Score-Fitted Indexes and Constant Length Indexes for Information Retrieval

Djoerd Hiemstra
Radboud University
Nijmegen, The Netherlands
djoerd.hiemstra@ru.nl

Abstract

We present two novel inverted index approaches and corresponding query processing strategies for information retrieval: 1) score-fitted indexes and 2) constant length indexes. These indexes do not store document lengths and document priors, but nevertheless support popular rankers like BM25 and language models by approximating their results. We answer the question: What is the effect of score-fitted indexes and constant length indexes, and the combination of both approaches, on the search quality? We show on three diverse datasets that the two indexes perform on par with approaches that use a standard inverted index in almost all cases. Our work suggests that it is possible to develop search engines that are more efficient than engines that store document lengths and/or document priors, such as Lucene and Terrier.

CCS Concepts

• **Information systems → Search engine indexing**.

Keywords

Search Engine Indexing, Search Efficiency, Document Renumbering

ACM Reference Format:
Djoerd Hiemstra. 2025. Score-Fitted Indexes and Constant Length Indexes for Information Retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3726302.3730248

1 Introduction

The inverted file or inverted index [11, 39] is still the backbone of modern search engines, providing search results very efficiently in the first stage of retrieval. An important limitation of many inverted index implementations is that they need to store the document lengths to enable ranked retrieval [39, Section 9]: Ranking algorithms like BM25 [28, 29] and language models (LM) [14, 17], use the document lengths to normalize the term frequencies stored in the inverted index. Lucene, for instance, stores document lengths in "norms" files [21]; Terrier uses a "document" data structure [32]; and CIFF, the common index file format, stores document lengths in its "DocRecord" part [19]. These lists of document lengths are similar to an additional posting list that needs to be accessed for every query, as if every single query has the same additional search term of which the data (the document lengths) need to be preloaded into memory. Similarly, ranking approaches that require a static component, such as PageRank [5] or LM document priors [16], store those in the index too, doubling the data that needs to be preloaded.

The objective of this paper is to develop indexes and approximations of BM25 and LM that do not use the preloaded document lengths and document priors. We investigate two novel indexes that rank documents by approximating document lengths and priors: *1)* the score-fitted index, which reorders documents by their prior or length and then fits a model that approximates the prior or length given the document identifier; and *2)* the constant-length index, that scales all documents such that they are the same length when they are stored in the inverted index. The two approaches can be combined in a single "score-fitted constant length" index, so both the lengths and the priors do not need to be stored anymore.

Sections 3 and 4 describe the two new indexing approaches that are combined in Section 5. The next section provides background information and related work on inverted indexing.

2 Inverted indexing

To process queries efficiently, search engines use a data structure called inverted index in the first retrieval stage. An inverted index assigns to each term t in the collection a list – the postings list – of document numbers d that contain the term, as well as the frequency of occurrence $tf(t,d)$ of the term in that document, tf for short. For each term, the index also stores its document frequency $df(t)$, df for short, which is the number of documents that contain the term [11].

Table 1: Part of an inverted index

t	df	posting list with items <d, tf>
`index`	982345	<6,2> <83,5> <1643,1> ...
`inverted`	69873	<1643,2> <9821,11> <54371,1> ...
`sigir`	2345	<18,17> <321,1> <1643,1> ...
:	:	:

d	1	2	3	4	5	...
doclen	94	543	84	6043	213	...
prior	6	2300	43	87	91	...

Table 1 shows part of an inverted index, in which the term "`index`" occurs in 982,345 documents: Two times in document 6, five times in document 83, etc. Ranking functions need additional information to compute the document scores for a query. BM25 for instance, needs the document length $doclen(d)$, *doclen* for short, to normalize the term frequencies with the following formula [29]:

$$\text{BM25}(d,q) = \sum_{t \in q} \frac{tf \cdot (k_1+1)}{tf + k_1((1-b) + b\frac{doclen}{avgdl})} \log\left(\frac{N+1}{df+0.5}\right), \quad (1)$$

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730248

where *avgdl* and N are the average document length and the number of documents, and $0 \leq b \leq 1$ and $k_1 > 0$ are free parameters. Language modelling approaches use such document normalization too, as well as document priors, for instance as follows [13] (where *sumdf* is the sum of document frequencies *df* over all terms in the index, and $0 \leq \lambda \leq 1$ is a free parameter; document priors do not need to be normalized as probabilities for ranking; we give this model a number, 42, to distinguish it from other LM variants):

$$\text{LM42}(d,q) = \log(prior) + \sum_{t \in q} \log\left(1 + \frac{\lambda \cdot tf \cdot sumdf}{(1-\lambda) \cdot df \cdot doclen}\right) \quad (2)$$

Therefore, as shown at the bottom of Table 1, many practical implementations of inverted indexes store the document lengths (*doclen*) and priors (*prior*) to compute ranking scores [21, 30, 32, 39]. The objective of this paper is to get rid of this second part of the inverted index, while still supporting BM25 and LM42.

