KMV-Peer: A Robust and Adaptive Peer-Selection Algorithm

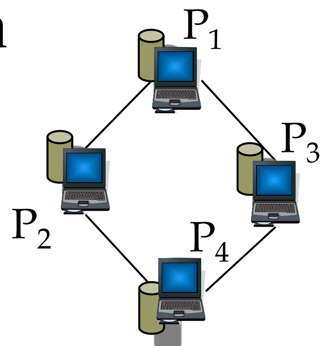
Yosi Mass, Yehoshua Sagiv, Michal Shmueli-Scheuer

IBM Haifa Research Lab

Hebrew University of Jerusalem

Motivation and Problem Statement

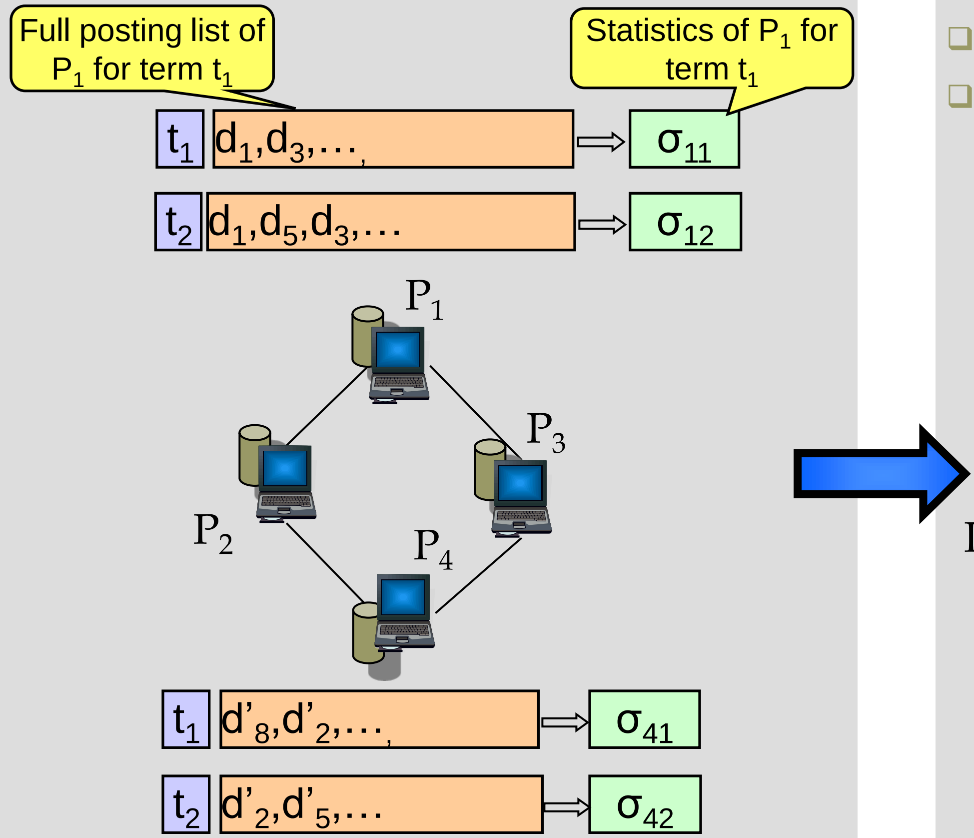
- **Motivation**
  - Scale up Indexing and retrieval of large data collections
  - Solution is described in the context of cooperative peers, each has its own collection



- **Problem Statement**
  - Find a good approximation of a centralized system for answering conjunctive multi-term queries, while keeping at a minimum both the number of peers that are contacted and the communication cost

