

“Efficient Document Retrieval in Main Memory”

Professor: Dr. Nazli Goharian

Course: CS429 Information Retrieval

Team: Rashi Khurana

Steve Talbot

Authors: W. Bruce Croft

Trevor Strohman

Background

> “Traditional IR Systems”

- Disk Access
 - Major Bottleneck
 - Seek times poor

> “Optimization Goals”

- “Galago” IR System
- Read less data, less data to process
- Fewer “disk seeks” (less disk I/O), better performance
- Skip Unnecessary Data

Background

- > “Optimization Approaches”
 - Store entire inverted index in **memory**
 - 64-bit MPUs, growing RAM capacity
 - Improves performance
 - “Dynamic” **skip pointers** $\sim f(\text{posting list length})$
 - “**Impact Weights**”
 - “w_{td}” stored in inverted index at **index build time**
 - “Accumulator” collection constructs SC
 - Dynamic “pruning” removes non-top documents
 - Parallel Processing (Utilize multi-core MPUs)
 - Abbreviated Vocabulary Table (Lexicon)

Background

- “Query Term Weights” (w_{tq})
 - Calculated at query processing time (QPT)
- “Document Term Weights” (w_{td})
 - Calculated at index building time (IBT)

$$w_{ij} = \frac{(1 + \log tf_{ij}) * idf_i}{\sum_{i=1}^t [(1 + \log tf_{ij}) * idf_i]^2}$$

$$w_{ij} = (1 + \ln tf_{ij}) * \ln(1 + \frac{df_{\max i}}{df_i})$$

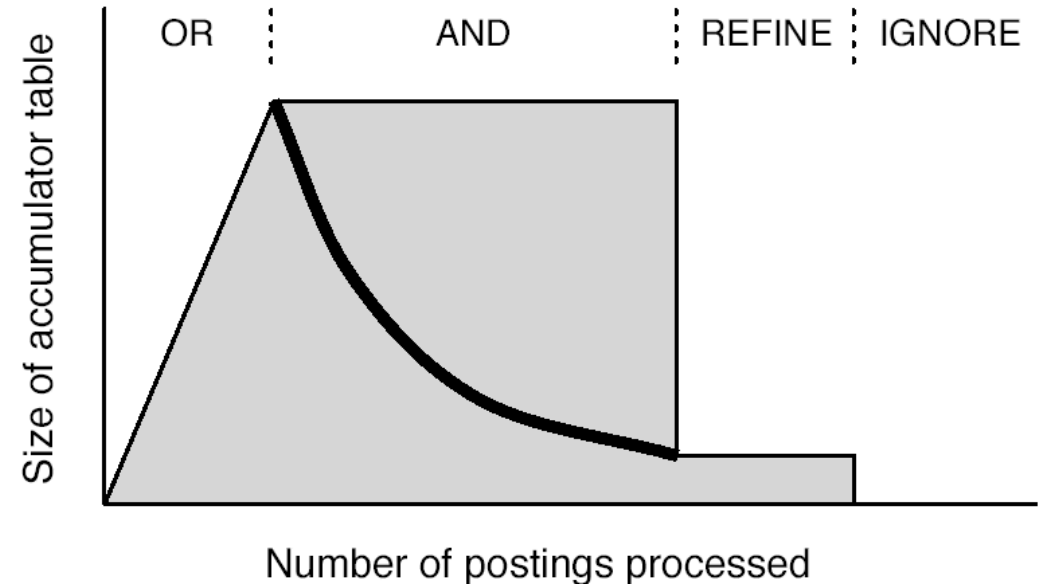
$$SC(Q, D_i) = \sum_{j=1}^t w_{t,q} * w_{t,d}$$

Background

- “Posting Lists” (PLs)
 - Posting List Entry (PLE)
 - “variable byte encoding compression”
 - PLE → <docId, w_td>
 - Segments / Bins / Pages
 - Segments → PLEs have the same “w_td”
 - PL segments sorted in order of “w_td”
 - “Blocks”
 - Each PL must reside in partitions of 32KB “blocks”

Background

- “Accumulator” collection
 - Array, sorted by docId
 - Top “k” tuples



- “**Tuples**” created at QPT
 - $\langle T, s, d \rangle$
 - “T” = term set, set of terms for this docId [1 B]
 - “s” = impact score, $SC(D_i)$ [3 B]
 - “d” = docId [4 B]
- “**Or**” mode / “**And**” mode

Parameters

- > **“ τ ” = threshold (“tao”)**
 - Lowest score in “A” shown to user
 - “ k^{th} ” largest score in “A”
 - Every document shown to user has $SC > \tau$

- > **$\tau \sim f(k)$**
 - “k” = top # of documents in A
 - k is constant, τ changes to accommodate k
 - $\tau = \text{“getKthUniqueScore(A, k)”}$

