Boosting Performance of Web Search Engines using Query Logs

Fabrizio Silvestri - ISTI CNR (Pisa)

Introduction

- Traditionally Web Search Engines (WSEs) keep track of requests from users.
- Query logs are a valuable source of information for multiple purposes:
  - Caching (*traditional use*).
  - Optimization of WSEs operations and data structures.
  - Marketing strategies.

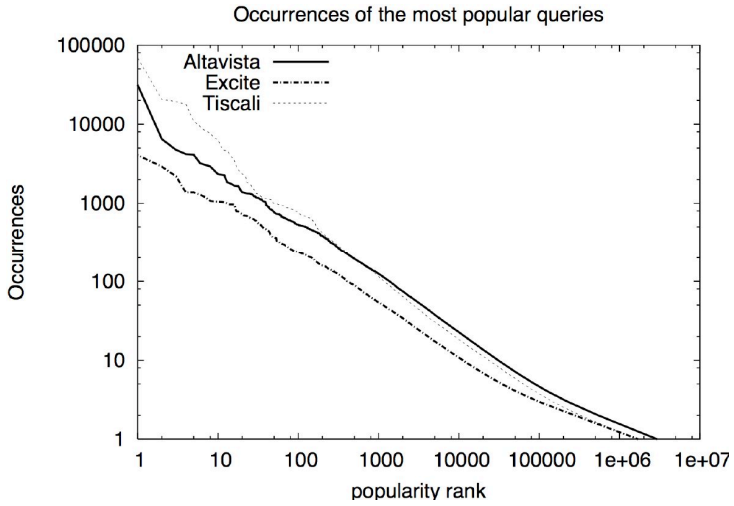
What is a Query Log

- It usually contains:
  - The query topic.
  - The requested page of results (e.g. the second page).
  - The user ID which the request come from.
  - The timestamp of the request.

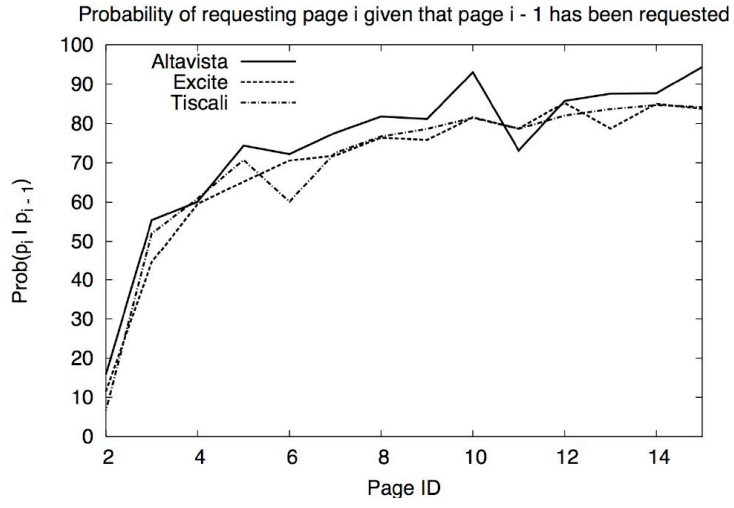
Our Logs

Query Log	Queries	Distinct Queries	Date
Excite	2,475,684	1,598,908	September 16th, 1997
Tiscali	3,278,211	1,538,934	April 2002
Alta Vista	7,175,648	2,657,410	Summer of 2001

What Does it Contain?



What Does it Contain?



How We Exploited Query Logs?

- Caching Policies
- Document Partitioning and Query Routing
- Term Partitioning Enhancement.

Caching Policies

- Traditionally caching policies are purely online.
- No statistics are used to enhance performance.
- Enhancing performance does not only mean enhancing hit-ratio.
  - Throughput is important too :-)

Probability Driven Caching

- R. Lempel, S. Moran. "*Predictive Caching and Prefetching of Query Results in Search Engines*". WWW2003. ACM Press, Budapest, May 2003. Pages 19-28.
- A statistic is drawn from a query log for prioritizing query log evictions and even insertions.
- Really really good hit-ratio results but...
  - O(log k) amortized complexity :-(

Static Dynamic Caching

- T. Fagni, S. Orlando, R. Perego, F. Silvestri. "*Boosting the Performance of Web Search Engines: Caching and Prefetching Query Results by Exploiting Historical Usage Data*". ACM Transactions on Information Systems, Vol. 24, No. 1, January 2006, Pages 1–28.
- Main ideas:
  - Zipf's law should be exploited directly (even PDC did)
  - Use Zipf's law even to enhance performance of the cache!