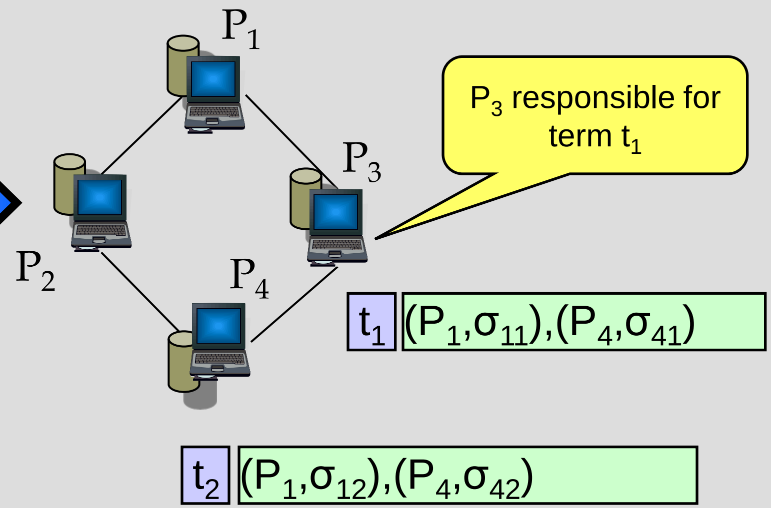
Solution Framework - Indexing

Create small-size per-term local statistics



Make all statistics globally available

- Use DHT to assign terms to peers
- A peer that is responsible for a term has the statistics of all other peers for that term



Our Contributions

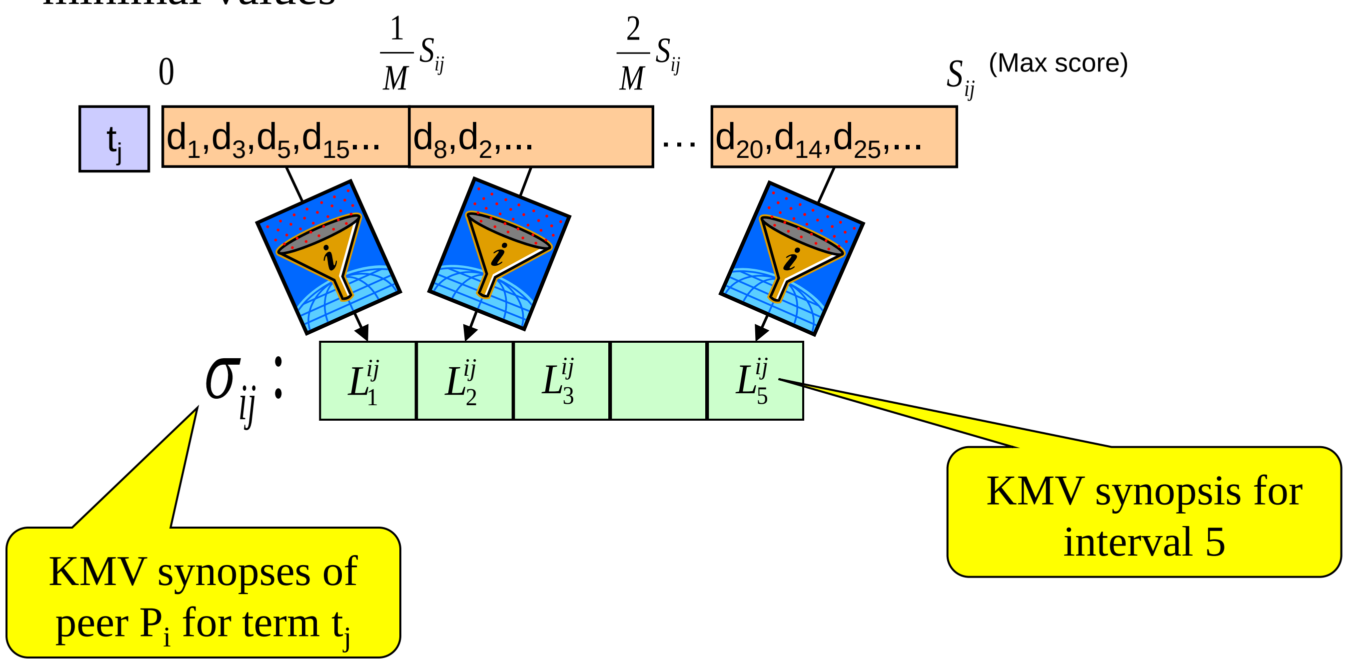
- A novel per-term statistics based on KMV (Beyer et el. 2007) synopses and histograms
- A peer-selection algorithm that exploits the above statistics
- An improvement of the state-of-the-art by a factor of four