- > **“ ρ ” = remainder (“rho”)**
 - $\rho(A_d) = SC$ possible for A_d with more processing
 - $A_d + \rho(A_d) < t$
 - If the accumulator (tuple) is not in the top k results now, it never will be, no matter how many additional posting list entries are processed

```

procedure PROCESSQUERY( $Q$ )
     $A \leftarrow \{\}$  ▷ The accumulator table
     $S \leftarrow \{\}$  ▷ List of accumulator segments
    for all query terms  $t \in Q$  do
         $I_t \leftarrow$  inverted list for term  $t$ 
         $w_{t,q} \leftarrow$  weight for term  $t$ 
        for all segments  $I_{t,s} \in I_t$  do
            add  $\langle t, w_{t,q} \times w_{t,d}, I_{t,s} \rangle$  to  $S$ 
        end for
    end for
    sort  $S$  in descending order by segment score
    for all segments  $\langle t, w, I_{t,s} \rangle$  in  $S$  do
        ProcessSegment( $A, w, I_{t,s}$ )
        TrimAccumulatorList( $A, S$ )
        if CanQuit( $A, S$ ) then
            break
        end if
    end for
    sort  $A$  by score, return top  $k$  results
end procedure

```



```

procedure PROCESSSEGMENT( $A, w, I_{q,s}$ )
  for all documents  $d$  in segment  $I_{q,s}$  do
    if  $A_d \notin A$  and OrMode = true then
       $A_d \leftarrow 0$ 
    end if
    if  $A_d \in A$  then
       $A_d \leftarrow A_d + w$ 
    end if
  end for
end procedure

```

```

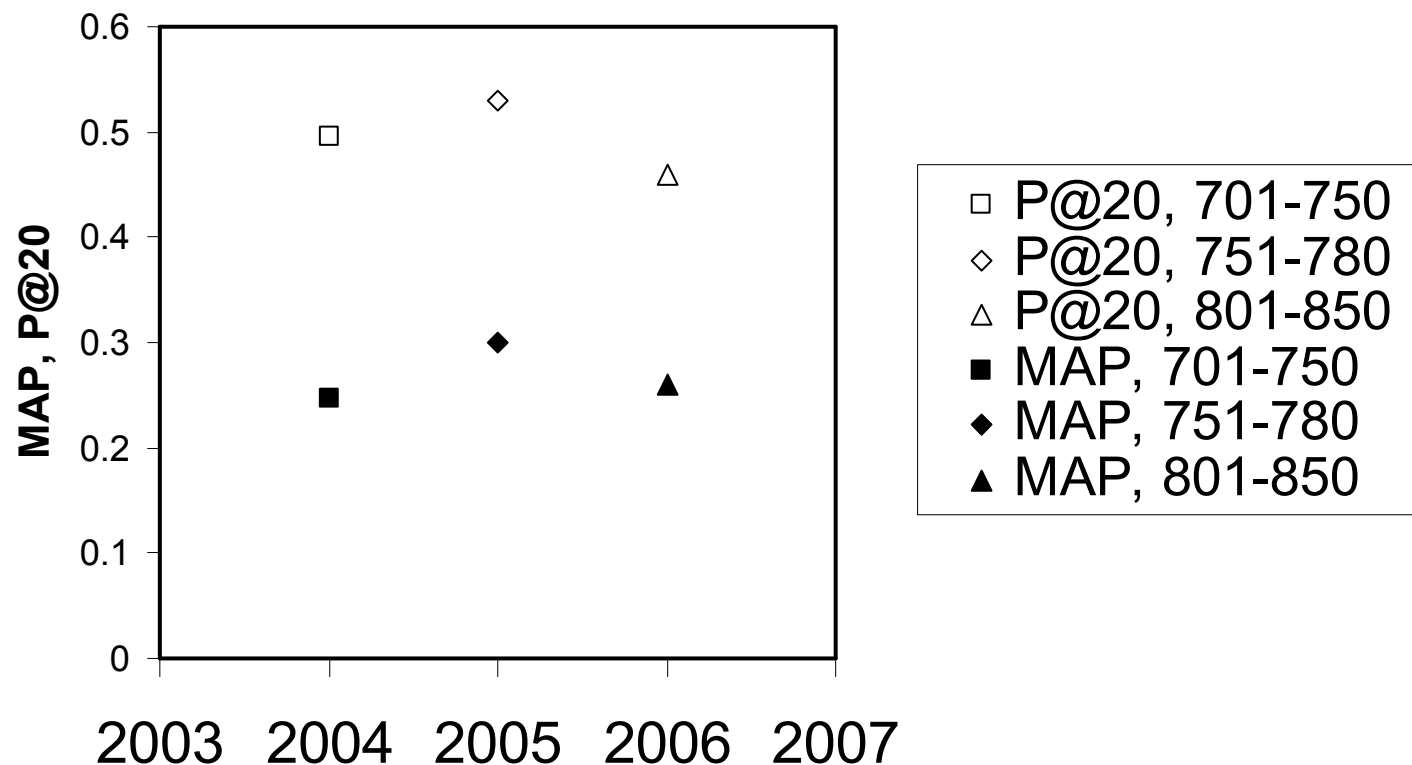
procedure PROCESSSEGMENTPRUNED( $A, q, w, I_{q,s}$ )
  OrMode  $\leftarrow \tau < \rho(Q)$ 
  ProcessSegment( $A, q, w, I_{q,s}$ )
end procedure

```

```
procedure TRIMACCUMULATORLIST( $A, S$ )  
  for all accumulators  $A_d$  in  $A$  do  
    if  $A_d + \rho(A_d) < \tau$  then  
      remove  $A_d$  from  $A$   
    end if  
  end for  
end procedure
```

