Using an Inverted Index Synopsis for Query Latency and Performance Prediction

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

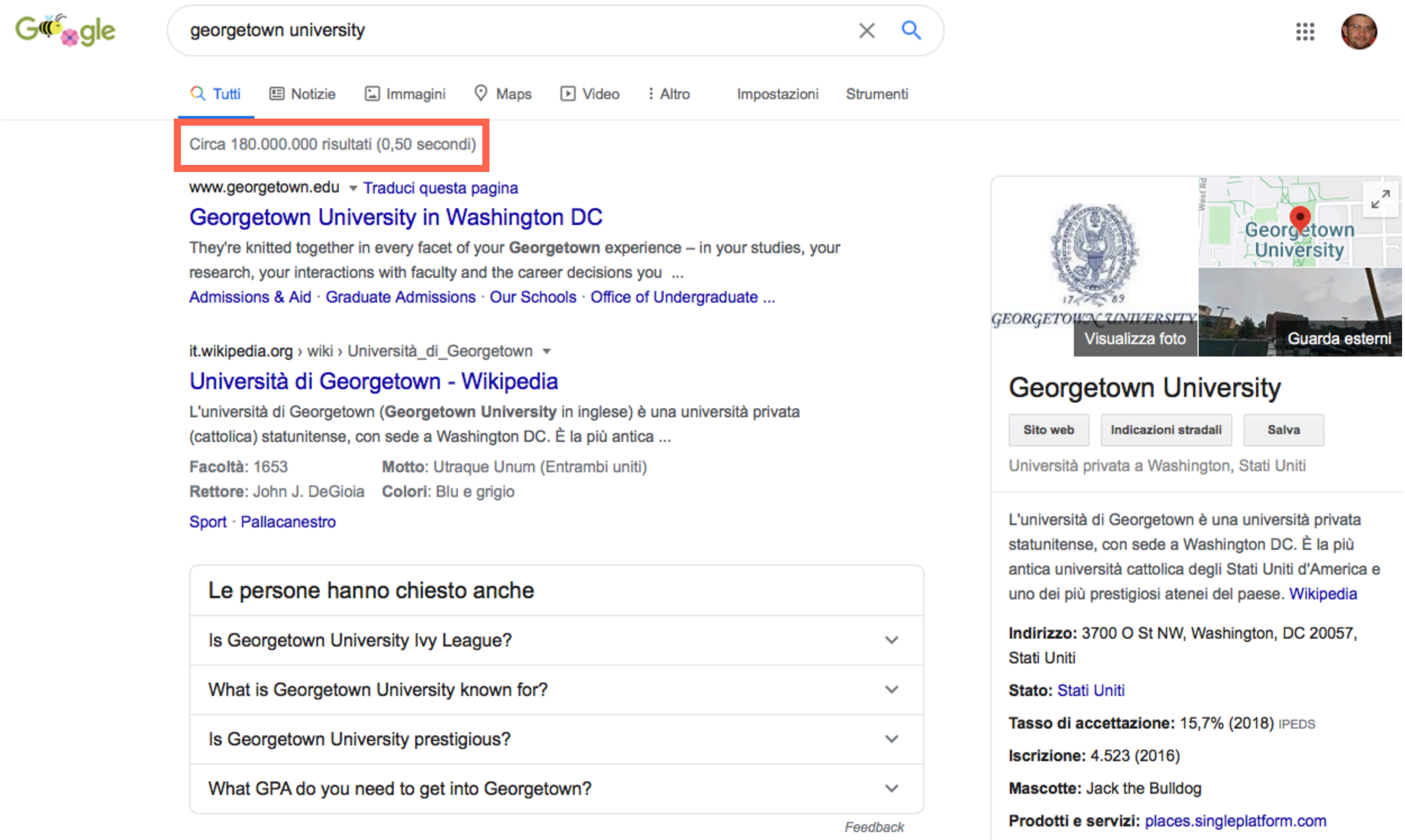
Joint work with Craig Macdonald
University of Glasgow
craig.macdonald@glasgow.ac.uk

To appear in
ACM Transactions on
Information Systems

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

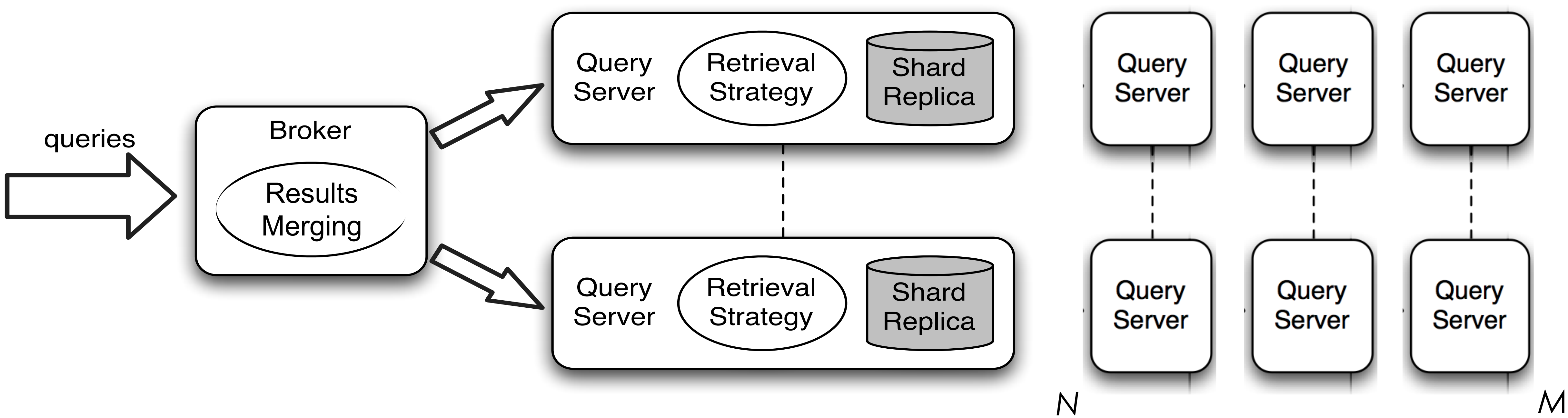


How many documents? In how long?