Agenda

- Collection statistics
- Peer-selection algorithm
- Experiments
- Summary and Future Work

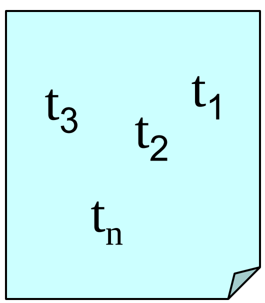
Per-term KMV Statistics

- Keep posting list for each term t_j, sorted by increasing score for $q=(t_j)$
- Divide the documents into *M* equi-width score intervals
- Apply a uniform hash function to the doc ids in each interval and take the *l* minimal values

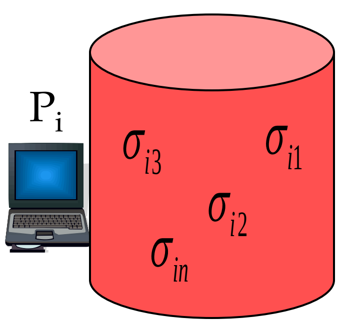


Peer-Scoring Functions

- Given a query $q=(t_1,\ldots,t_n)$ and the statistics of peer P_i for the query terms, use the histograms to estimate the score of a virtual document that belongs to P_i.



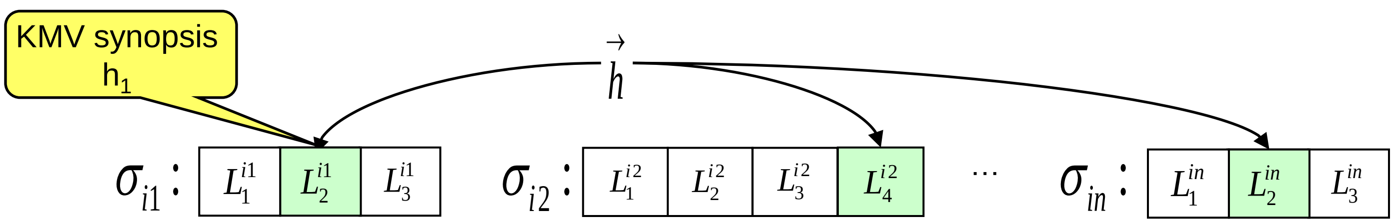
$$score_q(d) = g_{aggr}(score_{t_1}(d),\ldots,score_{t_n}(d))$$



$$score_q(p_i) = F\,?(\sigma_{i1},\ldots,\sigma_{in})$$

Peer-Scoring Functions - contd

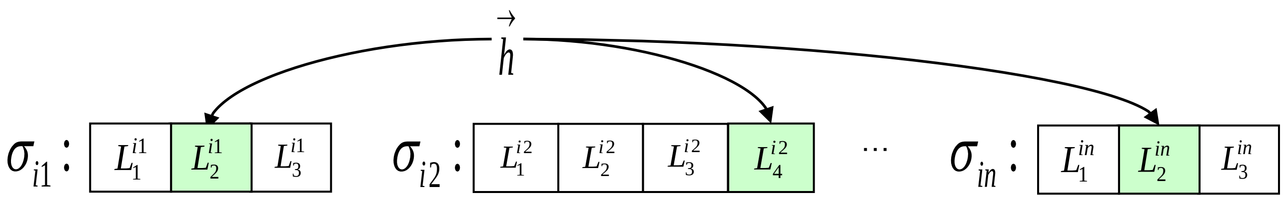
- Consider the set $C = \{\vec{h} = (h_1, \ldots, h_n) \mid h_j \in \sigma_{ij}\}$ namely all combinations of one KMV synopsis for each query term.
- The score associated with a KMV synopsis h_j, denoted by $mid(h_j)$, is the middle of the interval that corresponds to that synopsis



