Using an Inverted Index Synopsis for Query Latency and Performance Prediction

Nicola Tonellotto
University of Pisa
nicola.tonellotto@unipi.it

Joint work with Craig Macdonald
University of Glasgow
craig.macdonald@glasgow.ac.uk

To appear in
Who am I?

• MSc in Computer Engineering at University of Pisa (2002)
• PhD in Information Engineering at University of Dortmund & University of Pisa (2008)
• Researcher at ISTI-CNR from 2006 to 2019
• Assistant processor at UNIPI since 2019

Research topics
• High performance computing & Clouds
• Efficiency information retrieval & Web search
• Distributed computing & Big data platforms
• Machine learning efficiency
The scale of Web search challenge
How many documents? In how long?

- Reports suggest that Google considers a total of **30 trillion** pages in the indexes of its search engine.
  - Identifies **relevant results** from these 30 trillion **in 0.63 seconds**.
  - Clearly this a **big data** problem!
- To answer a user's query, a search engine doesn't read through all of those pages: the **index data structures** help it to efficiently find pages that effectively match the query and will help the user.
  - **Effective**: users want relevant search results.
  - **Efficient**: users aren't prepared to wait a long time for search results.
Search as a Distributed Problem

- To achieve efficiency at Big Data scale, search engines use many servers:

  - $N \& M$ can be very big:
    - Microsoft's Bing search engine has "hundreds of thousands of query servers"
Computing Platform

Source: https://www.pexels.com/photo/datacenter-server-449401/
If we know how long a query will take, can we reconfigure the search engines' ranking pipeline?

**BM25 + DAAT**
- 1,000 – 10,000 docs

**Learning To Rank**
- 10 – 100 docs

- Probabilistic models
- Few features
- Inverted indexes
- Optimised processing

- Machine learning
- Different models
- Hundreds of features
- (Optimised) Sequential processing
Query Efficiency Prediction

- Predict how long an unseen query will take to execute, before it has executed.
- This facilitates 3+ manners to make a search engine more efficient:
  1. Reconfigure the pipelines of the search engine, trading off a little effectiveness for efficiency
  2. Apply more CPU cores to long-running queries
  3. Decide how to plan the rewrites of a query, to reduce long-running queries
- In each case, increasing efficiency means increased server capacity and energy savings
Dynamic Pruning: MaxScore

\[ \sigma_1 \quad \sigma_2 \quad \sigma_3 \quad \sigma_4 \quad \sigma_5 \]

\[ \text{critical docid} \quad \text{critical docid} \quad \text{critical docid} \quad \text{critical docid} \quad \text{docid space} \]

\[ \text{score space} \]

threshold \( \theta \)
Dynamic Pruning: WAND

\[ \sigma_1 + \sigma_2 + \sigma_3 \]

\[ \sigma_2 + \sigma_3 \]

\[ \sigma_1 + \sigma_3 \]

\[ \sigma_3 \]

\[ \sigma_2 \]

\[ \sigma_1 \]

score

space

docid

critical
docid

critical
docid

critical
docid

critical
docid

critical
docid

critical
docid

threshold \( \theta \)
Efficient Query Processing for Scalable Web Search

Nicola Tonellotto, Craig Macdonald and Iadh Ounis
What makes a single query fast or slow?

- **Query processing strategy** (MaxScore, Wand, BMW)
- **Length of posting lists**
- **Number of terms**
- **Co-occurrence of query terms** (Posting list union/intersection)

2 term queries

4 term queries
Static QEP

- **Static QEP** (Macdonald et al., SIGIR 2012)
  - a *supervised learning* task
  - using *pre-computed* term-level *features* such as
    - the length of the posting lists
    - the variance of scored postings for each term
  - Extended for *long-running queries classification* on the Bing search engine infrastructure (Jeon et al., SIGIR 2014)
  - Extended to *rewritten queries* that include *complex query operators* (Macdonald et al., SIGIR 2017)
Analytical QEP

- **Analytical QEP** (Wu and Fang, CIKM 2014)
  - analytical *model* of query processing efficiency
  - key factor in their model was the number of documents containing pairs of query terms
  - **Intersection size** not precomputed but estimated with
    \[
    A(t_1, t_2) = \frac{N_1}{N} \times \left( \frac{N_2}{N} \right)^\delta \times N,
    \]
    - $N = \text{num docs in collection}$
    - $N_1 = t_1 \text{ posting list length}$
    - $N_2 = t_2 \text{ posting list length}$
    - $\delta = \text{control parameter set to 0.5}$
Dynamic QEP