SDC Sections

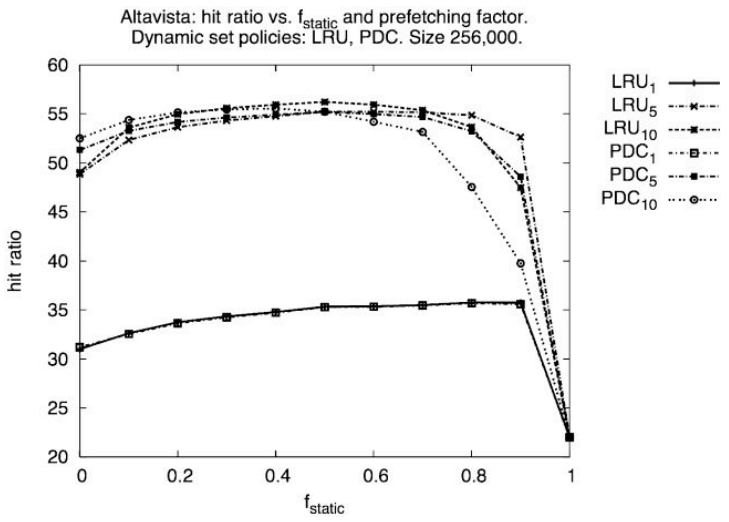
- Static Section
  - Filled-up with the most frequent queries.
  - Read-only portion never changed online.
  - First section to be queried when searching for previous results.
- Dynamic Section
  - Traditional cache.
  - Contains those queries which not fitted into the Static Section

SDC and Prefetching

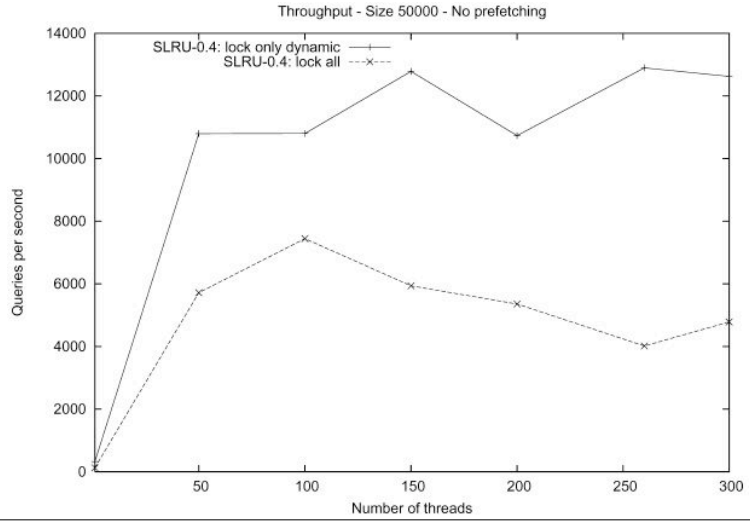
- Anticipate the requests from users by prefetching results from the back end.
- Using the following rules:

n = requested page_no	Hit/Miss	Cache Action
$n = 1$	Hit	Return page 1
	Miss	Submit query *(keywords, 1, 2)*
$n = 2$	Hit	Return page 2 and submit query *(keywords, 3, K)*
	Miss	Submit query *(keywords, 2, K)*
$n > 2$	Hit	Return page n
	Miss	Submit query *(keywords, n, K)*

Hit Ratio



Throughput



Collection Partitioning and Selection

- D. Puppin, F. Silvestri, D. Laforenza. "*Query-Driven Document Partitioning and Collection Selection*". Infoscale 2006, May 2006, Invited Paper, to Appear.
- Idea:
  - use the query log to drive the assignment of documents, with the goal of putting together the documents that answer a given query.
  - Co-clustering of queries and document id of the documents answering those queries.

