# Query-Driven Document Partitioning and Collection Selection

Diego Puppin, Fabrizio Silvestri and Domenico Laforenza

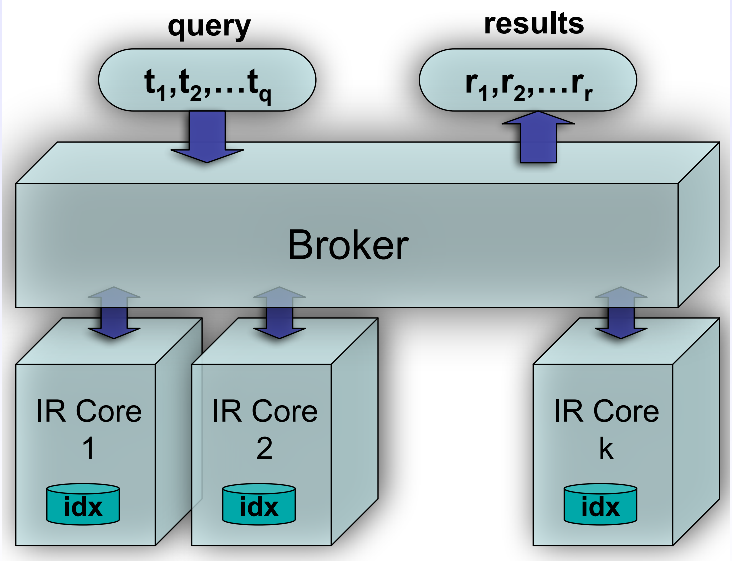
Institute for Information Science and Technologies
ISTI-CNR
Pisa, Italy

May 31, 2006

# Outline

# Distributed Search Engines

- The Web is growing larger and we need to manage more pages
- Replicated/Distributed Search Engines are a way to tackle
- Two main ways to partition the index
  - Document-partitioned
  - Term-partitioned
- Sometimes with different goals
  - Load-balancing
  - Throughput
  - Load-reduction


query
t1,t2,...tq
results
r1,r2,...rr
Broker
IR Core 1
idx
IR Core 2
idx
IR Core k
idx

# Term-partitioned Index

- Terms are assigned to servers
- Queries are submitted to servers holding the relevant terms
- Only a subset of servers is queried
- Results from each server are intersected/merged and ranked
- Problem of load-balancing, very hard to assign terms
  - Some recent works about this
- Can reduce the overall system load

# Document-partitioned Index

- Documents are assigned to servers
- A query can be submitted to each cluster, to improve throughput
- ... OR ... to reduce load, only to selected servers
- We must choose the “good servers” in advance
- Problem of partitioning and collection selection
- Back to the problems of heterogeneous collections (CORI etc.)

# Several Approaches to Partitioning and Selection

Document partitioning:

- Document clustering with k-means
- Semantic clustering with directories
- Random/round robin

Collection Selection:

- CORI
- Random
- All collections are queried
- Online sampling

Now, we are trying something new!

# Outline

# Two Birds with One Stone

- We are trying to make clusters of documents that answer to similar query
- We are also trying to clusters queries that recall similar documents
- We have to co-cluster [Dhillon 2003] the query-document matrix
- Very fast algorithm (much faster than k-means)

# Coclustering Example

$$p(X,Y) = \begin{bmatrix} .05 & .05 & .05 & 0 & 0 & 0 \\ .05 & .05 & .05 & 0 & 0 & 0 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ .04 & .04 & 0 & .04 & .04 & .04 \\ .04 & .04 & .04 & 0 & .04 & .04 \end{bmatrix}$$

$$p(\hat{X},\hat{Y}) = \begin{bmatrix} .3 & 0 \\ 0 & .3 \\ .2 & .2 \end{bmatrix}$$

Rows and columns are shuffled to minimize loss of information.

# Our Approach

- For every training query, we store the first 100 results of a reference search engine (centralized index)
- We create a query-document matrix, entries proportional to rank
- We co-cluster to put 1's and 0's together (actually, float numbers)
- We create $N$ document clusters and $M$ query clusters
- The process minimizes the loss of information between the original and the clustered matrix
- $\widehat{P}(qc_a, dc_b) = \sum_{i \in qc_b} \sum_{j \in dc_a} r_{ij}$

# Query-vector Representation

For each query, we store the Top-100 results with rank

| Query/Doc | d1 | d2 | d3 | d4 | d5 | d6 | ... | dn |
|---|---|---|---|---|---|---|---|---|
| q1 | - | 0.5 | 0.8 | 0.4 | - | 0.1 | ... | - |
| q2 | 0.3 | - | 0.2 | - | - | - | ... | 0.1 |
| q3 | - | - | - | - | - | - | ... | - |
| q4 | - | 0.4 | - | 0.2 | - | 0.5 | ... | 0.3 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| qm | 0.1 | 0.5 | 0.8 | - | - | - | ... | - |

We may have empty columns (documents never recalled, d5) and empty rows (queries with no results, q3). They are removed before co-clustering. About 52% of documents are recalled by NO query - we can put them in an *overflow* cluster.