- 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:



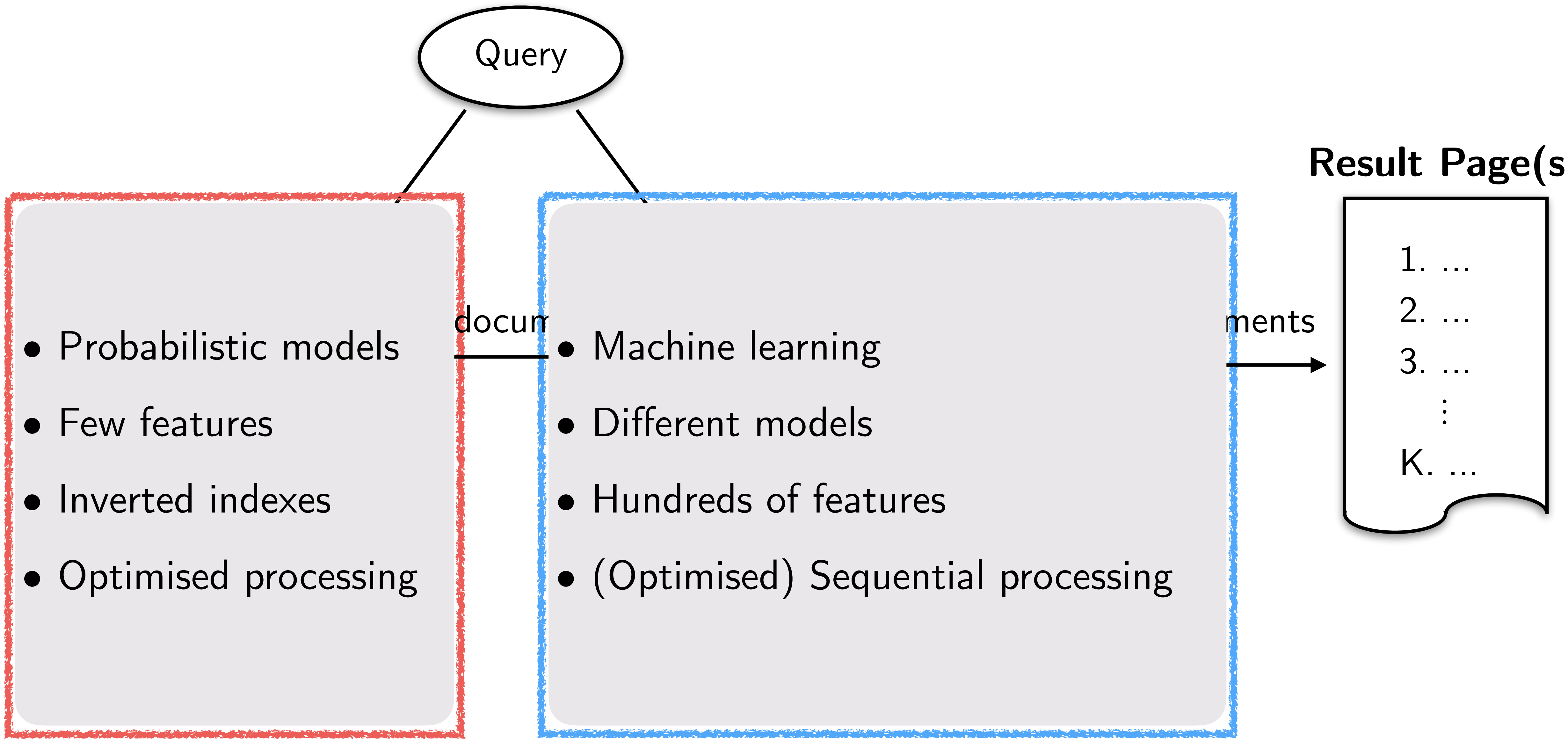
- N & M can be very big:
  - Microsoft's Bing search engine has "hundreds of thousands of query servers"

Computing Platform

Source: https://www.pexels.com/photo/datacenter-server-449401/

Ranking in IR

If we know how long a query will take, can we reconfigure the search engines' ranking pipeline?