Compression Techniques and Data Set

- “Porter2” English stemmer
 - <http://snowball.tartarus.org/algorithms/english/stemmer.html>
- Used stopwords list of 600 common terms
 - <http://goanna.cs.rmit.edu.au/~jz/resources/stopping.zip>
- “TREC GOV2 collection”
 - 25.2 M web pages crawled from the .gov domain
 - Text data consisting of 426 GB of space
- “TREC Terabyte Ad Hoc and Efficiency topics”
 - Same thing as “numbered queries”
 - <http://trec.nist.gov/pubs/trec13/papers/TERA.OVERVIEW.pdf>

*Data Set***MAP, P@20 vs. Topics**

Query set	MAP	P@20
Topics 701-750 (2004)	0.2460	0.4949
Topics 751-800 (2005)	0.3004	0.5290
Topics 801-850 (2006)	0.2592	0.4590

Table 1: Effectiveness on TREC Ad Hoc Queries

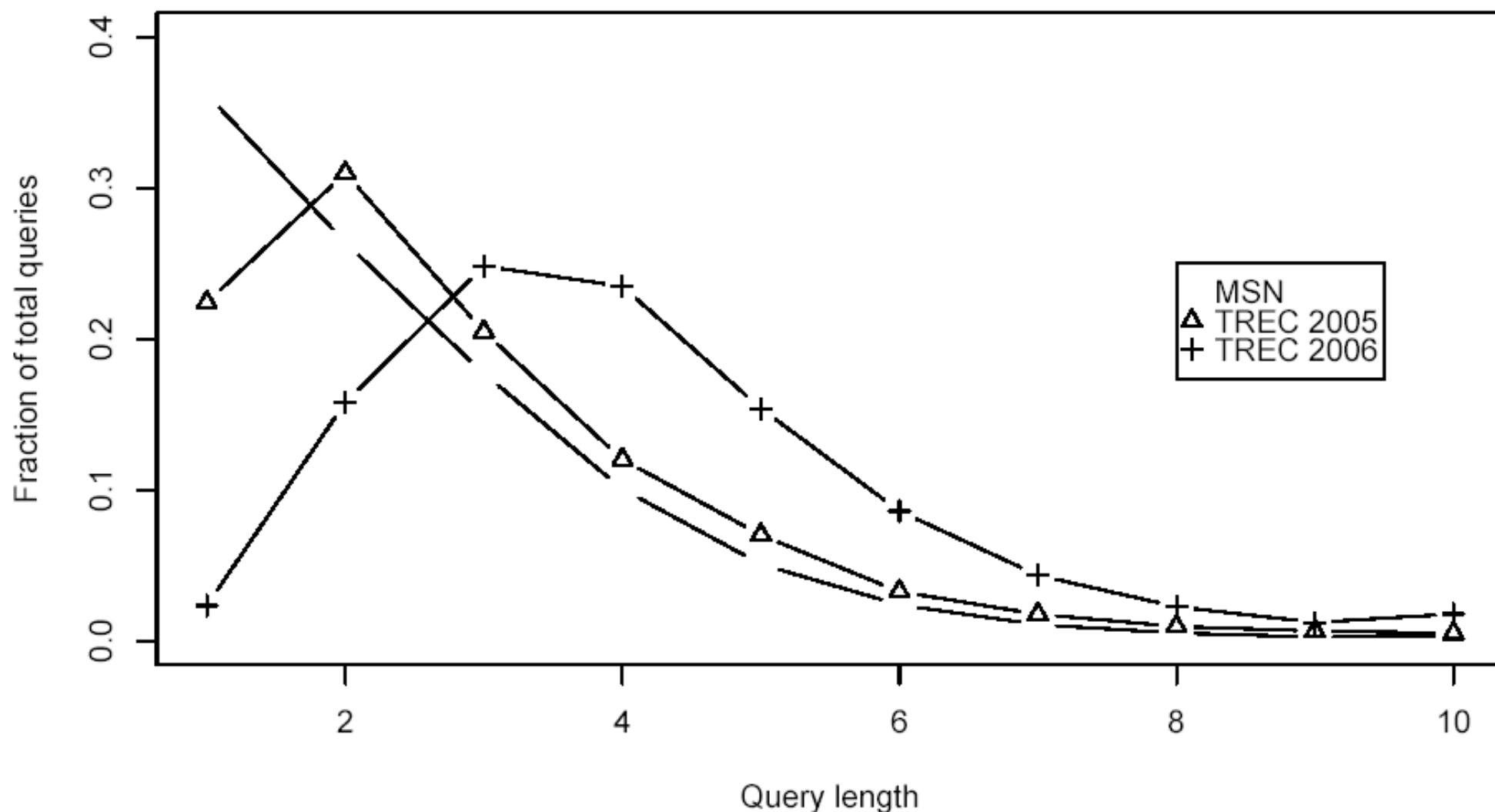
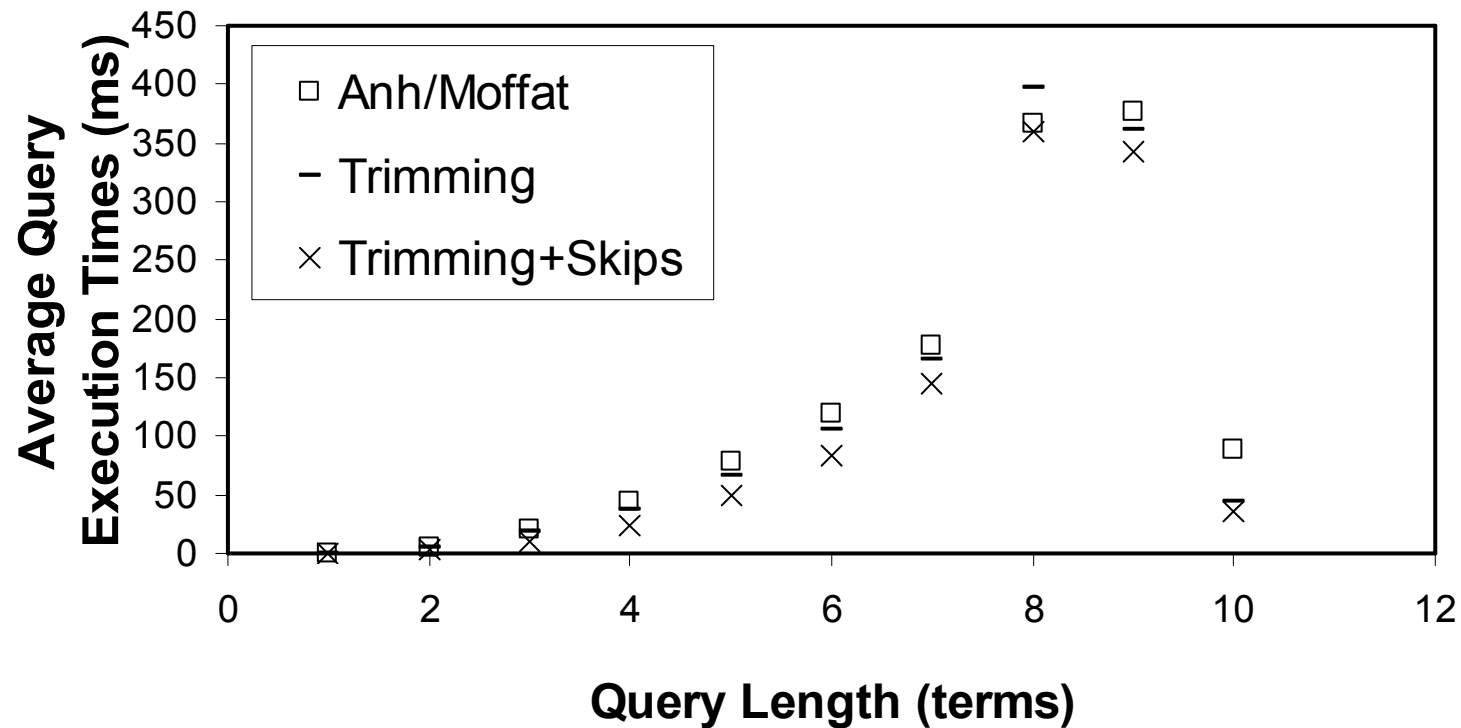
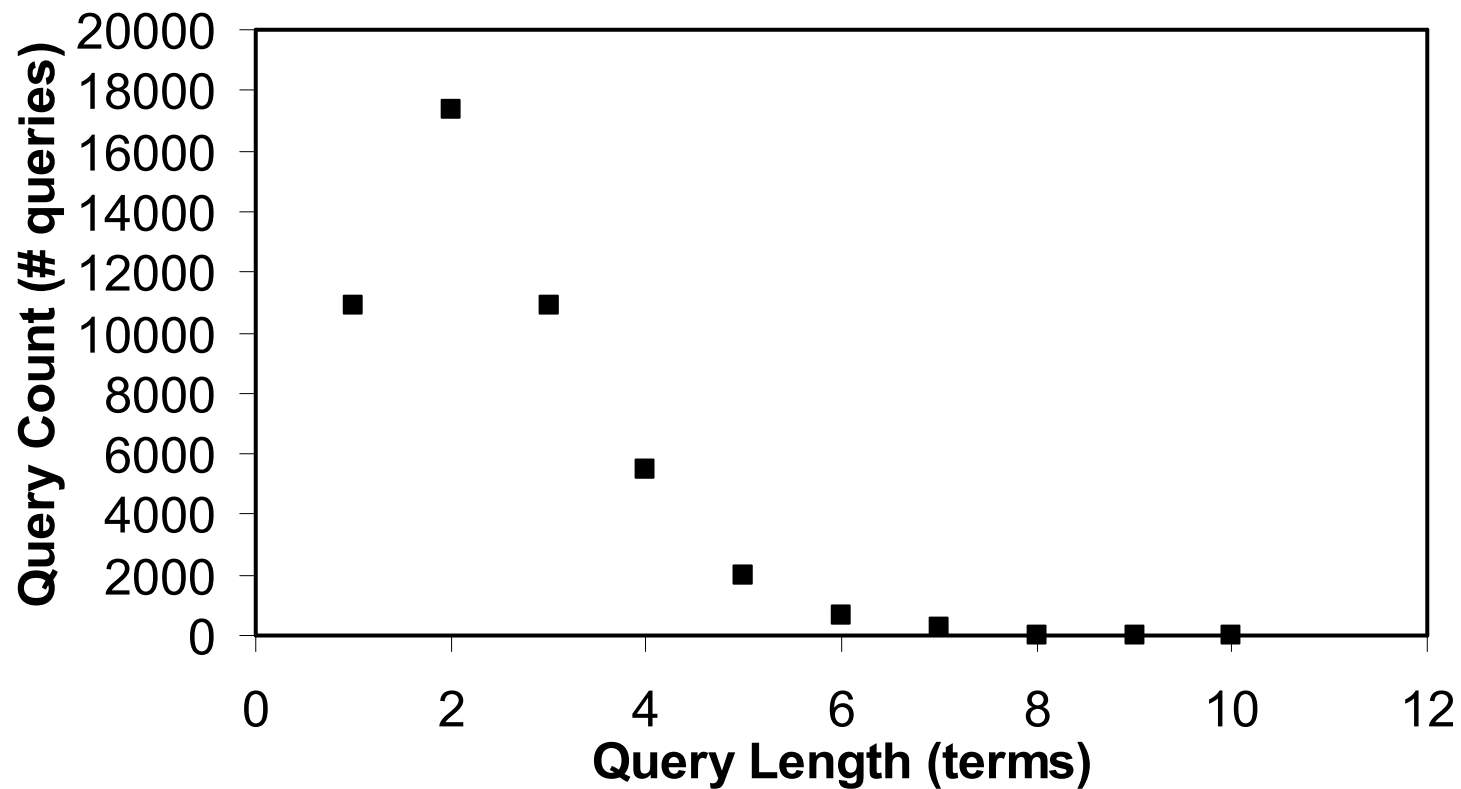
Data Set