One may be tempted to store the normalized floating point values directly in the index, but this makes it impossible to use index compression approaches that work on integers [26, 30]. To support index compression, a lossy, quantized floating point value may be stored instead [1, 7]. The document lengths can be stored lossy in a single byte too (as done by Lucene), but that does not remove the need to store them. Both approximations can be done without loosing search quality [15, 23], a fact that inspired our approaches.

Our first approach uses document re-ordering to re-assign the document numbers in the index, which has been proposed to improve compression [4, 31]. Such a globally ordered index may also improve query efficiency [20, 35–37], We use document re-ordering for a novel goal: scoring, to approximate the document length or prior from the document number.

Our second approach uses an idea from learned sparse retrieval approaches that learn term probabilities using transformer models, and subsequently quantize them into integers to be stored as *tf*s in the inverted index [8, 9, 18, 22, 25, 38]. The document lengths in the resulting index are rather artificial, not related to their actual lengths, removing the need to store them.

3 The score-fitted index

The score-fitted index is constructed as follows. We re-assign the internal document numbers by ordering the documents by their document length or document prior, from high to low. Then we infer a parametric model that assigns an approximate length or prior given the document number.

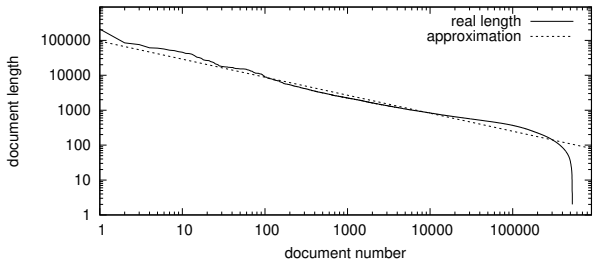

Figure 1: Document lengths for the Robust 2004 collection

Figure 1 shows the lengths of documents of the TREC Robust'04 collection [34] (indexed using porter stemming, removing stop words) plotted against their document number on a log-log scale. The figure shows that a small number of documents are very large, but most document have small lengths. The lengths approximately follow a negative exponential decay function of the form in Equation 3, where a and c are unknown parameters and $doclen(d)$ is the document length of document number d:

$$doclen(d) = c \cdot d^a, \qquad \text{where } a < 0 \quad (3)$$

The function fits the data well, except for documents smaller than about 30 terms, for which we see a sharp drop off. If we take the logarithm on both sides, we get the linear relation between the logarithm of the document length and the logarithm of the document number, as shown in Figure 1 and defined by Equation 4:

$$\log(doclen(d)) = b + a \cdot \log(d), \quad \text{where } a < 0 \wedge b = \log(c) \quad (4)$$

Now, we can fit the slope a and intercept b of the linear model to the data using ordinary least squares linear regression, if we sample the data appropriately. Figure 1 shows a linear fit ($a = -0.51$, $b = 11.3$). if data points are taken from the histogram of unique document lengths. A fit using 8 logarithmically binned data points results in a similar fit ($a = -0.50$, $b = 11.3$). Other sample approaches are reported to work better on synthetic power law distributions [10], but perform similar on our data. The exponential decay function seems a good fit of the document lengths in other collections as well, but with different parameters. The slope a and intercept b on the MS MARCO passage collection [3], which has smaller passages of similar lengths, results in $a = -0.16$, $b = 5.6$ using the histogram approach above. On the Web Track WT10g collection [12], that contains web documents, some of which are very large, we find $a = -0.66$, $b = 14.4$. The WT10g collection is particularly suitable for testing document priors and we explore it further in Section 5.

Table 2: Results of original and score-fitted index

collection	experiment	MAP	nDCG_{10}
Robust'04	BM25 original	0.221	0.448
Robust'04	BM25 fitted	0.219	0.446
Robust'04	LM42 original	0.220	0.424
Robust'04	LM42 fitted	0.219	0.425
MS MARCO	BM25 original	0.307	0.400
MS MARCO	BM25 fitted	0.301	0.405
MS MARCO	LM42 original	0.330	0.408
MS MARCO	LM42 fitted	0.327	0.417
WT10g	BM25 original	0.185	0.276
WT10g	BM25 fitted	0.189	0.287
WT10g	LM42 original	0.174	0.251
WT10g	LM42 fitted	0.173	0.251

We implemented our score-fitted index using the DuckDB full text extension described by Mühleisen et al. [24, 27]. Table 2 shows experimental results of the original index and a score-fitted index that approximates document lengths using Equation 4. The indexes are tested on three collections: Robust'04 on topics 301–450, MS MARCO on the TREC deep learning 2019 topics, and WT10g on the TREC web track adhoc topics 451–500. Indexes are created with

the Porter stemmer, removing stop words. We test the approximate document lengths in two ranking approaches: BM25 with standard parameters $k_1 = 0.9$, $b = 0.4$ [33], and LM42 with standard parameter $\lambda = 0.3$, and *prior* = *doclen*. The results show that the approximate document lengths lead to better or equal $nDCG_{10}$ on two collections for both rankers but slightly worse MAP on two collections for both rankers. In all cases, the differences between rankers (BM25 vs. LM42) are bigger than the differences between the original index and the score-fitted index.[1]