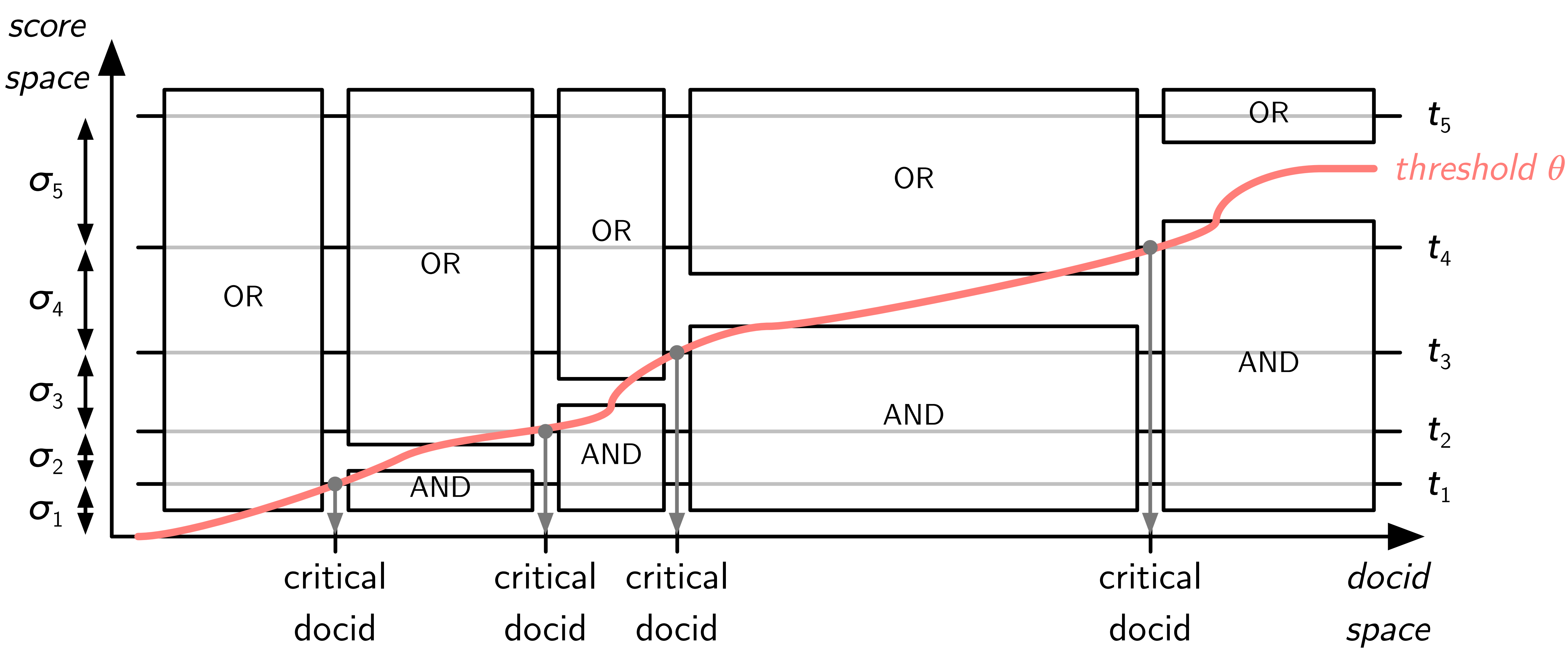
BM25 + DAAT
1,000 – 10,000 docs

Learning To Rank
10 – 100 docs

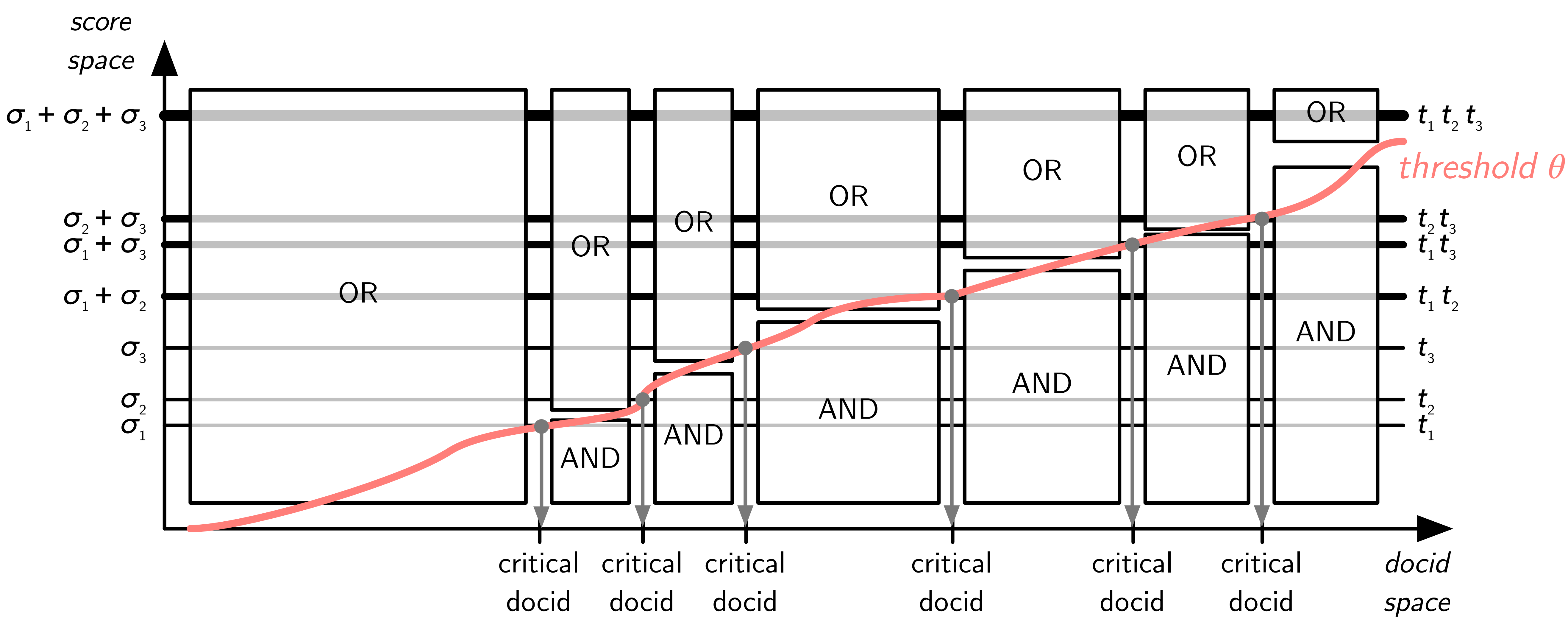
Query Efficiency Prediction

- 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



Dynamic Pruning: WAND



Foundations and Trends® in
Information Retrieval
12:4-5

Efficient Query Processing for Scalable Web Search

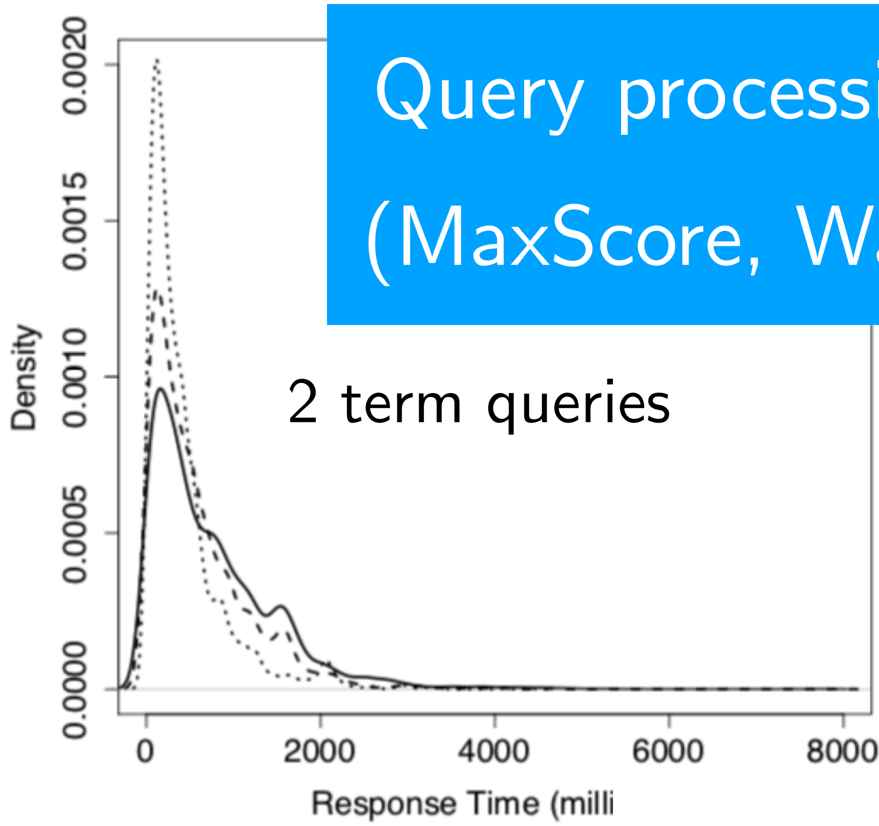
Nicola Tonellotto, Craig Macdonald
and Iadh Ounis