- **Dynamic QEP** (Kim et al, WSDM 2015)
  - Predictions after a **short period** of query processing **has elapsed**
  - Able to determine **how well** a query is **progressing**
  - Use the period to **better estimate** the query’s completion time
  - **Supervised learning** task
  - Must be **periodically re-trained** as new queries arrive
  - The dynamic **features** are naturally **biased towards the first portion** of the index used to calculate them
  - With various index orderings possible, it is plausible that the **first portion of the index does not reflect well the term distributions** in the rest of the index
  - **More accurate** than **predictions** based on pre-computed features or an analytical model
Can be used to estimate the expected number of documents processed in any query, processed either in OR mode (union of posting lists) or in AND mode (intersection of posting lists)
Research Questions

1. Compression of an index synopsis
2. Space overheads of an index synopsis
3. Time overheads of an index synopsis
4. Posting list estimates accuracy w.r.t. AND/OR retrieval
5. Posting list estimates accuracy w.r.t. dynamic pruning
6. Accuracy of overall response time prediction
7. Accuracy of long-running queries classification
Experimental Setup

- TREC ClueWeb09-B corpus (50 million English web pages)
- Indexing and retrieval using the **Terrier** IR platform
- Stopwords removal and stemming
- Docids are assigned according to their descending **PageRank** score
- Compressed using **Elias-Fano** encoding
- Retrieving **50,000 unique queries** from the TREC 2005 Efficiency Track topics
- Scoring with **BM25**, with a block size of 64 postings for BMW
- Retrieved **1000** documents per query
- **Learning** performed 4,000 train and 1,000 test queries
- All indices are **loaded in memory** before processing starts
- Single core of a 8-core Intel i7-7770K with 64 GiB RAM
- **Sampling probabilities** $\gamma = 0.001, 0.005, 0.01, 0.05$
# Compression & Space Overheads

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>Postings (M)</th>
<th>Original docids</th>
<th>Remapped docids</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Space (GiB)</td>
<td>Reduction</td>
</tr>
<tr>
<td>1</td>
<td>14,795</td>
<td>19.07</td>
<td>–</td>
</tr>
<tr>
<td>0.001</td>
<td>15</td>
<td>0.29</td>
<td>66×</td>
</tr>
<tr>
<td>0.005</td>
<td>74</td>
<td>0.41</td>
<td>47×</td>
</tr>
<tr>
<td>0.01</td>
<td>148</td>
<td>0.56</td>
<td>34×</td>
</tr>
<tr>
<td>0.05</td>
<td>739</td>
<td>1.58</td>
<td>12×</td>
</tr>
</tbody>
</table>

![Graphs](Image)
## Time Overheads

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>0.001</th>
<th></th>
<th></th>
<th>0.005</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Syn</td>
<td>Total</td>
<td></td>
<td>Syn</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+0.1%)</td>
<td></td>
<td></td>
<td>(+0.6%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+0.1%)</td>
<td></td>
<td></td>
<td>(+0.5%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+0.1%)</td>
<td></td>
<td></td>
<td>(+0.5%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+0.1%)</td>
<td></td>
<td></td>
<td>(+0.7%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+0.2%)</td>
<td></td>
<td></td>
<td>(+0.8%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+1.2%)</td>
<td></td>
<td></td>
<td>(+5.9%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+1.0%)</td>
<td></td>
<td></td>
<td>(+4.9%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+0.9%)</td>
<td></td>
<td></td>
<td>(+5.2%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+1.1%)</td>
<td></td>
<td></td>
<td>(+5.8%)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(+1.6%)</td>
<td></td>
<td></td>
<td>(+7.9%)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>0.01</th>
<th></th>
<th></th>
<th>0.05</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Syn</td>
<td>Total</td>
<td></td>
<td>Syn</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(+1.2%)</td>
<td></td>
<td></td>
<td>(+5.9%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(+1.0%)</td>
<td></td>
<td></td>
<td>(+4.9%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(+0.9%)</td>
<td></td>
<td></td>
<td>(+5.2%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(+1.1%)</td>
<td></td>
<td></td>
<td>(+5.8%)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(+1.6%)</td>
<td></td>
<td></td>
<td>(+7.9%)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Union & Intersection Estimates Accuracy