# Collection Selection using PCAP

- We create big *query dictionaries* by chaining together all the queries from one query-cluster
- We index the dictionaries as documents
- For a new query $q$, we choose the best query-clusters with TF.IDF
  - For each query-cluster $qc_i$, we get a rank $r_q(qc_i)$
- We can compute the rank of each document-cluster:

$$r_q(dc_j) = \sum_i r_q(qc_i) \times \widehat{P}(i,j)$$

- The overflow IR core is always queried as the last one

# PCAP Example

| | dc1 | dc2 | dc3 | dc4 | dc5 | Rank for q |
|---|---|---|---|---|---|---|
| qc1 | | 0.5 | 0.8 | 0.1 | | 0.2 |
| qc2 | 0.3 | | 0.2 | | 0.1 | 0.8 |
| qc3 | 0.1 | 0.5 | 0.8 | | | 0 |

Query $q$ ranks the $qc$ respectively 0.2, 0.8 and 0.

$$
\begin{aligned}
r_q(dc_1) &= 0 \times 0.2 + 0.3 \times 0.8 + 0.1 \times 0 = 0.24 \\
r_q(dc_2) &= 0.5 \times 0.2 + 0 + 0 = 0.10 \\
r_q(dc_3) &= 0.8 \times 0.2 + 0.2 \times 0.8 + 0 = 0.32 \\
r_q(dc_4) &= 0.1 \times 0.2 + 0 + 0 = 0.02 \\
r_q(dc_5) &= 0 + 0.1 \times 0.8 + 0 = 0.08
\end{aligned}
$$

Clusters will be chosen in the order dc3, dc1, dc2, dc5, dc4.

# Outline

# Data Statistics

| | | |
|---|---|---|
| $dc$: | no. of document clusters | 16 + 1 |
| $qc$: | no. of query clusters | 128 |
| $d$: | no. of documents | 5,939,061 |
| | total size | 22 GB |
| $t$: | no. of unique terms | 2,700,000 |
| $t'$: | no. of unique terms in the query dictionary | 74,767 |
| $tq$: | no. of unique queries in the training set | 190,057 |
| $q1$: | no. of queries in the first test set | 194,200 |
| $q2$: | no. of queries in the second test set | 189,848 |
| $ed$: | empty (not recalled) documents | 3,128,366 |

**Table:** Statistics about collection representation. Data and query-logs from WBR99.

# Benchmarks

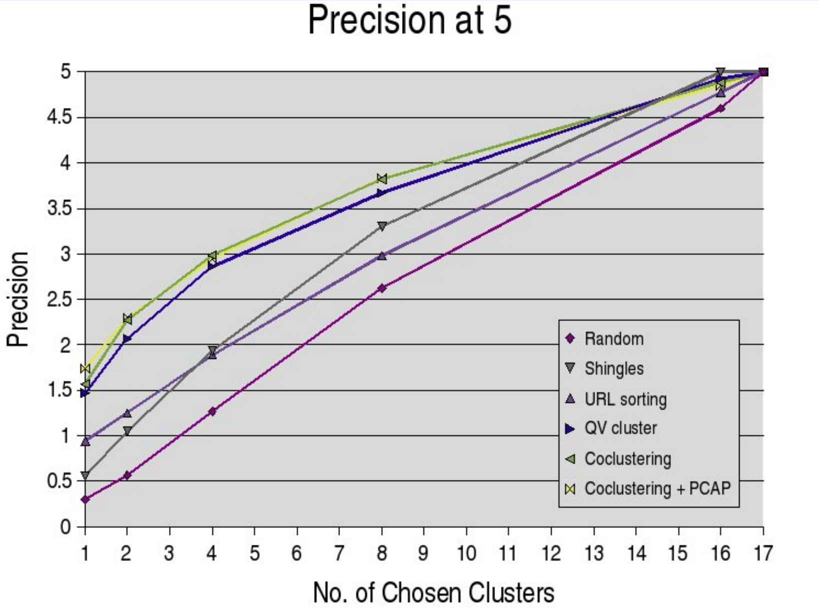
Partitions based on document contents:

- Random allocation
- Clusters with shingles UNPUBLISHED!!!
  - Signature of 64 permutations
- URL sorting UNPUBLISHED!!!

Partitions based on query-vector representation:

- Clustering with k-means UNPUBLISHED!!!
- Co-clustering (*)

(*) We could use PCAP in this case!


