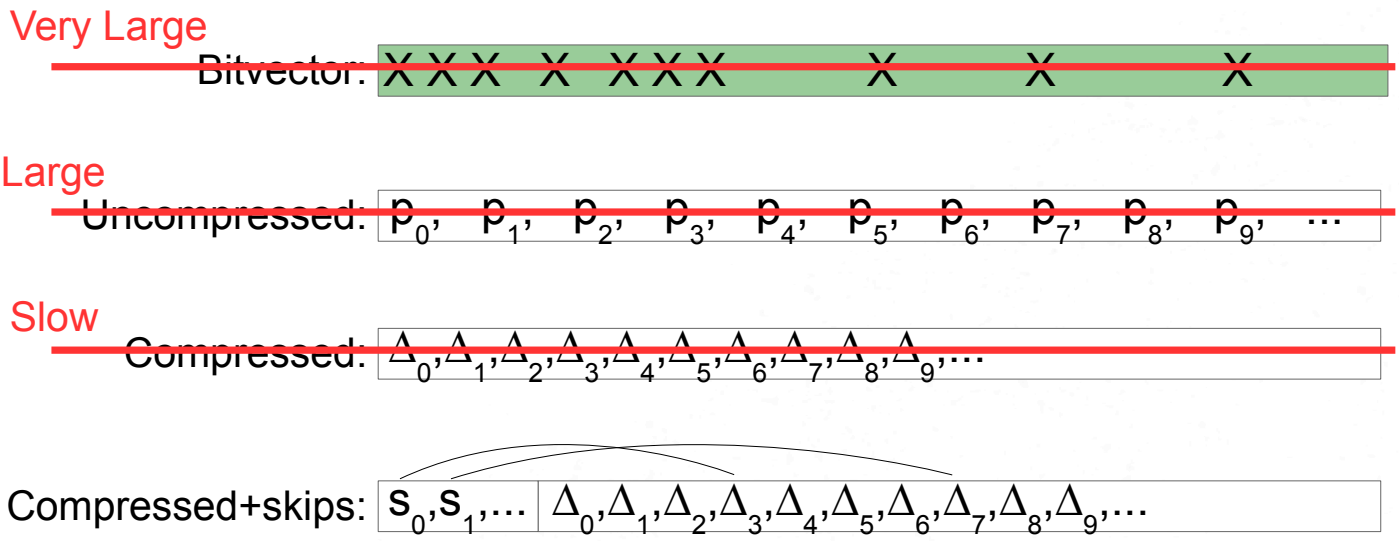
- Execute list intersection (small to large).
  - We run pairwise (term-at-a-time) processing of conjunctive-AND with lists ordered by document ID (allows merge).

List Encoding



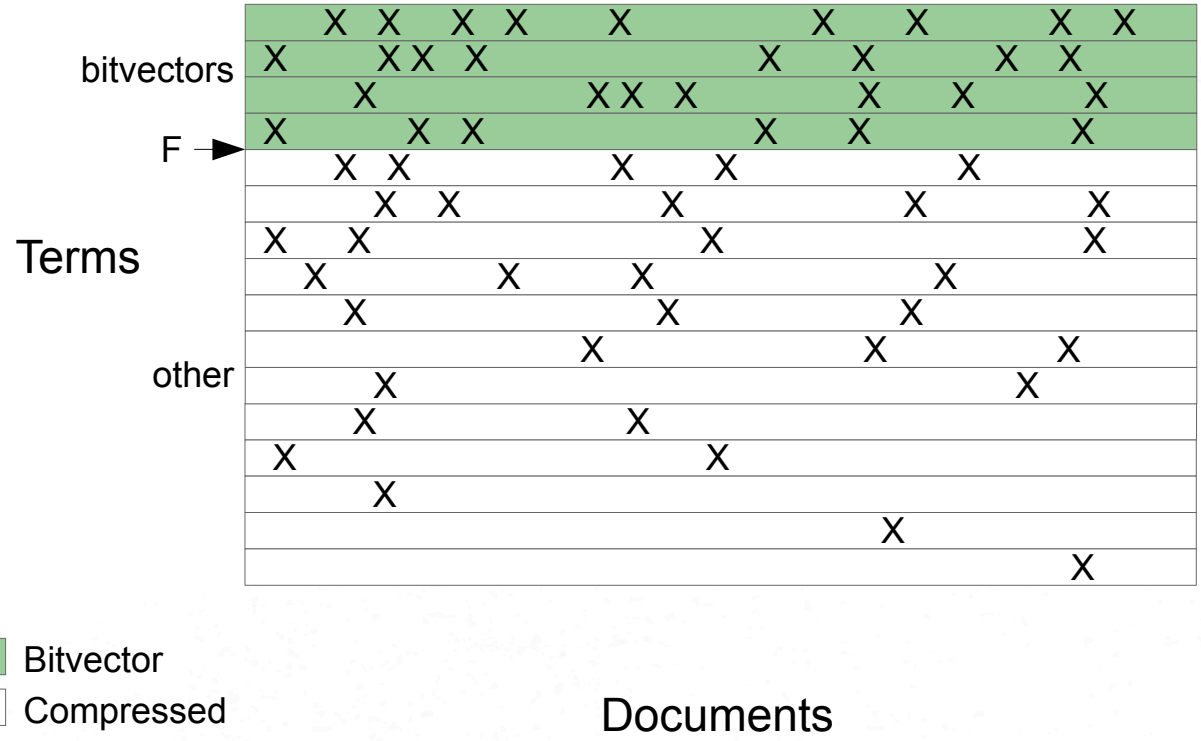
- Bitvectors are fast.
- Uncompressed lists are fast, but waste probes.
- Compressed lists have no random access.