now
the essence of knowledge

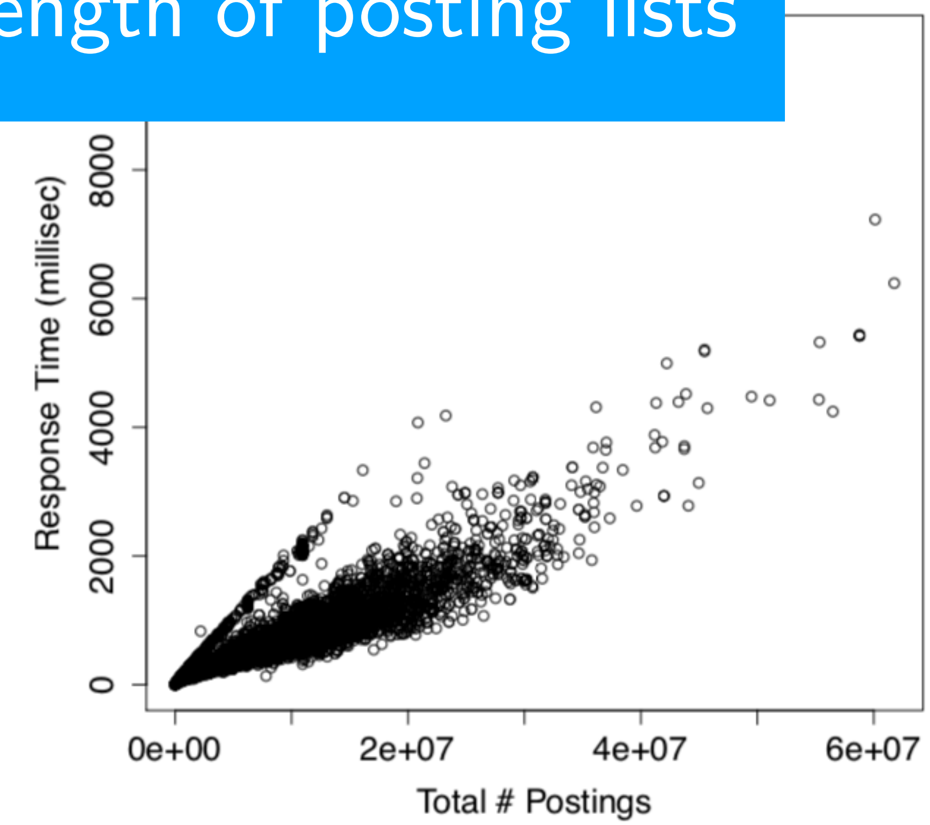
What makes a single query fast or slow?

Query processing strategy

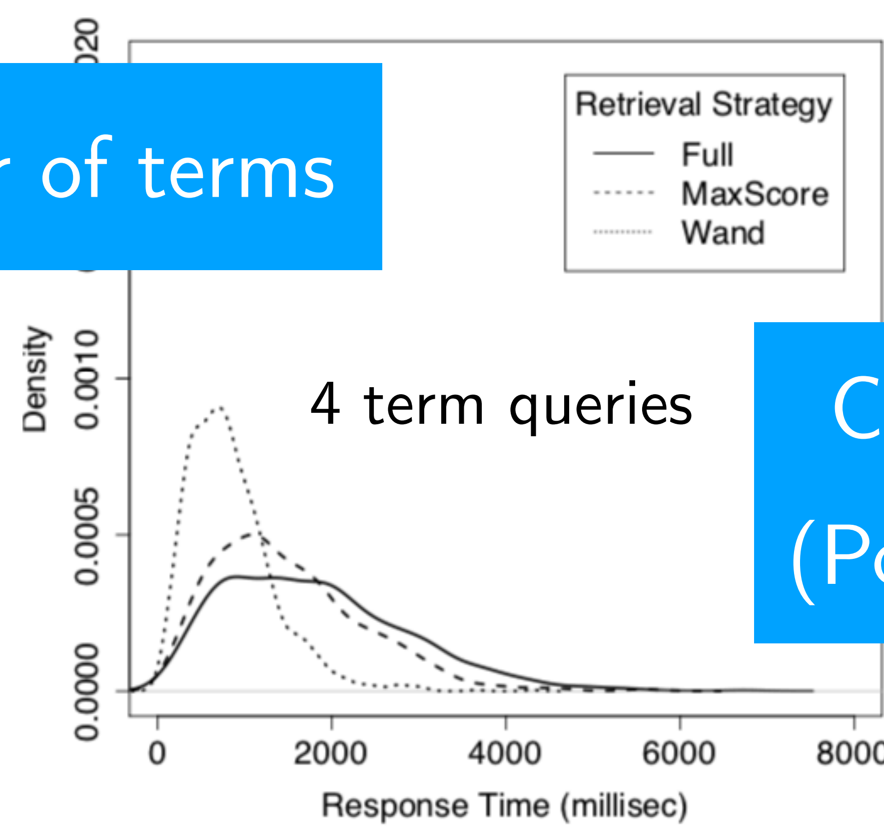
(MaxScore, Wand, BMW)



Length of posting lists



Number of terms



Co-occurrence of query terms

(Posting list union/intersection)