$$score_q(d) = g_{aggr}(score_{t_1}(d), \ldots, score_{t_n}(d))$$

$$score(\vec{h}) = g_{aggr}(mid(h_1), \ldots, mid(h_n))$$

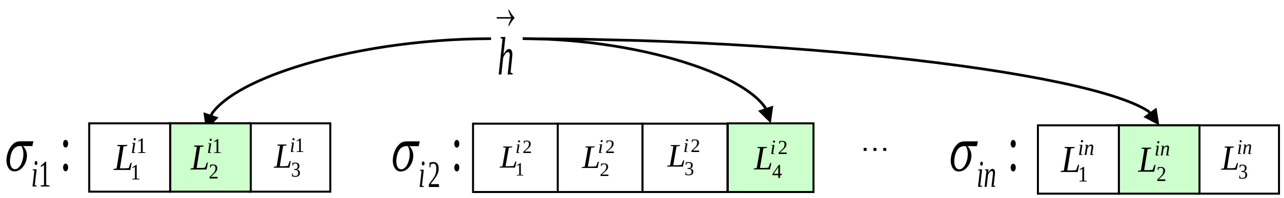
KMV-int: The Peer Intersection Score



- Non-emptiness estimator $\vec{h}_{\cap}$ is true if the intersection of $\{h_1,\ldots,h_n\}$ is not empty
- Intersection score - $score_q^{\cap}(p_i) = \max_{\vec{h} \in C \wedge \vec{h}_{\cap}} (score(\vec{h}))$
- If $\vec{h}_{\cap}$ is true, then we are guaranteed there is a document d with all query terms
- But $\vec{h}_{\cap}$ can be an underestimate (false negative) especially for queries with a large number of terms

KMV-exp: The Peer Expected Score

- Measures the expected relevance of the documents of P_i to the query q



$$score_q^E(p_i) = |D_i| \sum_{\vec{h} \in C} score(\vec{h}) \Pr(\vec{h})$$

$$\Pr(\vec{h}) = \prod_{j=1}^{n} \frac{e(h_j)}{|D_i|}$$

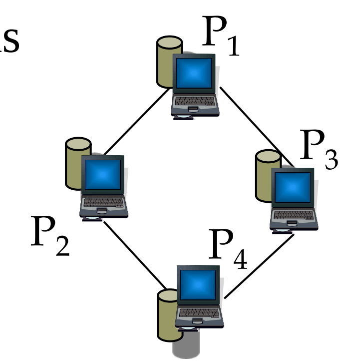
KMV size estimator for h_j

All docs in peer P_i

A Basic Peer-Selection Algorithm

- Input: $q=(t_1,\ldots,t_n)$, k (top-k results), K (max number of peers to contact)
- Locate the peers that are responsible for the query terms
- Get all their statistics

t_1	$(P_1,\sigma_{11}),(P_4,\sigma_{41})$
t_2	$(P_1,\sigma_{12}),(P_4,\sigma_{42})$
	…
t_n	$(P_1,\sigma_{1n}),(P_5,\sigma_{5n}),(P_9,\sigma_{9n})$



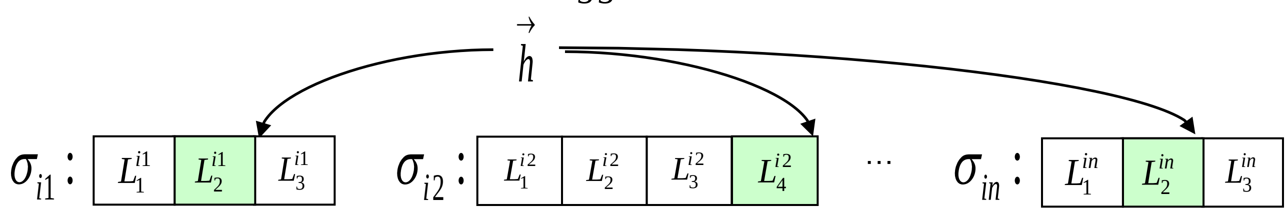
- Rank the peers using KMV-int and if less than K peers have non-empty intersection then rank the rest by KMV-exp
- Select the top-K peers and contact them to get their top-k results
- Merge the returned results and return the top-k