List Encoding



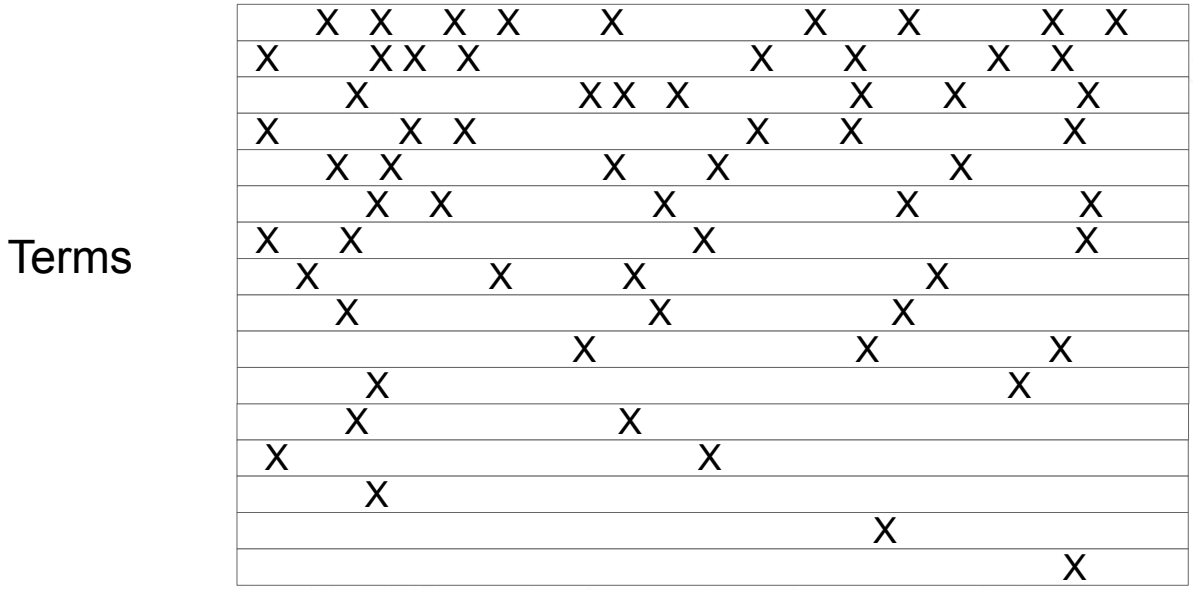
- Can also use a hybrid bitvector approach:
  - Use bitvectors if freq. > F and compressed lists for others [Culpepper and Moffat 2010].

List Encoding



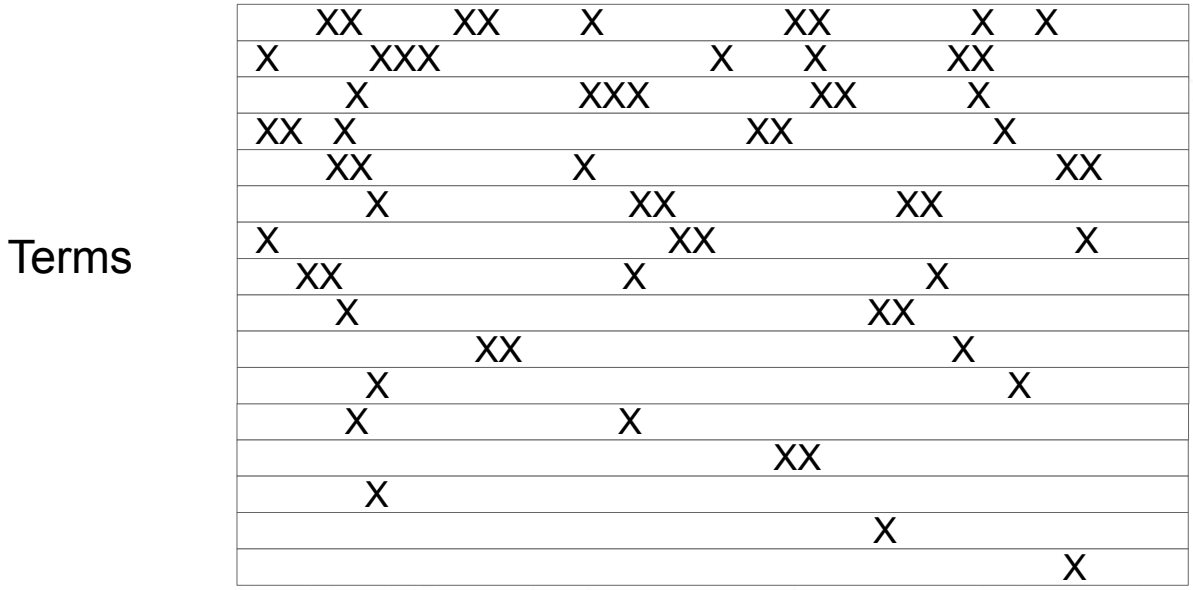
- Add (large overlaid) skips to compressed lists [new]