Precision at 5
Precision
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
No. of Chosen Clusters
Random
Shingles
URL sorting
QV cluster
Coclustering
Coclustering + PCAP

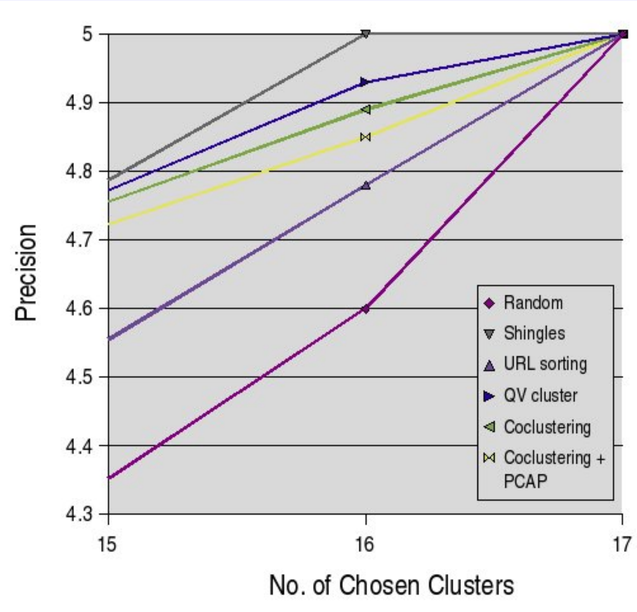
# Precision with one cluster

| | |
|---|---|
| random allocation (CORI) | 0.3 |
| clustering with shingles (CORI) | 0.56 |
| URL sorting (CORI) | 0.94 |
| clustering with k-means on query-vectors (CORI) | 1.47 |
| co-clustering (CORI) | 1.57 |
| co-clustering (PCAP) | 1.74 |

**Table:** Precision at 5 on the first cluster.

# Impact

- If a given precision is expected, we can use FEWER servers
- With a given number of servers, we get HIGHER precision
  - Confirmed with different metrics
- Smaller load for the IR system, with better results
- *No load balancing (for now)*
- 50% of pages contribute to 97% precision
  - We can remove the rest



Precision
5
4.9
4.8
4.7
4.6
4.5
4.4
4.3
15
16
17
No. of Chosen Clusters
Random
Shingles
URL sorting
QV cluster
Coclustering
Coclustering + PCAP

# Robustness to Topic Drift

Results do not change significantly if we do our test with later queries.

FOURTH WEEK

| Precision at | 1 | 2 | 4 | 8 | 16 | 17 |
|---|---|---|---|---|---|---|
| 5 | **1.74** | 2.30 | 2.95 | 3.83 | 4.85 | 5.00 |
| 10 | **3.45** | 4.57 | 5.84 | 7.60 | 9.67 | 10.00 |
| 20 | **6.93** | 9.17 | 11.68 | 15.15 | 19.31 | 20.00 |

FIFTH WEEK

| Precision at | 1 | 2 | 4 | 8 | 16 | 17 |
|---|---|---|---|---|---|---|
| 5 | **1.73** | 2.26 | 2.89 | 3.76 | 4.84 | 5.00 |
| 10 | **3.47** | 4.51 | 5.75 | 7.50 | 9.66 | 10.00 |
| 20 | **6.92** | 9.02 | 11.47 | 14.98 | 19.29 | 20.00 |

**Table:** Precision at 5 of the PCAP strategy, on the 4th and the 5th week.

# Representation Footprint

CORI representation includes:

- $df_{i,k}$, the number of documents in collection $i$ containing term $k$, which is $O(dc \times t)$ (before compression),
- $cw_i$, the number of different terms in collection $i$, $O(dc)$,
- $cf_k$, the number of resources containing the term $k$, $O(t)$.

Total: $O(dc \times t) + O(dc) + O(t)$ (before compression)

$dc$, number of document clusters (16+1)
$t$, number of distinct terms, 2,700,000

# Representation Footprint (2)

The PCAP representation is composed of:

- the PCAP matrix, with the computed $\widehat{p}$, which is $O(dc \times qc)$,
- the index for the query clusters, which can be seen as $n_{i,k}$, the number of occurences of term $k$ in the query cluster $i$, for each term occurring in the queries — $O(qc \times t')$.

TOTAL: $O(dc \times qc) + O(t' \times qc)$ = 9.4M (uncompressed)
CORI: $O(dc \times t) + O(dc) + O(t)$ = 48.6M (uncompressed)

$dc$, number of document clusters, 16+1
$qc$, number of query clusters, 128
$t'$, number of distinct terms in the query dictionary, 74,767
$t$, number of distinct terms, 2,700,000

# Outline

1. Introduction
2. The Query-vector Model
3. Experiments, With Exciting Unpublished Data!
4. **Conclusions**

# Main Contributions

- New (smaller) document representation as query-vectors
  - 2.7 M terms vs. 190 K queries
  - More effective on clustering (k-means)
  - Helps with the curse of dimensionality
- New partitioning strategy based on co-clustering
  - Very quick running time
- New (smaller) collection representation based on PCAP matrix
  - About 19% in size before compression
- New strategy PCAP for collection selection
  - 10% better than CORI on different metrics
- Removal of 50% of rarely-asked-for documents with minimal loss
  - They contribute only to 3% of recalled documents

# Next Steps

We would like to:

- include click-through data in the reference engine and precision evaluation;
  - ...if you have them, please share :-)...
- address load-balancing and overall system performance;
- complete a deeper analysis of the query-vector representation for IR tasks;
- compare of document- and term-partitioning.

# Acknowledgments


- MIUR CNR Strategic Project L 499/97-2000 (5%)
- NextGrid
- CoreGRID
- ISTI-CNR