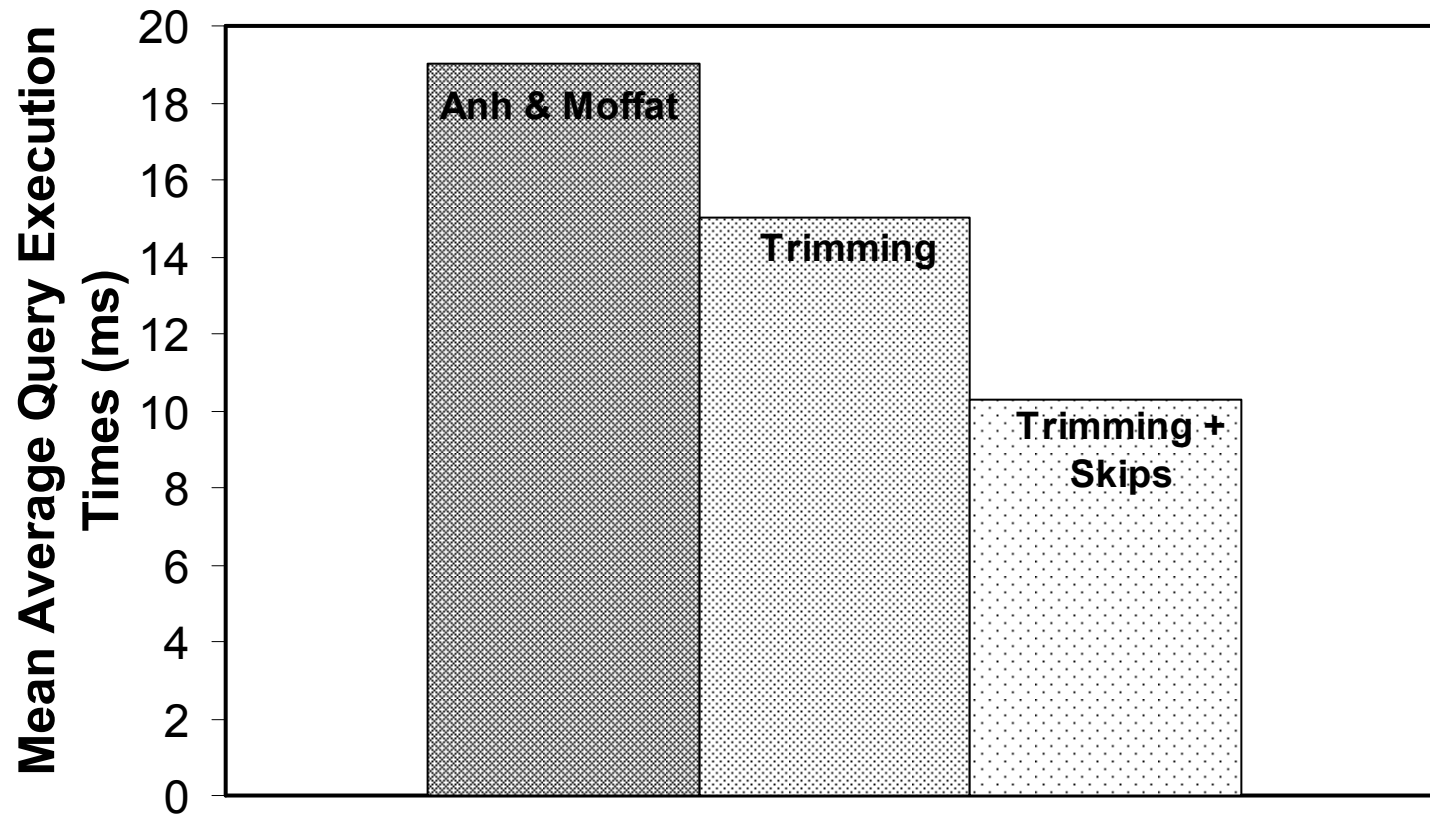
Figure 3: Distribution of query lengths (before removing stopwords) across collections.