Document Ordering (orig)



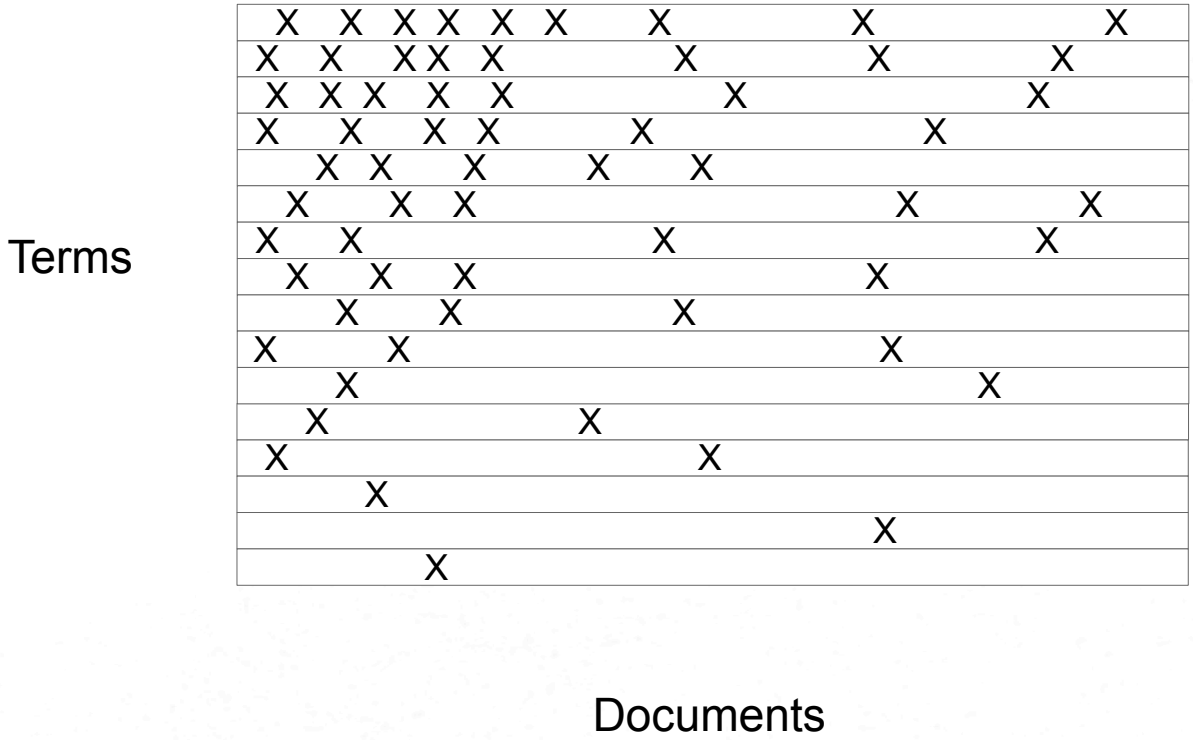
- Renumber the documents (columns) for improvements.

Document Ordering (url)



- URL ordering (≈ similarity) gives tight clustering.

Document Ordering (td)