Algorithm Improvements – Save Communication Cost

- At the query initiating peer P_q :
  - Locate the two peers that are responsible for the terms with the smallest statistics. Call them P^{t_f} and P^{t_s}
  - Forward the query to peer P^{t_s}
- At peer P^{t_s} :
  - Get all statistics from peer P^{t_f}
  - Apply KMV-int on the peers in the two lists and obtain a set of candidate peers P
  - Get the rest of the statistics about q but only for peers in P

Algorithm Improvements – Adaptive Ranking

- Work in rounds
  - In each round contact the next best *k'* peers (*k'* < *K*)
  - Obtain a threshold score (*min-k*) which is the score of the last (i.e., *k-th*) document among the current top-k
  - Adaptively rank the remaindered peers
    - Define $high(\vec{h}) = g_{aggr}(high(h_1),\ldots,high(h_n))$



    - In the scoring functions (KMV-int and KMV-exp), ignore tuples whose $high(\vec{h}) < min\text{-}k$

KMV-Peer: The Peer-Selection Algorithm

k – top-k results are requested
k' – number of peers to contact in each iteration
K – max number of peers to contact

Algorithm 1 KMV-peer

Input: $q = \{t_1, \ldots, t_n\}, k, k', K$

locate $p^{t_1}, \ldots, p^{t_n}$ and get the sizes of their statistics;
let p^{t_f} and p^{t_s} have the two smallest statistics;
switch to p^{t_s};
get the statistics about t_f from p^{t_f};
$P \leftarrow$ all peers s.t. $score^{\cap}_{\bar{q}}(p) > 0$, where $\bar{q} = \{t_f, t_s\}$;
get the rest of the statistics about q for all $p \in P$;
$n \leftarrow 0$; $ct \leftarrow 0$; $res \leftarrow \emptyset$;
repeat
    $P_1 \leftarrow$ **get-next-real-peers**(P, k', ct);
    $res \leftarrow$ top-$k(P_1, res)$;
    $ct \leftarrow$ min-$k(res)$;
    remove from P all virtual peers $p_{(i,g)}$ s.t. $p_i \in P_1$;
    $n \leftarrow n + 1$;
until $(nk' \geq K) \vee (|P_1| < k')$;
return res

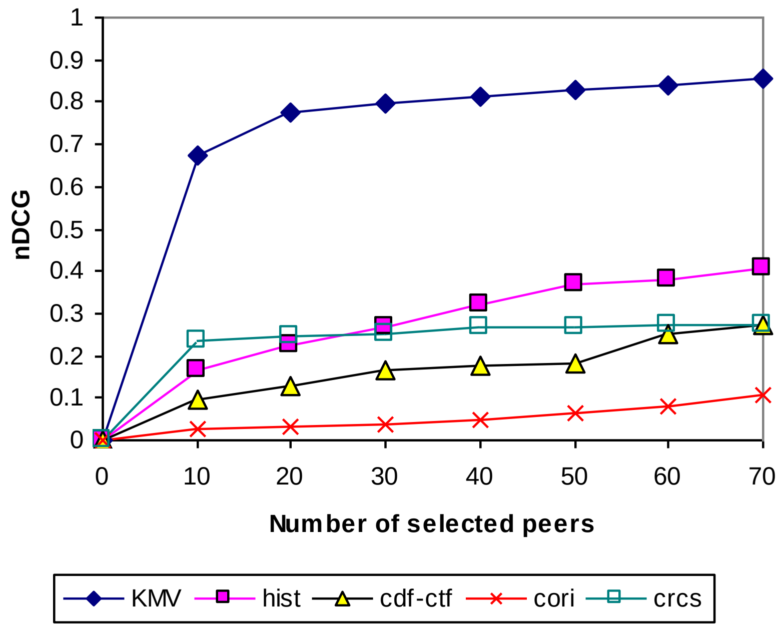
Score peers by **KMV-int,** but if less than k' peers have a non-zero score then use **KMV-exp**