*Data Set***Average Query Execution Times vs. Query Length, TREC 2005**

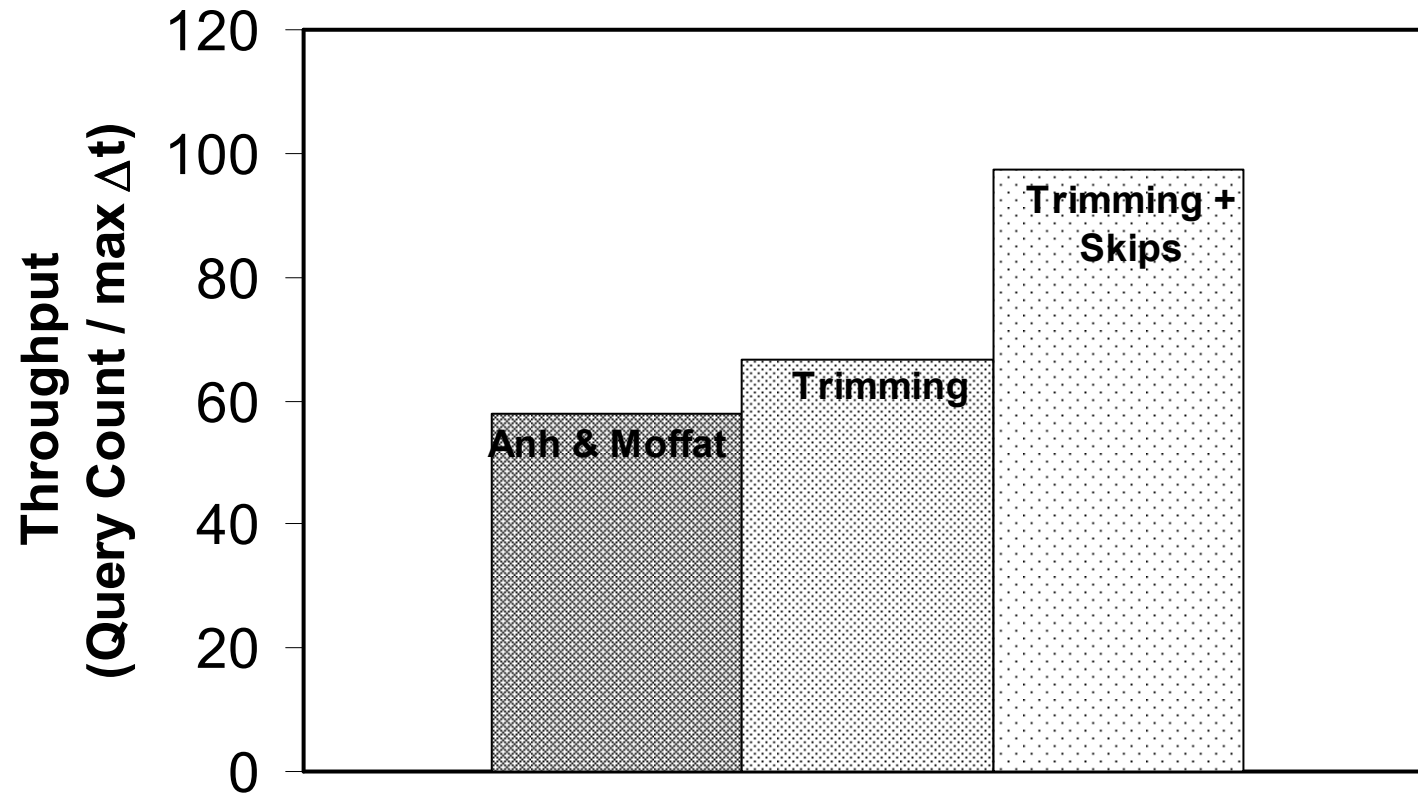
Method	All queries		Query length (terms)									
	Mean	Throughput	1	2	3	4	5	6	7	8	9	10
Query count	47,543	47,543	10,899	17,347	10,888	5,489	1,965	683	233	32	6	1
Unoptimized	317.2	3.2	22	144	410	724	1149	1535	1972	3499	3197	1181
Anh/Moffat	19.0	57.8	0.2	5.5	21	44	79	119	178	366	376	89
Trimming	15.0	66.7	0.1	5.6	18	37	67	105	166	397	362	44
Trimming+Skips	10.3	97.5	0.2	2.9	10	24	49	84	145	360	342	35