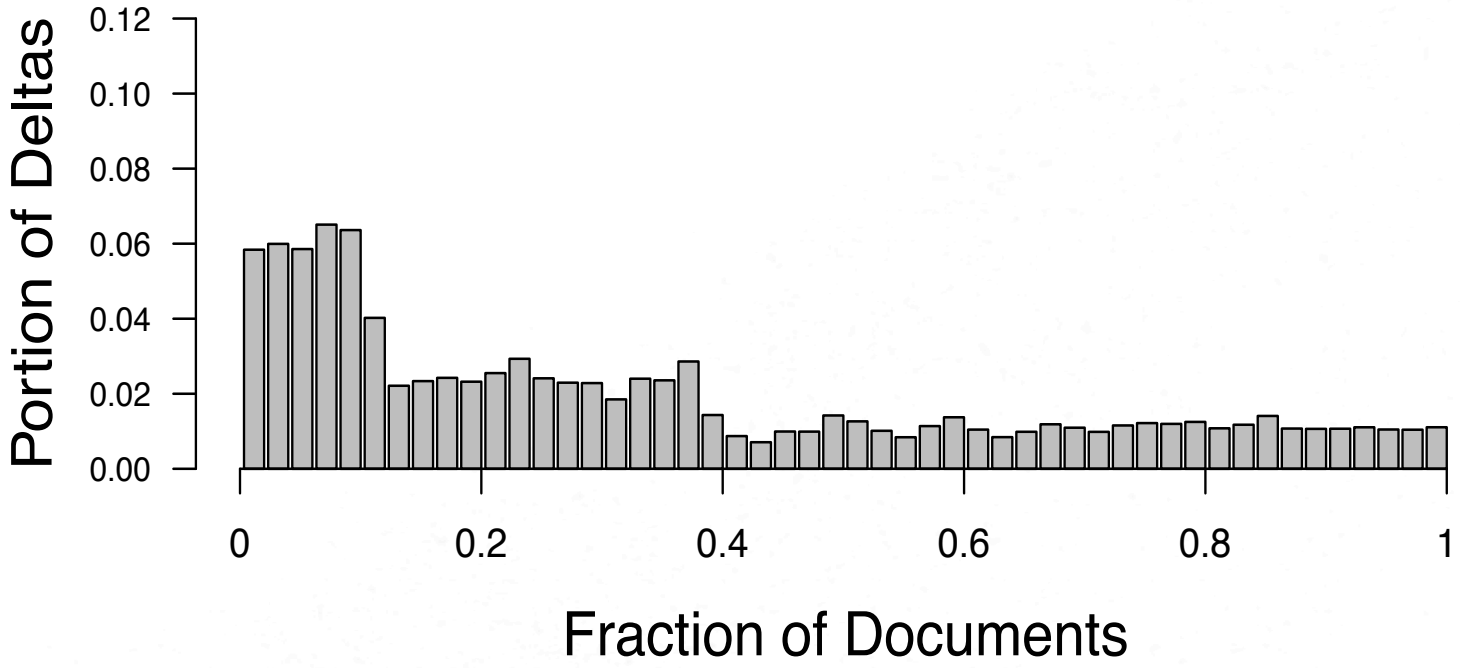


- Terms-in-document (td) ordering gives skewed clustering.

Combine Using Groups [new]

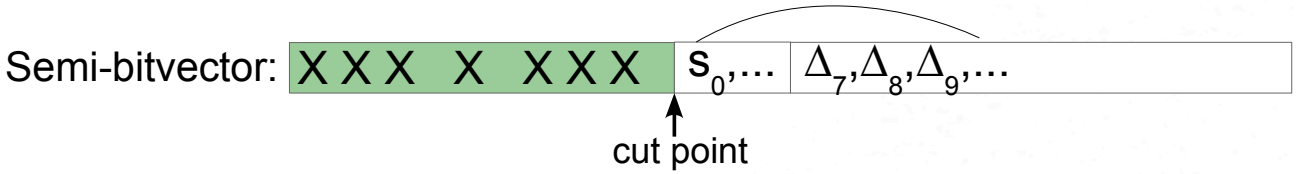
- Group documents by terms-in-document to get skewed clustering.
- Order within each group by URL to get tight clustering.
- Call this td-g#-url ordering.

Grouped Order



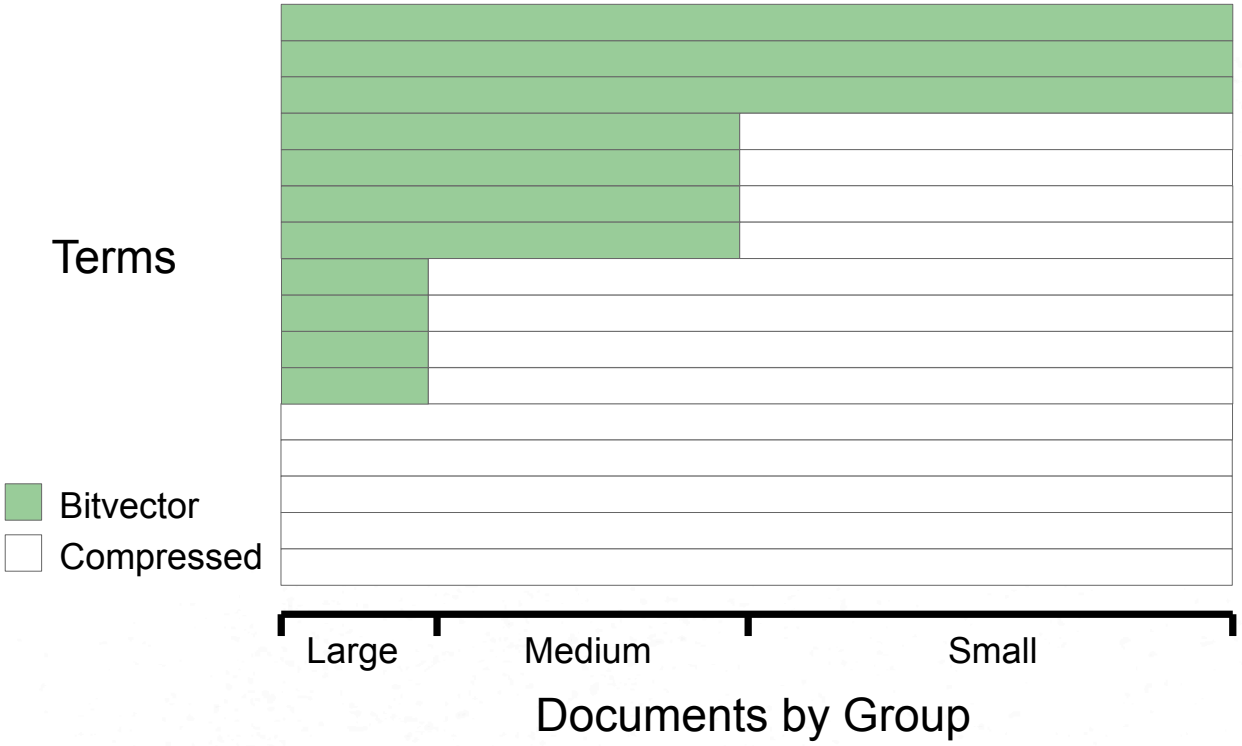
- Skew still remains in td-g3-url ordering.

Partial Bitvectors [new]



- Semi-bitvector data structure [new]:
  - Encode the front of a list as a bitvector.
  - Encode the remainder using skips and delta compression.
- Stores more postings in bitvectors (faster) for given space (so more efficient).