Innovations

- **query-vector model**: documents are represented by the queries that they answer.
- collections are represented by the co-clustering results.
- new partitioning strategy
- new selection strategy
- identification of rarely asked documents on the basis of the information of the query log.
  - More than 50% of documents were never hit by any query on our log.

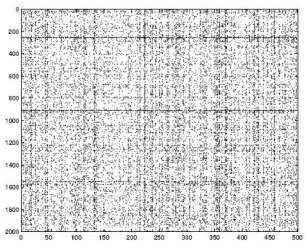
Contingency Matrix

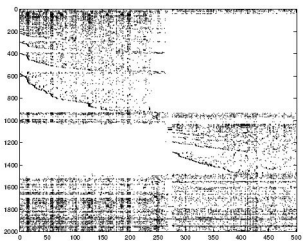
- Co-clustering works by manipulating a contingency matrix.
- Our contingency matrix is the matrix *Queries* X *Documents*.
- Each entry is given by $\Upsilon_{ij} = r_{ij} / \sum_{i \in D} \sum_{j \in \Phi} r_{ij}$

Co-clustering Results

- The result of co-clustering is the following matrix:
- qc's are the query clusters.
- dc's are the document clusters.

$$\widehat{P}(qc_a, dc_b) = \sum_{i \in qc_b} \sum_{j \in dc_a} r_{ij}$$





Experiments

- WBR99 collection: 5,939,061 documents, 2,700,000 different terms. Query log from January 2003 to October 2003.
- For each query we considered the top 50 results returned by a centralized search engine as the most relevant document for that query.
- Zettair as the engine for carrying out tests.

Precision

- CORI on Random Allocation

Precision at	1	2	4	8	16	17
5	0.3	0.57	1.27	2.62	4.6	5
10	0.59	1.16	2.55	5.0	9.3	10
20	1.20	2.49	5.04	9.77	18.71	20

Precision

- CORI vs. PCAP. Queries from the 4th week.

CORI Precision at	1	2	4	8	16	17
5	1.57	2.27	2.99	3.82	4.89	5.00
10	3.06	4.46	5.89	7.56	9.77	10.00
20	6.01	8.78	11.64	15.00	19.52	20.00

PCAP 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

Better Term Partitioning

- C. Lucchese, S. Orlando, R. Perego, and F. Silvestri. *Statistics Driven Term Partitioning to Enhance Performance of Parallel IR Systems*. Ongoing Work.
- Main idea:
  - Organize the partitioned vocabulary in Term Partitioned IRSs.
  - Exploit the co-occurrence of terms in queries from query log.

Why Term Partitioning?

- A. Moffat, W. Webber, J. Zobel, and R. Baeza-Yates. *A pipelined architecture for distributed text query evaluation*. Manuscript submitted for publication.

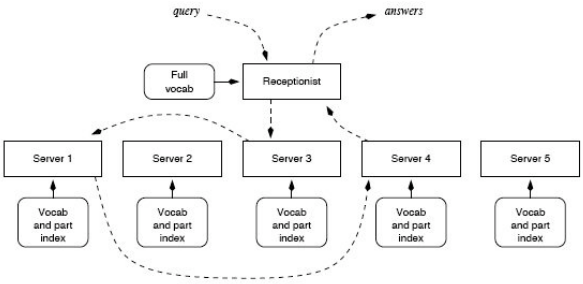


Figure 2: Pipelined query processing with a term-partitioned index. In this example, the query contains terms that necessitate routing the query bundle through processors 3, 1, and 4 in a system containing five servers and five index partitions.

Encouraging Figures...