*Data Set***Query Count vs. Query Length**

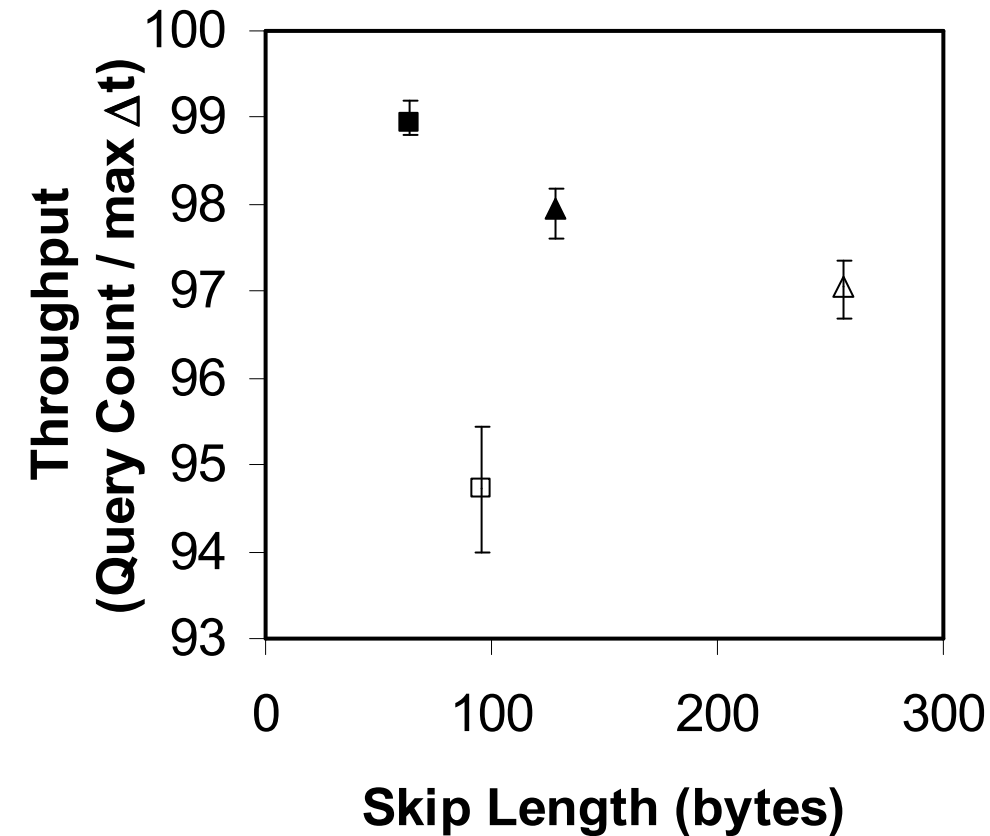
Method	All queries		Query length (terms)									
	Mean	Throughput	1	2	3	4	5	6	7	8	9	10
Query count	47,543	47,543	10,899	17,347	10,888	5,489	1,965	683	233	32	6	1
Unoptimized	317.2	3.2	22	144	410	724	1149	1535	1972	3499	3197	1181
Anh/Moffat	19.0	57.8	0.2	5.5	21	44	79	119	178	366	376	89
Trimming	15.0	66.7	0.1	5.6	18	37	67	105	166	397	362	44
Trimming+Skips	10.3	97.5	0.2	2.9	10	24	49	84	145	360	342	35

*Data Set***Mean Average Query Execution Times**

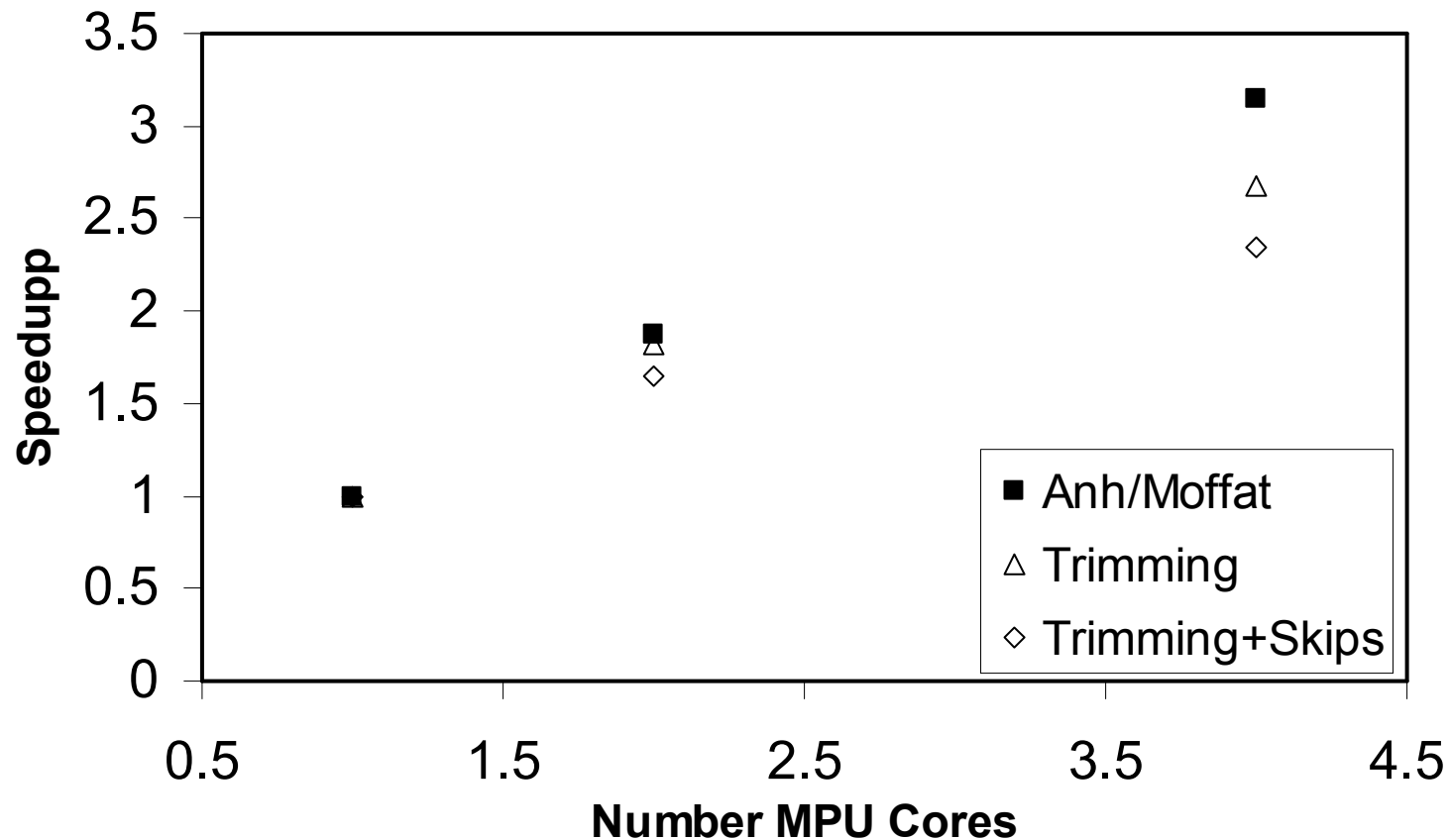
Method	All queries		Query length (terms)									
	Mean	Throughput	1	2	3	4	5	6	7	8	9	10
Query count	47,543	47,543	10,899	17,347	10,888	5,489	1,965	683	233	32	6	1
Unoptimized	317.2	3.2	22	144	410	724	1149	1535	1972	3499	3197	1181
Anh/Moffat	19.0	57.8	0.2	5.5	21	44	79	119	178	366	376	89
Trimming	15.0	66.7	0.1	5.6	18	37	67	105	166	397	362	44
Trimming+Skips	10.3	97.5	0.2	2.9	10	24	49	84	145	360	342	35