List Encoding

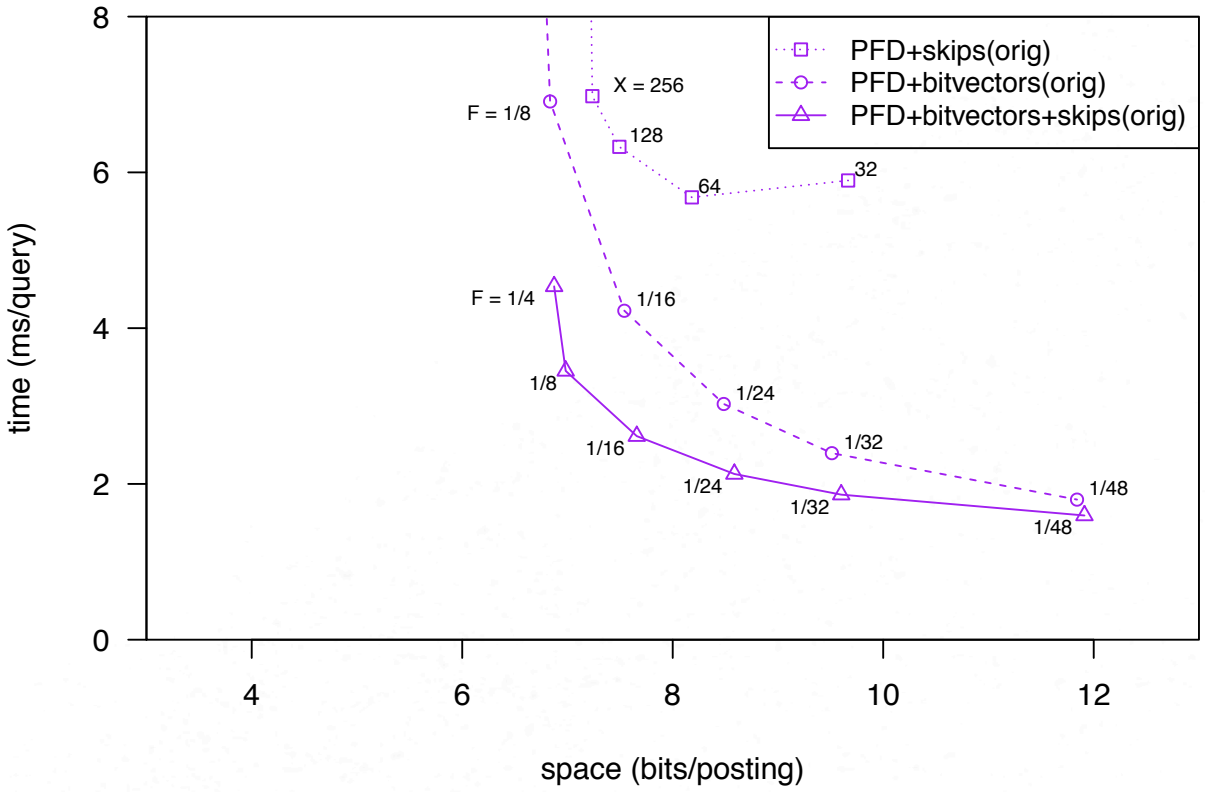


- Semi-bitvectors with grouping are fast and compact.

Experiments

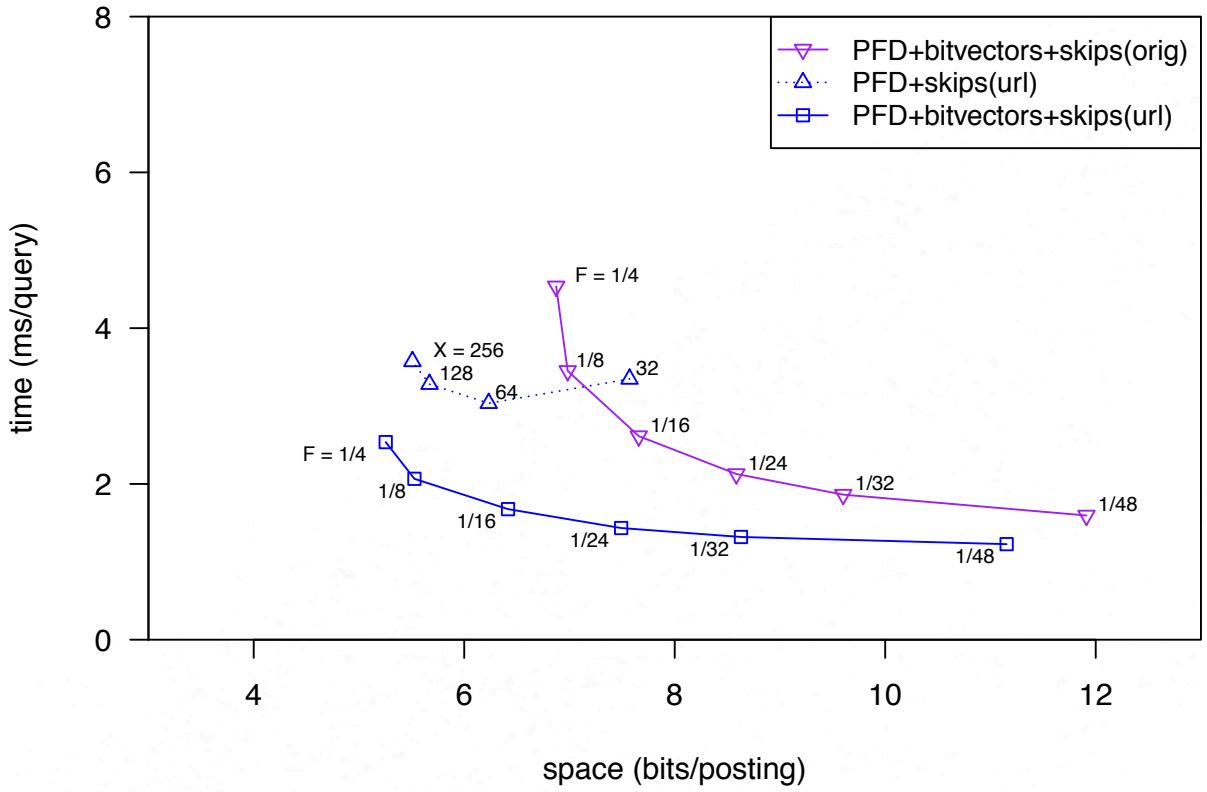
- Conjunctive-AND list intersection in memory.
- Space = encoded size of lists (no dictionary).
- Time = list intersection (no lookup in dictionary).
- Using GOV2 dataset (426GB) and 5000 corpus queries (4.1 terms per query).
  - Original order ≈ random order.