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

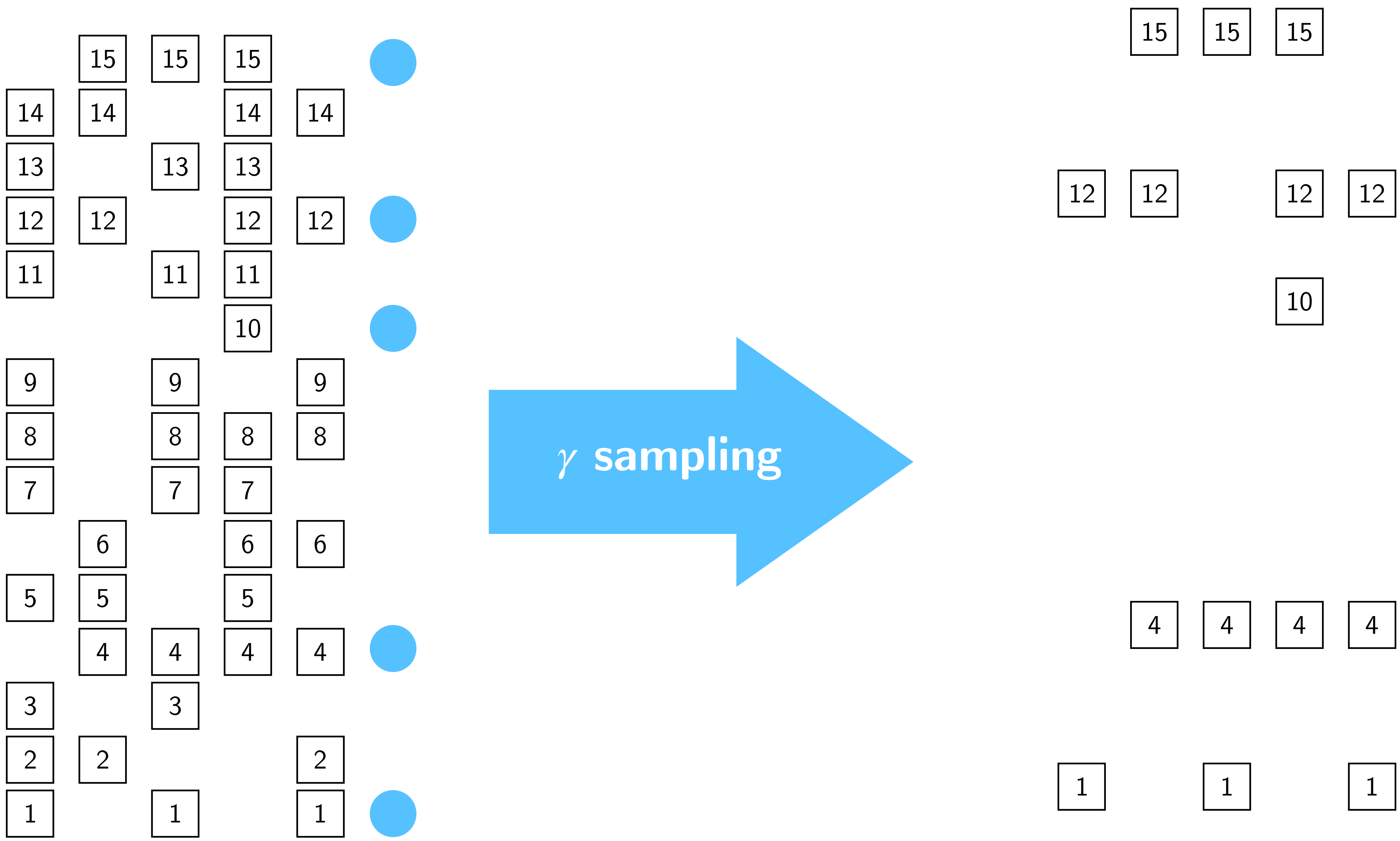
$$A(t_1, t_2) = \frac{N_1}{N} \times \left(\frac{N_2}{N}\right)^{\delta} \times N,$$

    - N = num docs in collection
    - N1 = t1 posting list length
    - N2 = t2 posting list length
    - δ = 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

Index Synopsis



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

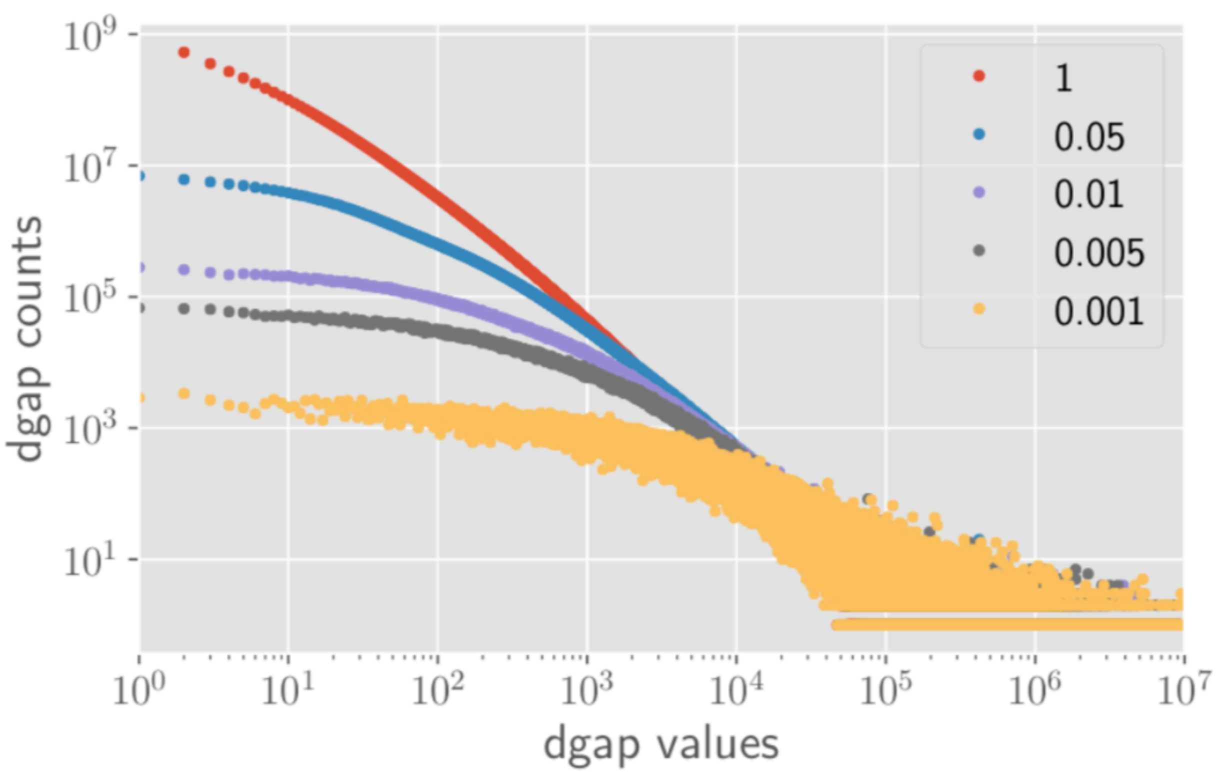
1. **Compression** of an index synopsis
2. **Space overheads** of an index synopsis
3. **Time overheads** of an index synopsis
4. **Posting list estimates** accuracy w.r.t. AND/OR retrieval
5. **Posting list estimates** accuracy w.r.t. dynamic pruning
6. Accuracy of **overall response time prediction**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