Experimental Setting

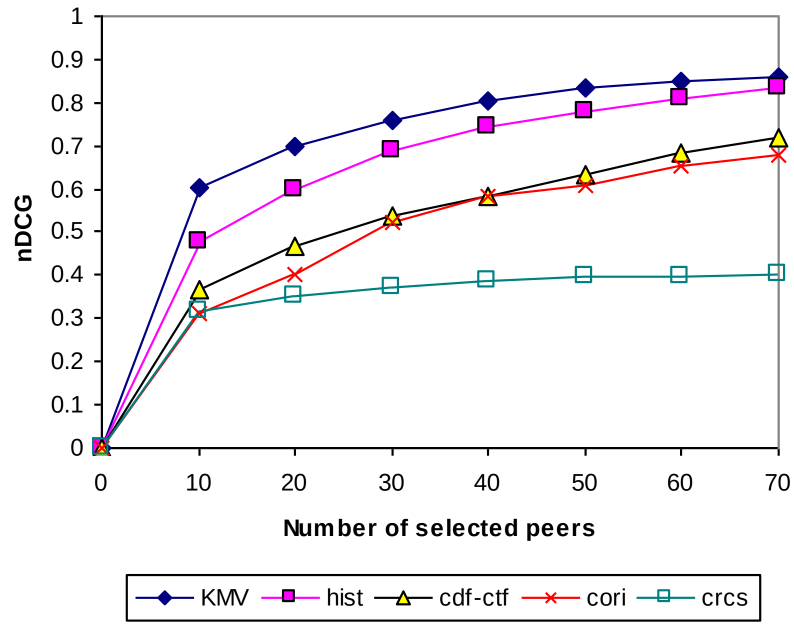
- **Datasets**
  - **Trec –** 10M web pages from Trec GOV2 collection
  - **Blog –** 2M Blog posts from Blogger.com
- **Setups**
  - **Trec-10K –** 10,000 peers, each having 1,000 documents
  - **Trec-1K –** 1,000 peers, each having 10,000 documents
  - **Blog –** 1,000 peers, each having 2,000 documents
- **Queries**
  - Trec – 15 queries from the topic-distillation track of the TREC 2003 Web Track benchmark
  - Blog – 75 queries from the blog track of TREC 2008
- **Parameters**
  - *l* (KMV size), *M* (num score intervals), *G* (num groups)
- **Evaluation**
  - Normalized DCG (nDCG), which considers the order of the results in the ground truth (i.e., a centralized system)
  - MAP

KMV-Peer Compared to State-of-the-Art

Trec-10K (l10,M5)



Blog (l10,M5)

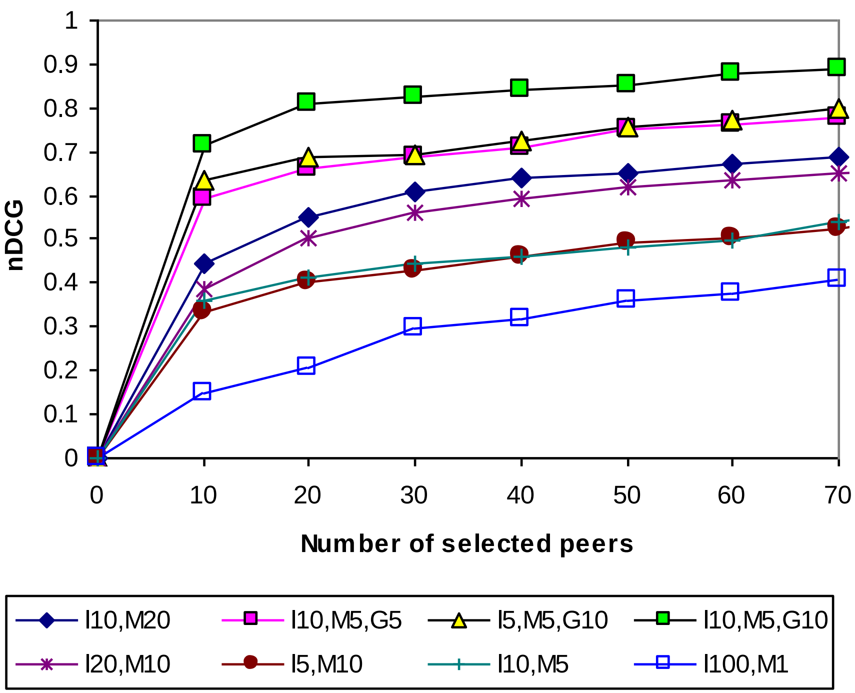


Communication cost (KBytes)