Experimental Results (orig)



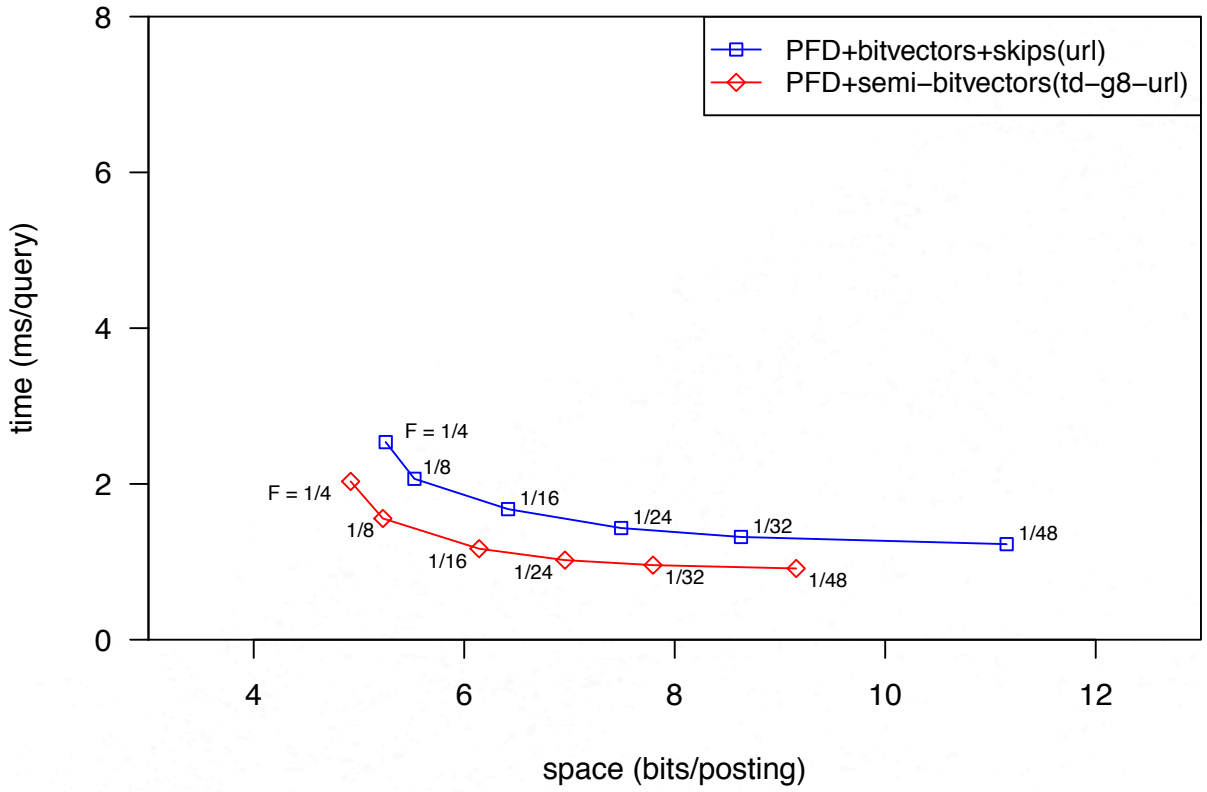
- Bitvectors are very fast and skips (X=256) help other lists.

Experimental Results (url)



- Bitvectors+skips still faster after reordering.

Experimental Results (td-g8-url)



- Improvements using semi-bitvectors with 8 groups.

Conclusions

- Skips can improve hybrid bitvector approaches.
- Grouping can combine skewed and tight clustering.
- Semi-bitvectors with grouping improves space and time:
  - More postings in bitvectors for given space.
  - Smaller deltas for compression.
  - Better locality of access and shared decoding.
- Future work:
  - Combine with ranking based systems.

Thank you.

Questions?