Experimental Setup

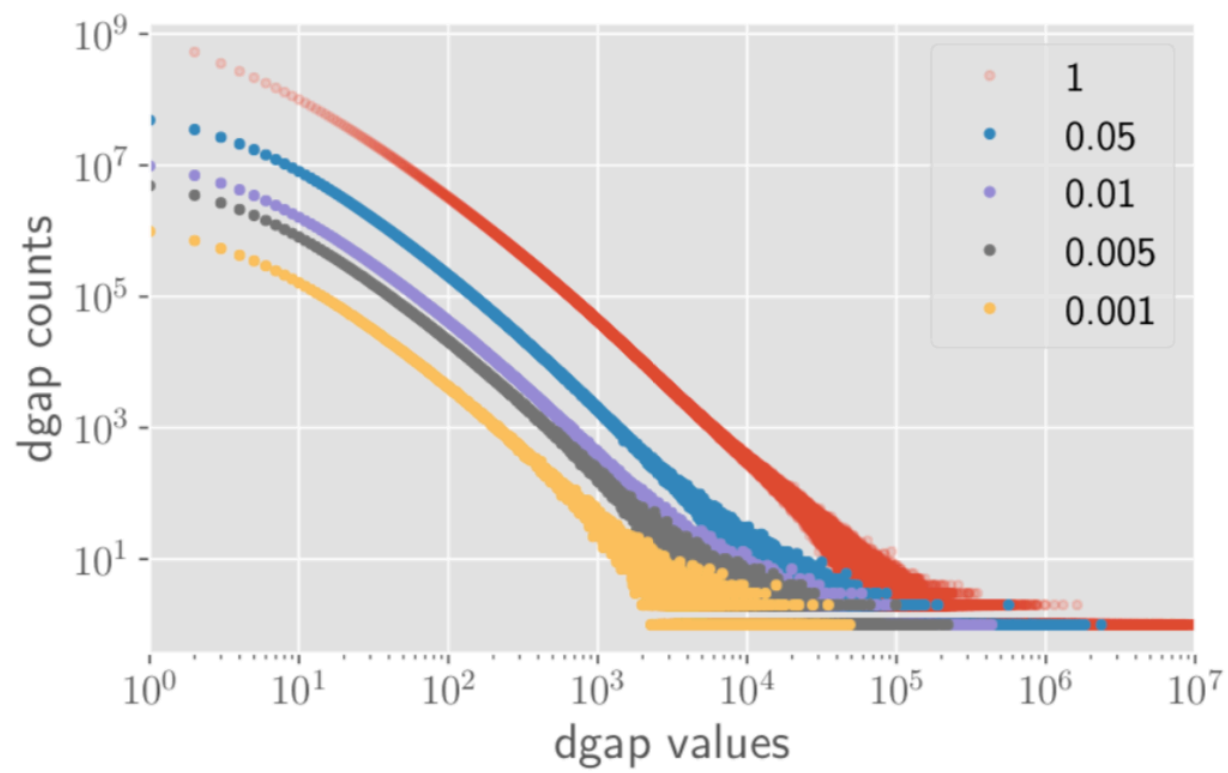
- TREC ClueWeb09-B corpus (**50 million English web pages**)
- Indexing and retrieval using the **Terrier** IR platform
- Stopwords removal and stemming
- Docids are assigned according to their descending **PageRank score**
- Compressed using **Elias-Fano** encoding
- Retrieving **50,000 unique queries** 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

γ	Postings (M)	*original* docids		*remapped* docids	
		Space (GiB)	Reduction	Space (GiB)	Reduction
1	14,795	19.07	–	19.07	–
0.001	15	0.29	66×	0.18	106×
0.005	74	0.41	47×	0.27	71×
0.01	148	0.56	34×	0.37	52×
0.05	739	1.58	12×	1.14	17×



