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

Full posting list of \( P_1 \) for term \( t_1 \)

1. \( t_1; d_1, d_3, \ldots \)
2. \( t_2; d_1, d_5, d_3, \ldots \)

Statistics of \( P_1 \) for term \( t_1 \)

- \( \sigma_{11} \)

Statistics of \( P_1 \) for term \( t_1 \)

- \( \sigma_{12} \)

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

<table>
<thead>
<tr>
<th>Term</th>
<th>Responsible</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>( P_1 )</td>
<td>( (P_1, \sigma_{11}), (P_4, \sigma_{41}) )</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( P_4 )</td>
<td>( (P_1, \sigma_{12}), (P_4, \sigma_{42}) )</td>
</tr>
</tbody>
</table>
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

\[
\begin{align*}
\frac{1}{M} S_{ij} & \quad \frac{2}{M} S_{ij} \\
t_j & \quad d_1, d_3, d_5, d_{15}, \ldots \\
& \quad d_8, d_2, \ldots \\
& \quad \ldots \\
& \quad d_{20}, d_{14}, d_{25}, \ldots \\
\end{align*}
\]

\( S_{ij} \) (Max score)

\( K MV \) synopsis for interval 5

\( \sigma_{ij} \): KMV synopses of peer \( P_i \) for term \( t_j \)
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 \( P_i \).

\[
score_q(d) = g_{\text{aggr}}(score_{t_1}(d), \ldots, score_{t_n}(d))
\]

\[
score_q(p_i) = F(?)(\sigma_{i1}, \ldots, \sigma_{in})
\]
Peer-Scoring Functions - contd

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

$\text{score}_q(d) = g_{\text{aggr}}(\text{score}_{t_1}(d), \ldots, \text{score}_{t_n}(d))$

$\text{score}(h) = g_{\text{aggr}}(\text{mid}(h_1), \ldots, \text{mid}(h_n))$
KMV-int: The Peer Intersection Score

- Non-emptiness estimator $h_\cap$ is true if the intersection of $\{h_1, \ldots, h_n\}$ is not empty.

- Intersection score - $score_q(p_i) = \max_{h \in C \land h_\cap} (score(h))$

- If $h_\cap$ is true, then we are guaranteed there is a document d with all query terms.

- But $h_\cap$ 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 $q$

\[ \text{score}_q^E(p_i) = |D_i| \sum_{h \in C} \text{score}(h) \Pr(h) \]

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

KMV size estimator for $h_i$

All docs in peer $P_i$
A Basic Peer-Selection Algorithm

- Input: $q=(t_1,…,t_n)$, $k$ (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 (\( \text{min-}k \)) which is the score of the last (i.e., \( k-th \)) document among the current top-\( k \)
  - Adaptively rank the remaindered peers
    - Define \( \text{high}(h) = g_{aggr}(\text{high}(h_1),...,\text{high}(h_n)) \)

\[
\sigma_{i_1}: \begin{array}{llll}
L_1^i & L_2^i & L_3^i & \ldots \\
\end{array} \quad \sigma_{i_2}: \begin{array}{llll}
L_1^{i_2} & L_2^{i_2} & L_3^{i_2} & L_4^{i_2} \\
\end{array} \quad \ldots \quad \sigma_{i_n}: \begin{array}{llll}
L_1^{i_n} & L_2^{i_n} & L_3^{i_n} \\
\end{array}
\]

- In the scoring functions (K\( \text{MV-int} \) and K\( \text{MV-exp} \)), ignore tuples whose \( \text{high}(h) < \text{min-}k \)
KMV-Peer: The Peer-Selection Algorithm

Algorithm 1 KMV-peer

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

1: locate \( p^{t_1}, \ldots, p^{t_n} \) and get the sizes of their statistics;
2: let \( p^{t_f} \) and \( p^{t_s} \) have the two smallest statistics;
3: switch to \( p^{t_s} \);
4: get the statistics about \( t_f \) from \( p^{t_f} \);
5: \( P \leftarrow \) all peers s.t. \( \text{score}_{\bar{q}}(p) > 0 \), where \( \bar{q} = \{t_f, t_s\} \);
6: get the rest of the statistics about \( q \) for all \( p \in P \);
7: \( n \leftarrow 0; \; ct \leftarrow 0; \; \text{res} \leftarrow \emptyset \);
8: repeat
9: \( P_1 \leftarrow \text{get-next-real-peers}(P, k', ct) \);
10: \( \text{res} \leftarrow \text{top-k}(P_1, \text{res}) \);
11: \( ct \leftarrow \text{min-k}(\text{res}) \);
12: remove from \( P \) all virtual peers \( p_{(i, g)} \) s.t. \( p_i \in P_1 \);
13: \( n \leftarrow n + 1 \);
14: until \( (nk' \geq K) \lor (|P_1| < k') \);
15: return \( \text{res} \)

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

Score peers by \text{KMV-int}, but if less than \( k' \) peers have a non-zero score then use \text{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)

<table>
<thead>
<tr>
<th></th>
<th>KMV</th>
<th>hist</th>
<th>cdf-ctf/cori</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trec-10K</td>
<td>233</td>
<td>632</td>
<td>164</td>
</tr>
<tr>
<td>Trec-1K</td>
<td>198</td>
<td>151</td>
<td>23</td>
</tr>
<tr>
<td>Blog</td>
<td>53</td>
<td>110</td>
<td>24</td>
</tr>
</tbody>
</table>
Tuning The Parameters of KMV-Peer

Trec-1K

Blog

Wsdm’11, Feb 9 – 12, Hong Kong
Testing Different Variants of KMV-Peer

Trec-1K

Blog

Wsdm'11, Feb 9 – 12, Hong Kong
Testing Different Scoring Functions

- 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

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

nDCG at K=20
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?