/* Comments */

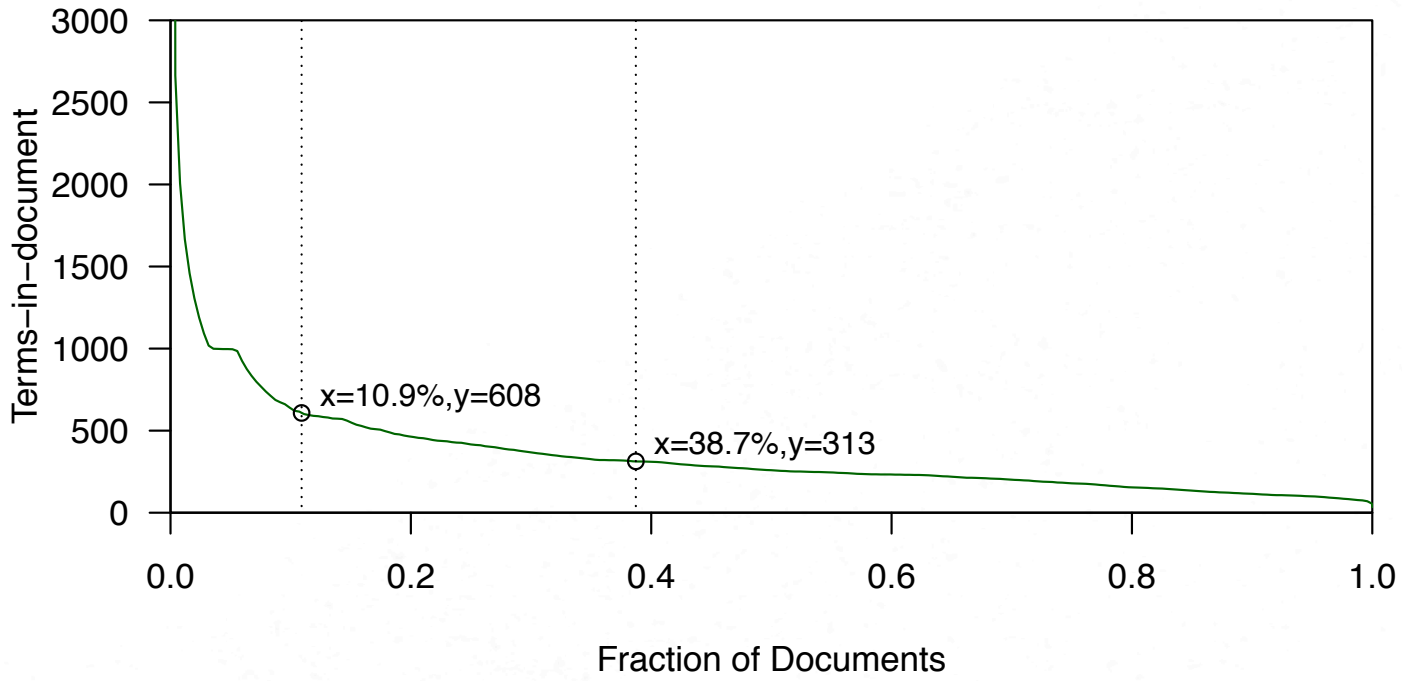
Andrew Kane: arkane@cs.uwaterloo.ca

(Student travel grant generously provided by SIGIR)

Existing Ordering Approaches

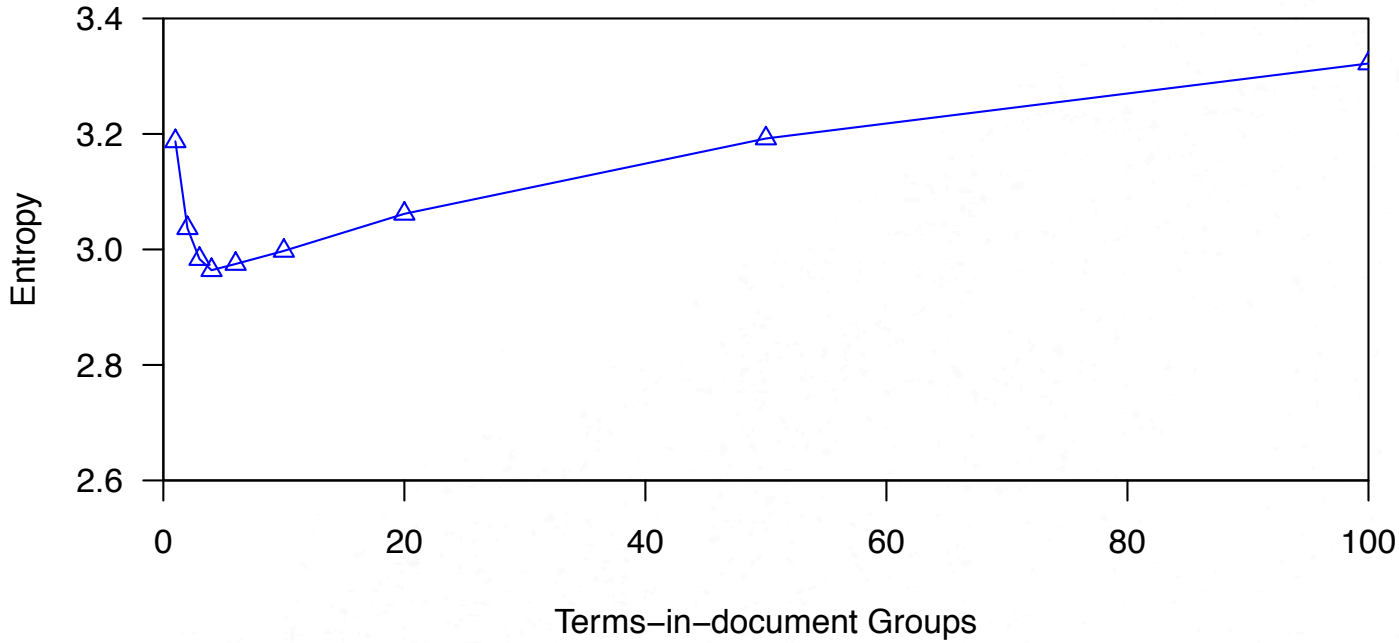
- Rank based – PageRank
- Matrix manipulation – block diagonal
- Document size
- Content similarity – similarity clustering, TSP
- Metadata – URL
- Hybrid – url.server.suffix-td-url