Original docids



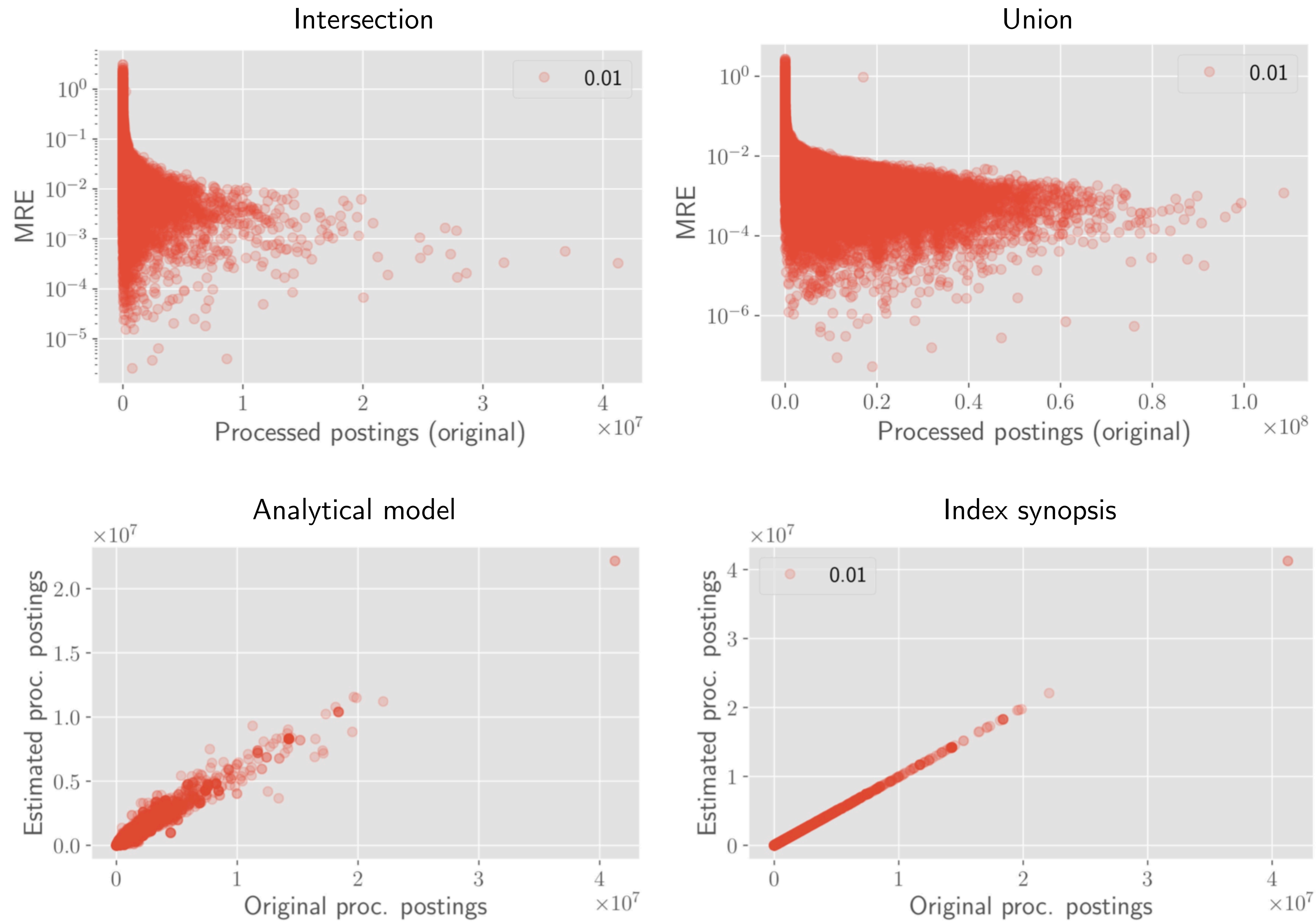
Remapped docids

Time Overheads

	Full	0.001 Syn		0.001 Total		0.005 Syn		0.005 Total	
AND	54.3	0.06	(835×)	54.36	(+0.1%)	0.32	(170×)	54.62	(+0.6%)
OR	450.0	0.45	(1004×)	450.45	(+0.1%)	2.22	(202×)	452.22	(+0.5%)
MaxScore	87.7	0.08	(1129×)	87.78	(+0.1%)	0.40	(220×)	88.10	(+0.5%)
Wand	107.4	0.12	(905×)	107.52	(+0.1%)	0.61	(175×)	108.01	(+0.7%)
BMW	77.8	0.12	(664×)	77.92	(+0.2%)	0.60	(130×)	78.40	(+0.8%)

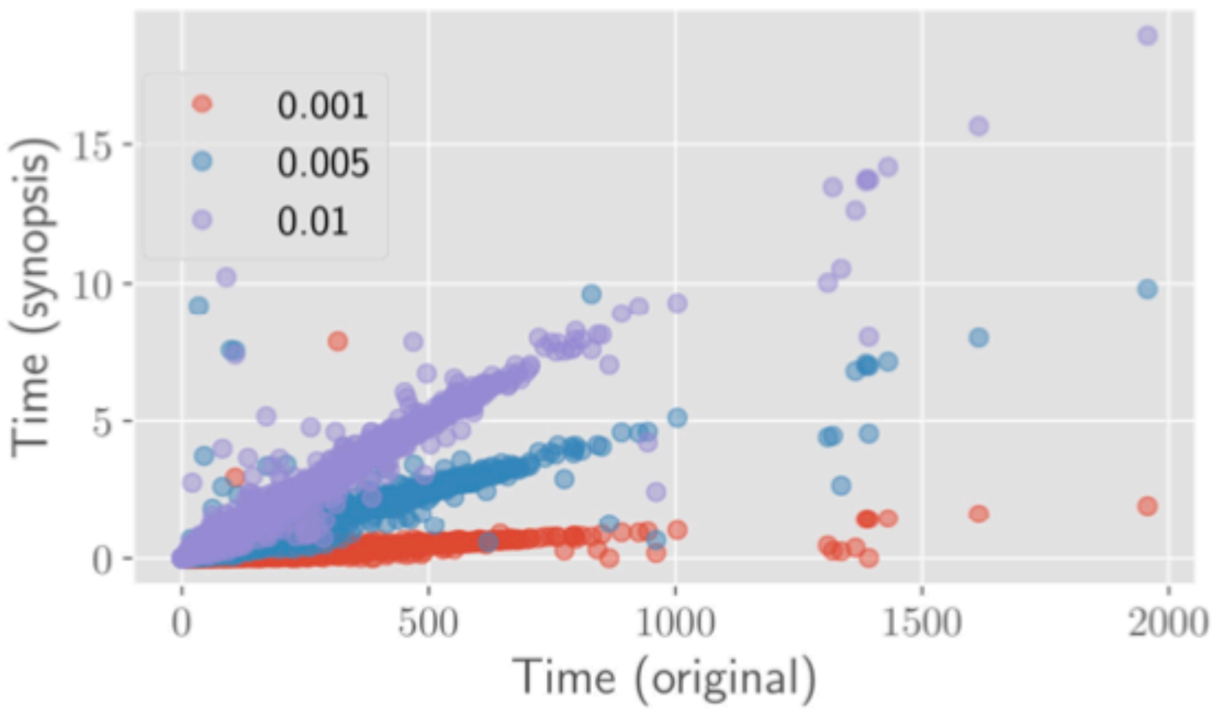
	Full	0.01 Syn		0.01 Total		0.05 Syn		0.05 Total	
AND	54.3	0.64	(85×)	54.94	(+1.2%)	3.22	(17×)	57.52	(+5.9%)
OR	450.0	4.36	(103×)	454.36	(+1.0%)	22.25	(20×)	472.25	(+4.9%)
MaxScore	87.7	0.79	(111×)	88.49	(+0.9%)	4.33	(20×)	92.03	(+5.2%)
Wand	107.4	1.20	(90×)	108.60	(+1.1%)	6.24	(17×)	113.64	(+5.8%)
BMW	77.8	1.21	(65×)	79.01	(+1.6%)	6.15	(13×)	83.95	(+7.9%)