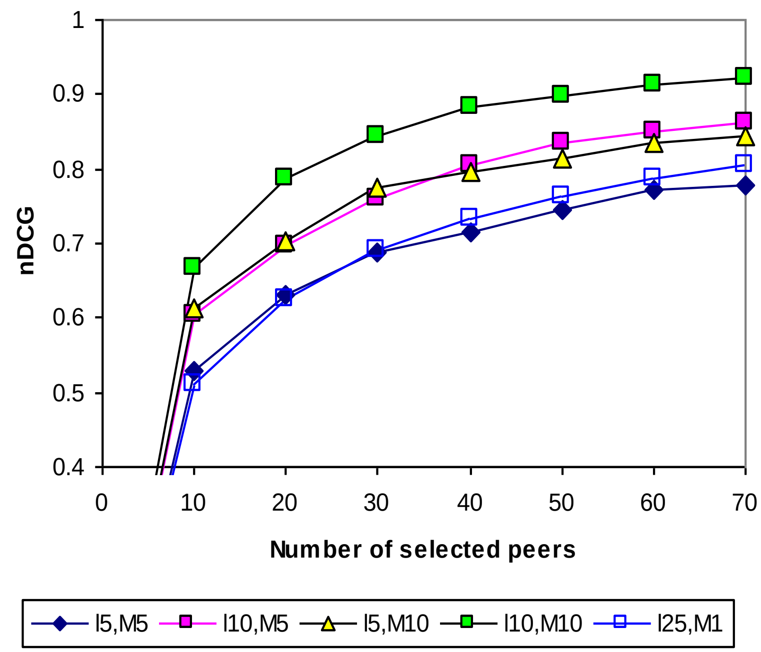
	KMV	hist	cdf-ctf/cori
Trec-10K	233	632	164
Trec-1K	198	151	23
Blog	53	110	24