Terms-in-document Order



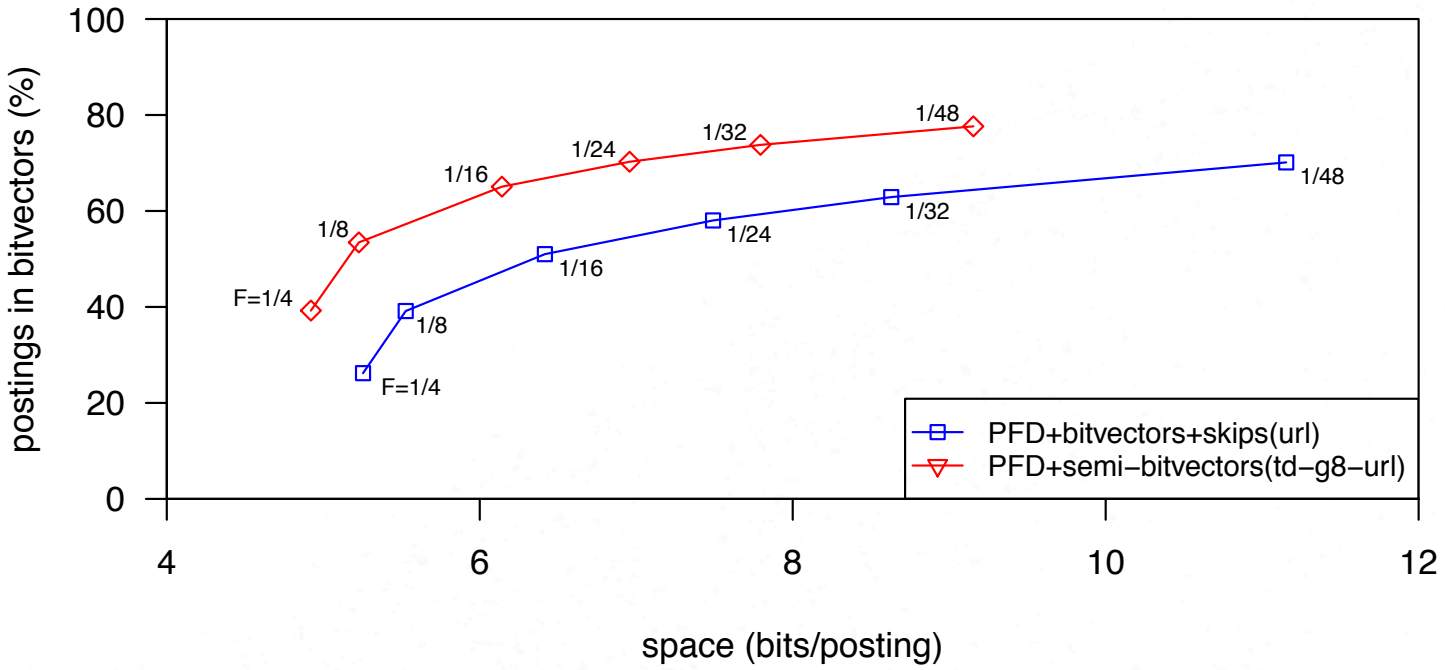
- Terms-in-document count for GOV2 dataset, split by number of postings into three groups, produces skew.
  - Area under the curve is equal for each partition.

Entropy for Grouping



- Grouping can make the data more compressible.
- Degradation is slow (entropy of td = 5.07)

Postings in Bitvectors



- Grouping and semi-bitvectors allows more postings to be stored in bitvectors

Ranking vs. Semi-bitvector

Algorithm/System	time (ms/q)	space (GB)
Lucene (vbyte)	26.0	42.1
Quasi-succinct indices (QS*)	11.9	36.9
Exhaustive AND	6.56	4.5
Hierarchical Block-Max (HIER 10G)	4.29	14.5
PFD+semi-bitvectors($\frac{1}{32}$, td-g8-url)	0.96	8.8

- Semi-bitvectors with conjunctive-AND are faster than these ranking based systems (from published results with similar hardware) on GOV2.

Semi-bitvectors in Ranking

- Pre-filter – run AND first then ranking.
- Sub-document pre-filter – AND over windows.
- High density filters – store dense regions as bitvectors, since low rank information.
- Query specific filter – dynamically pick terms to use semi-bitvectors.
- Guided processing – execute on subset of documents using AND to decide how to process query.