Union & Intersection Estimates Accuracy

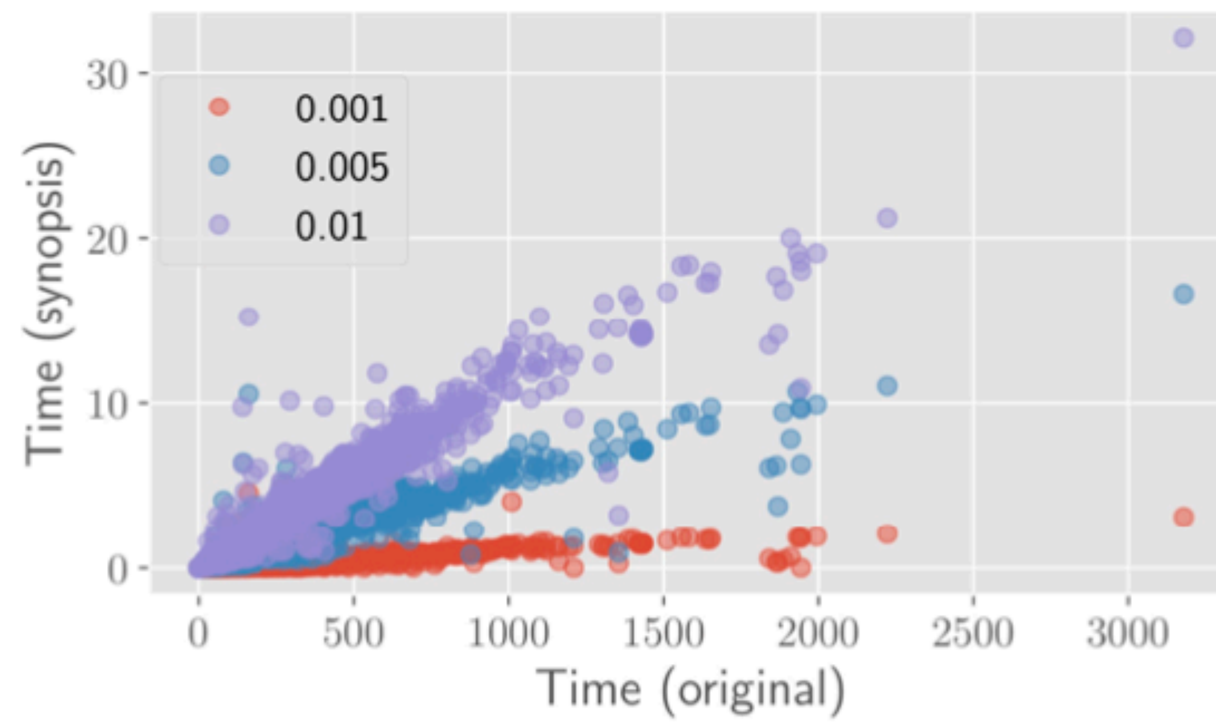


Actual vs. Synopsis Response Times

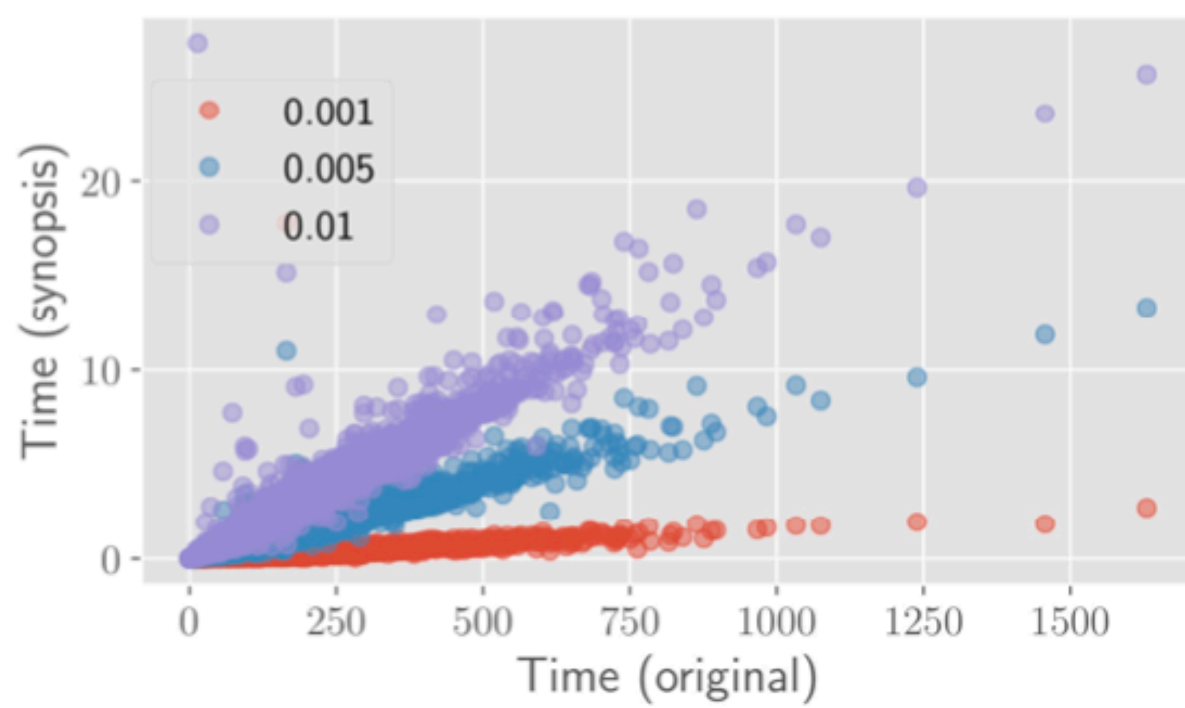
MaxScore



WAND



BMW



Overall Response Time Accuracy

Strategy	MRT	Static RMSE	Dynamic RMSE	Synopsis RMSE			
				0.001	0.005	0.01	0.05
MaxScore (Post)	87.7	37.8	48.7	**37.0**	**25.3**	23.2	23.5
MaxScore (Time)				48.3	26.1	**19.7**	**17.9**
WAND (Post)	107.4	52.3	63.7	**71.4**	62.7	62.2	62.5
WAND (Time)				88.5	**39.5**	**33.0**	**33.0**
BMW (Post)	77.8	30.0	33.8	**65.2**	60.5	60.8	60.2
BMW (Time)				78.1	**20.1**	**17.6**	**15.1**

Long-running Query Classification

	Precision				Recall			
	0.001	0.005	0.01	0.05	0.001	0.005	0.01	0.05
					MaxScore			
Static		89.1				76.0		
Dynamic		89.4				54.5		
Synopsis (Post)	86.1‡	86.0‡	86.9†‡	87.3†‡	**77.2**‡	84.9‡	85.0†‡	85.9†‡
Synopsis (Time)	**96.1**†	**92.9**†‡	**93.9**†‡	**95.4**†‡	46.8†	**91.0**†‡	**95.0**†‡	**94.8**†‡
					WAND			
Static		88.5				75.7		
Dynamic		89.1				57.9		
Synopsis (Post)	**91.7**†	**90.8**†	**90.5**†	**90.9**†	54.0†	57.8†	56.6†	57.4†
Synopsis (Time)	89.7‡	87.6†‡	88.7†‡	87.5†‡	**76.7**‡	**89.9**†‡	**91.5**†‡	**92.5**†‡
					BMW			
Static		81.2				67.7		
Dynamic		83.0				65.5		
Synopsis (Post)	55.4†‡	56.6†‡	56.9†‡	55.1†‡	24.9†‡	29.0†‡	28.0†‡	28.8†‡
Synopsis (Time)	**87.3**†‡	**89.0**†‡	**91.0**†‡	**90.7**†‡	**80.0**†‡	**85.2**†‡	**85.9**†‡	**88.9**†‡

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!