Statistic	Method	
	Document	Pipelined
Sectors read ($\times 10^6$)	6.61	2.60
Distinct reads ($\times 10^4$)	10.27	1.46
I/O wait seconds ($\times 10^1$)	13.85	3.28

Table 10: Read behavior of document-partitioned and pipelined distributed systems, $k = 8$ nodes, and `TB/01`*. Figures are totaled across all nodes. Disk sectors are 512 bytes. Numbers reflect physical disk activity only, and requests handled from in-memory buffers are not included in the totals.*

Discouraging Results

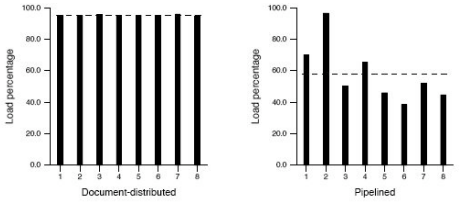


Figure 3: Average per-processor busy load for $k = 8$ and TB/01*, for document-distributed processing and pipelined processing. The dashed line in each graph is the average busy load over the eight processors.*

These positive results cannot, of course, be claimed as improvements, since they involve an atypical query stream. But they do offer encouragement, in that if a way of balancing the processing requirements of the pipelined scheme can be found, it should be possible to fully crystallize the benefits generated by its inherent advantages in terms of disk operations. Options that we plan to explore in future work include dynamic reassignment of lists while the query stream is being processed, to adjust for term usage patterns in the query stream; and selective list replication.

Term Partitioning Enhancements

- The method is aimed to assign terms to servers, by exploiting the knowledge about frequencies of occurrence and co-occurrence of terms in queries submitted in the past.
- Optimize:

DEFINITION 8. *[α-**weight**] Given a query stream Φ, a lexicon $\mathcal{T}$, and a p-partitioning function λ, we define α-weight on Φ according to λ as:*

$$\Omega_\lambda(\Phi) = \alpha \cdot \frac{\overline{\omega}_\lambda(\Phi)}{N_\omega} + (1-\alpha) \cdot \frac{\widehat{L}_\lambda(\Phi)}{N_L} \qquad (6)$$

where α, $0 \leq \alpha \leq 1$, is a parameter that can be used to tune our α-weight, while N_ω and N_L are constants aimed to normalize *the two factors, so that $0 \leq \overline{\omega}_\lambda(\Phi)/N_\omega \leq 1$ and $0 \leq \widehat{L}_\lambda(\Phi)/N_L \leq 1$. Note that, in both cases 0 (1) is the best (worst) value.*

Training and Test Sets

- We optimized the alfa-omega measure on a training set by using frequent itemset mining approach.
- We evaluate the assignment induced by it on a test set.

Table 1: Main characteristics of the query logs used.

Query log	queries	terms	query len.	date
TodoBR	22,589,568	959,833	3.433	2001
Excite	2,477,283	419,603	3.364	Sep. 16^{th} 1997
AltaVista	7,175,648	895,792	2.507	Summer 2001

Servers per Query

Servers	baseline cases random	baseline cases bin packing	term assignment $\alpha = 0.9$
$\Phi_{test} = TodoBR$			
1	28	28	50
2	31	30	20
3	17	17	14
> 3	24	25	16
$\Phi_{test} = Excite$			
1	22	22	33
2	33	33	34
3	22	21	19
> 3	23	25	14
$\Phi_{test} = AltaVista$			
1	29	29	41
2	39	39	38
3	21	21	16
> 3	11	11	5

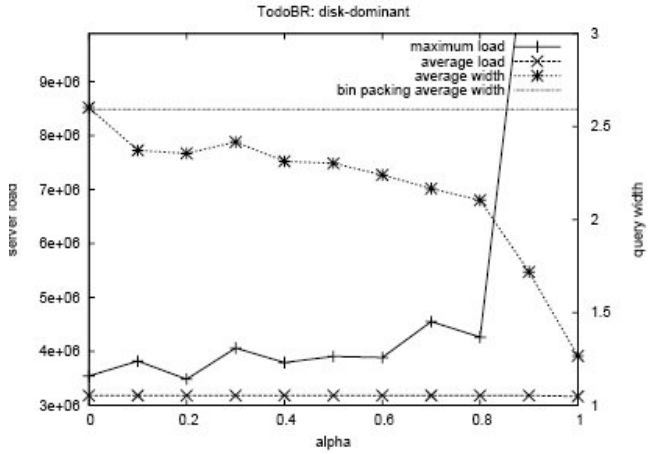
List Replication

- To balance load and reduce the server usage we selectively replicate the most frequent terms.