Intersection

Analytical model

Index synopsis
Actual vs. Synopsis Response Times

MaxScore

WAND

BMW
## Overall Response Time Accuracy

<table>
<thead>
<tr>
<th>Strategy</th>
<th>MRT</th>
<th>Static RMSE</th>
<th>Dynamic RMSE</th>
<th>Synopsis RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxScore (Post)</td>
<td>87.7</td>
<td>37.8</td>
<td>48.7</td>
<td>37.0</td>
</tr>
<tr>
<td>MaxScore (Time)</td>
<td></td>
<td></td>
<td></td>
<td>48.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23.5</td>
</tr>
<tr>
<td>WAND (Post)</td>
<td>107.4</td>
<td>52.3</td>
<td>63.7</td>
<td>71.4</td>
</tr>
<tr>
<td>WAND (Time)</td>
<td></td>
<td></td>
<td></td>
<td>88.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>62.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>62.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>62.5</td>
</tr>
<tr>
<td>BMW (Post)</td>
<td>77.8</td>
<td>30.0</td>
<td>33.8</td>
<td>65.2</td>
</tr>
<tr>
<td>BMW (Time)</td>
<td></td>
<td></td>
<td></td>
<td>78.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>17.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15.1</td>
</tr>
</tbody>
</table>
## Long-running Query Classification

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>MaxScore</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.001 0.005 0.01 0.05</td>
<td>0.001 0.005 0.01 0.05</td>
<td></td>
</tr>
<tr>
<td><strong>Static</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Dynamic</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synopsis (Post)</td>
<td>86.1 ‡ 86.0 ‡ 86.9 ‡‡ 87.3 ‡‡</td>
<td>77.2 ‡ 84.9 ‡ 85.0 ‡‡ 85.9 ‡‡</td>
<td></td>
</tr>
<tr>
<td>Synopsis (Time)</td>
<td>96.1 † 92.9 ‡‡ 93.9 ‡‡ 95.4 ‡‡</td>
<td>46.8 † 91.0 ‡‡ 95.0 ‡‡ 94.8 ‡‡</td>
<td></td>
</tr>
<tr>
<td><strong>WAND</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td>88.5</td>
<td>75.7</td>
<td></td>
</tr>
<tr>
<td>Dynamic</td>
<td>89.1</td>
<td>57.9</td>
<td></td>
</tr>
<tr>
<td>Synopsis (Post)</td>
<td>91.7 † 90.8 † 90.5 † 90.9 †</td>
<td>54.0 † 57.8 † 56.6 † 57.4 †</td>
<td></td>
</tr>
<tr>
<td>Synopsis (Time)</td>
<td>89.7 ‡ 87.6 ‡‡ 88.7 ‡‡ 87.5 ‡‡</td>
<td>76.7 ‡ 89.9 ‡‡ 91.5 ‡‡ 92.5 ‡‡</td>
<td></td>
</tr>
<tr>
<td><strong>BMW</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Static</td>
<td>81.2</td>
<td>67.7</td>
<td></td>
</tr>
<tr>
<td>Dynamic</td>
<td>83.0</td>
<td>65.5</td>
<td></td>
</tr>
<tr>
<td>Synopsis (Post)</td>
<td>55.4 ‡‡ 56.6 ‡‡ 56.9 ‡‡ 55.1 ‡‡</td>
<td>24.9 ‡‡ 29.0 ‡‡ 28.0 ‡‡ 28.8 ‡‡</td>
<td></td>
</tr>
<tr>
<td>Synopsis (Time)</td>
<td>87.3 ‡‡ 89.0 ‡‡ 91.0 ‡‡ 90.7 ‡‡</td>
<td>80.0 ‡‡ 85.2 ‡‡ 85.9 ‡‡ 88.9 ‡‡</td>
<td></td>
</tr>
</tbody>
</table>
Query Performance Prediction

- QPP is another use case for index synopsis
- Can we use synopsis for post-retrieval QPP?
- Performance w.r.t. pre-retrieval QPP on full index
- Performance w.r.t. post-retrieval QPP on full index
- Main findings:
  1. many of the post retrieval predictors can be effective on very small synopsis indices
  2. high correlations with the same predictors calculated on the full index
  3. more effective than the best pre-retrieval predictors
  4. computation requires an almost negligible amount of time
- More details in the journal article
Conclusions & Future Works

- QEP is fundamental component that plans a query’s execution appropriately.
- Index synopses are random samples of complete document indices.
- Able to reproduce the dynamic pruning behavior of the MaxScore, WAND and BMW strategies on a full inverted index.
  - 0.5% of the original collection is enough to obtain accurate query efficiency predictions for dynamic pruning strategies.
  - Used to estimate the processing times of queries on the full index.
- Post-retrieval query performance predictors calculated on an index synopsis can outperform pre-retrieval query performance predictors.
  - 0.1% of the original collection outperforms pre-retrieval predictors by 73%.
  - 5% of the original collection outperforms pre-retrieval predictors by 103%.
- What about applying index synopses across a tiered index layout?
- What about sampling at snippet/paragraph granularity?
- How document/snippet sampling can be combined with a neural ranking model for the first-pass retrieval to achieve efficient neural retrieval?
Thanks for your attention!