*Data Set***Throughput**

Method	All queries		Query length (terms)									
	Mean	Throughput	1	2	3	4	5	6	7	8	9	10
Query count	47,543	47,543	10,899	17,347	10,888	5,489	1,965	683	233	32	6	1
Unoptimized	317.2	3.2	22	144	410	724	1149	1535	1972	3499	3197	1181
Anh/Moffat	19.0	57.8	0.2	5.5	21	44	79	119	178	366	376	89
Trimming	15.0	66.7	0.1	5.6	18	37	67	105	166	397	362	44
Trimming+Skips	10.3	97.5	0.2	2.9	10	24	49	84	145	360	342	35

*Data Set***Throughput vs. Skip Length**

Skip Length	Throughput		
	Mean	Low	High
64	98.94	98.80	99.20
96	94.73	93.99	95.45
128	97.95	97.60	98.18
256	97.05	96.68	97.36
Model	99.21	98.44	99.83
Model (Large)	99.14	99.05	99.27
Model (Small)	98.69	98.40	98.96

*Data Set***Throughput vs. Speedup**

Method	1 core		2 cores		4 cores	
	Throughput	Speedup	Throughput	Speedup	Throughput	Speedup
Anh/Moffat	57.8	1.00	108.3	1.87	181.6	3.14
Trimming	66.7	1.00	121.1	1.81	178.2	2.67
Trimming+Skips	97.5	1.00	161.2	1.65	228.8	2.35

Conclusions

- > Less memory accessed
 - Index organized for fast skipping
 - Similar to “dynamic index pruning” of QPT
 - “Skip lengths” changed dynamically (based on segment size)
 - Query termination
- > Memory-based index preferred
- > Parallel MPU's increase throughput, not linearly scalable
- > “Trimming + Skipping” improves throughput by 69%
- > **“Query Processing Efficiency” increases,
no loss in effectiveness**

Papers

- [1] “Efficient Document Retrieval in Main Memory”, Croft, Bruce, W.; Strohman, Trevor; p 175 - 182, SIGIR 2007 Proceedings, 2007
- [2] “Simplified Similarity Scoring Using Term Ranks”, Anh, Vo Ngoc; Moffat, Alistair; p 1 – 8

END

THANK YOU

```

procedure CANQUIT( $A, S$ )
  for all accumulators  $A_d$  in  $A$  do
    if  $A_d < \tau$  and  $A_d + \rho(A_d) \geq \tau$  then return False
    end if
  end for
   $A' \leftarrow$  all accumulators  $A_i$  in  $A$  such that  $A_i + \rho(A_i) > \tau$ 
  sort  $A'$  in ascending order by score
  for all accumulators  $A'_i$  in  $A'$  do
    if  $A'_i = A'_{i+1}$  and  $\rho(A'_i) > 0$  then return False
    else if  $\rho(A'_i) > A'_{i+1} - A'_i$  then return False
    end if
  end for
end procedure

```

Build Index

> “Indexing”

- 1 - Generate Tuples
 - <docId, term, tf>
- 2 - Form Posting Lists
 - Parallel process #1: calculate idf for each term
 - Parallel process #2: create docId table
- 3 - Binning
 - Create PL segments using (1) and (2)
- 4 – Merging
 - Merge and sort segments into proper PLs