Tuning The Parameters of KMV-Peer

Trec-1K

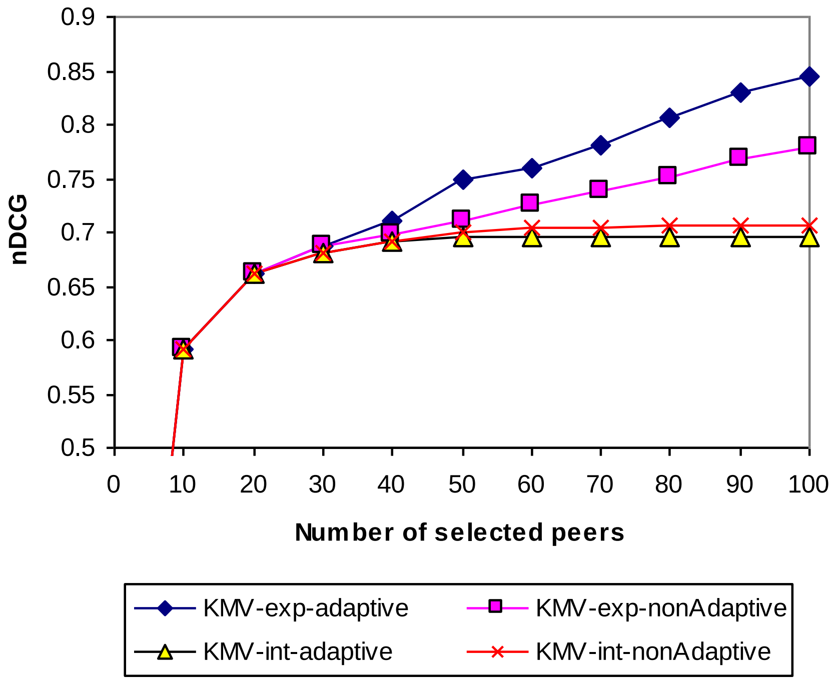


Blog

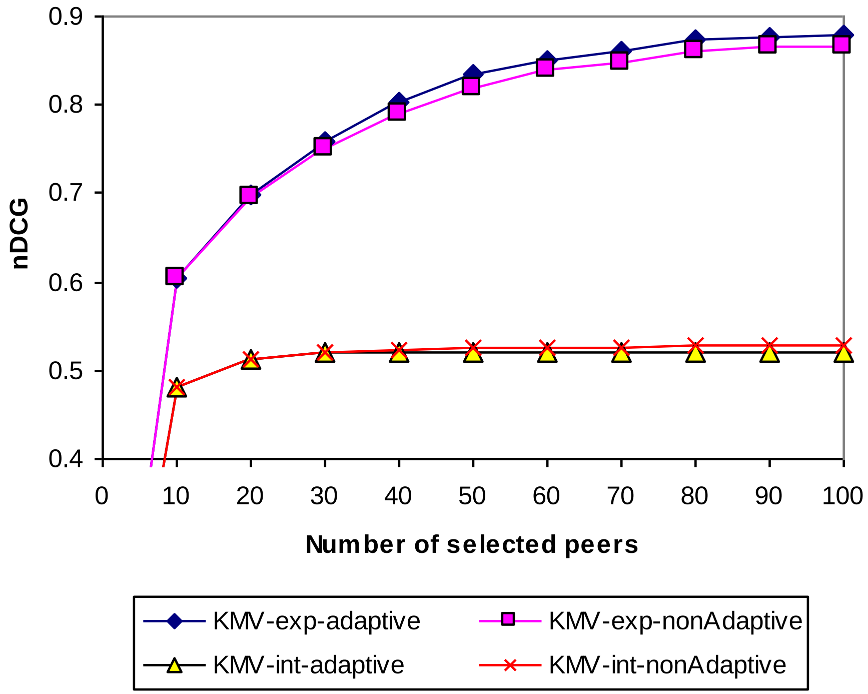


Testing Different Variants of KMV-Peer

Trec-1K



Blog



Testing Different Scoring Functions

nDCG at K=20

	score	KMV	hist	cdf-ctf	cori	crcs
Trec-10K	Lucene	**0.77**	0.22	0.12	0.03	0.24
	BM25	**0.81**	0.14	0.12	0.04	0.16
	Lucene*	**0.67**	0.22	0.11	0.03	0.21
Trec-1K	Lucene	**0.66**	0.21	0.12	0.09	0.29
	BM25	**0.69**	0.18	0.13	0.11	0.23
	Lucene*	**0.58**	0.17	0.12	0.09	0.20
Blog	Lucene	**0.69**	0.59	0.46	0.40	0.35
	BM25	**0.63**	0.52	0.51	0.40	0.31
	Lucene*	**0.62**	0.54	0.44	0.37	0.27

- Lucene – Apache Lucene score with global synchronization
- BM25 – Okapi BM25 score with global synchronization
- Lucene* – Lucene score with the parameters (e.g., idf) derived by each peer from its own collection

Conclusions

- We presented a fully decentralized peer-selection algorithm (KMV-peer) for approximating the results of a centralized search engine, while using only a small subset of the peers and controlling the communication cost.
- The algorithm employs two scoring functions for ranking peers. The first is the intersection score and is based on a non-emptiness estimator. The second is the expected score.
- KMV-peer outperforms the state-of-the-art methods and achieves an improvement of more than 400% over other methods
- Regarding communication-cost, we showed how to filter out peers in early stages of the algorithm, thereby saving the need to send their synopses.

Future Work

- Investigate further reductions in communication cost by using top-k algorithms with a stopping condition
- Consider less restrictive non-emptiness estimators (disjunctive queries)

Thank You!

Questions ?