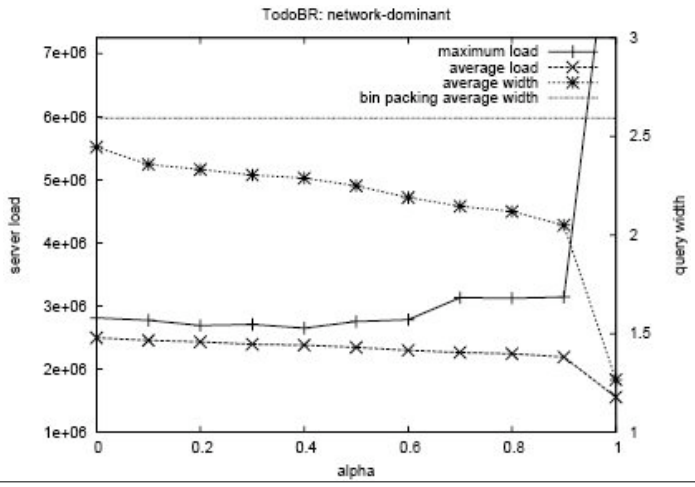
Query per Servers - Replicated Lists

	replication Factors					
	0.0001		0.0005		0.001	
Servers	*bin pack.*	*term ass.*	*bin pack.*	*term ass.*	*bin pack.*	*term ass.*
$\Phi_{test} = TodoBR$						
1	42	54	56	62	63	67
2	31	22	22	18	19	16
3	12	10	9	8	8	8
> 3	15	14	12	11	10	9
$\Phi_{test} = Excite$						
1	30	40	40	48	46	53
2	39	38	38	36	36	33
3	20	15	16	12	13	10
> 3	11	6	7	4	5	3
$\Phi_{test} = AltaVista$						
1	36	45	45	54	50	60
2	42	37	39	34	37	30
3	16	13	12	10	10	8
> 3	5	4	3	3	3	2

Load Balancing (No Buffer Cache)



Load Balancing - Network is Dominant



Evaluating the *Topic Shift* Effect

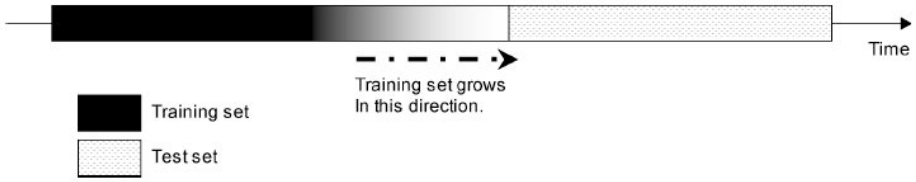


Fig. 9. Schema of the training process for the static set: the training set grows on the right, thus reducing the size of the test set.

Various Training Times

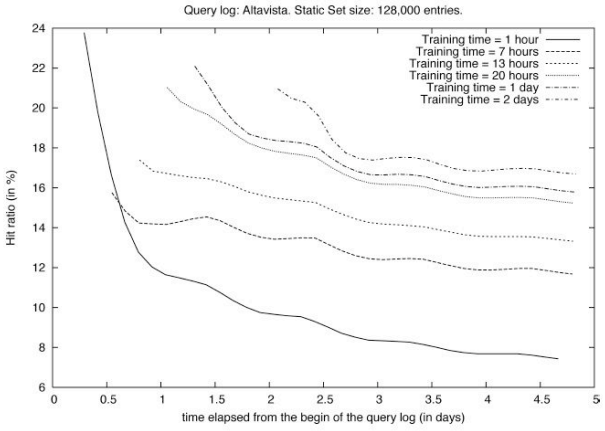


Fig. 10. Static set hit ratio on the *Alta Vista* query log as a function of time. The size of the static set has been set to 128,000 elements, while the training time was varied between 1 h and 2 days.

Thank You!

- Fabrizio Silvestri - ISTI (CNR)
  Pisa - Italy
- **fabrizio.silvestri@isti.cnr.it**
- **http://hpc.isti.cnr.it/~silvestr**