4 The constant length index

An alternative approach for omitting the document lengths (but not necessarily the document priors) is the constant length index which is constructed as follows: We recompute the term frequencies *tf* as shown in Equation 5, inspired by the document length normalization component of the BM25 formula:

$$tf_{\text{new}} = tf \cdot \left((1-\beta) + \beta \frac{constdl}{doclen}\right) \tag{5}$$

Here, *doclen* is again the document length (we omit *d* here for simplicity). Furthermore, the formula has two parameters that we may tune: β, which functions as the parameter b in BM25, and *constdl*, which functions as the average document length. So, Equation 5 rescales all documents to the same length: *constdl*, which by definition now equals the average document length. We assume that the index stores integers only, so tf_{new} is rounded to the nearest integer. The constant length index now replaces all term frequencies *tf* in the original index by tf_{new}. For long documents and small values of *constdl* some new term frequencies tf_{new} will be rounded to zero, and those postings will be omitted from the constant length index. The index does not store the document lengths. To use the index, we replace *doclen* in the BM25 and LM42 ranking formulas by *constdl*. Interestingly, this simplifies the BM25 formula considerably, because its original document length normalization component (given that *constdl* = *avgdl*) can now be omitted. The LM42 formula is also simplified if the document length is used as a prior, because all documents have equal lengths the prior can be omitted.

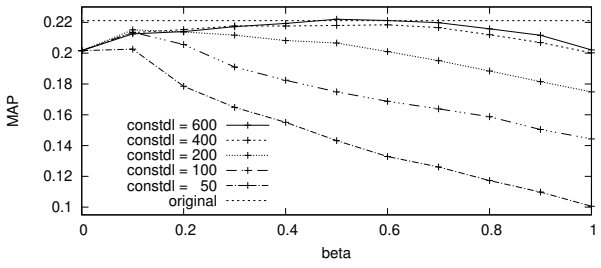

Figure 2: MAP of BM25 on Robust'04 given β and *constdl*

Figure 2 shows the mean average precision (MAP) of BM25 on the constant length index of the Robust'04 collection (stemmed and stopped) for different values of β and *constdl*. The average document length of Robust'04 is 237 terms. If *constdl* is large enough (*constdl* $\geq$ 600), the MAP of the constant length index equals or outperforms the MAP on the original index for $0.5 \leq \beta \leq 0.6$. The dotted line shows the performance of BM25 on the original index for $b = 0.4$. We did not optimize b of BM25 on the original index, even though it leads to small improvements on some collections on the development queries. The LM42 ranker shows similar behaviour: Equal or better performance for *constdl* $\geq$ 600 and $\beta = 0.6$.

Table 3 shows experimental results of the original index and the constant length index on the three collections: Robust'04, MS MARCO and WT10g and two ranking models: BM25 with $k_1 = 0.9$ and $b = 0.4$ (b is needed only for the original BM25) and LM42 with $\lambda = 0.3$. We fine-tuned *constdl* and β on Robust'04 on topics 301–450, MS MARCO on the TREC deep learning 2019 topics, and WT10g on the TREC web track adhoc topics 451–500 (shown as "dev. queries" in Table 3). For MS MARCO, for which the average document length is 28, the optimum values are achieved for *constdl* $\geq$ 50. For WT10g which has an average document length of 315, the optimum is surprisingly at a low value of *constdl* = 100 and $\beta = 0.1$. We did not optimize b of BM25 on the original index; even though optimizing b improves the quality on some collections on the development queries, it also decreases quality on the test queries.

Table 3: Results of the original and constant-length index

		dev. queries		test queries	
collection	experiment	MAP	$nDCG_{10}$	MAP	$nDCG_{10}$
Robust'04	BM25 original	0.221	0.448	0.290	0.418
Robust'04	BM25 constant	0.222	0.449	0.285	0.409
Robust'04	LM42 original	0.220	0.424	0.286	0.400
Robust'04	LM42 constant	0.221	0.441	0.285	0.406
MS MARCO	BM25 original	0.307	0.400	0.320	0.453
MS MARCO	BM25 constant	0.321	0.401	0.325	0.447
MS MARCO	LM42 original	0.330	0.408	0.321	0.423
MS MARCO	LM42 constant	0.323	0.401	0.332	0.427
WT10g	BM25 original	0.185	0.276	0.150	0.244
WT10g	BM25 constant	0.195	0.294	0.145	0.235
WT10g	LM42 original	0.174	0.251	0.120	0.196
WT10g	LM42 constant	0.188	0.283	0.127	0.209

To check how well our values for β and *constdl* generalize, we ran another test on Robust'04 on topics 601–700, MS MARCO on the TREC deep learning 2020 topics, and WT10g on the TREC web track adhoc topics 501–550 (shown as "test queries"). The results show that the constant length index achieves small gains on the test queries for some experiments (on MS MARCO for both BM25 and LM42; and on WT10g for LM42), but also small losses of others (on Robust'04 for both BM25 and LM42; and on WT10g for BM25). Overall, the constant length index seems a viable alternative of the standard indexes that store document lengths.

5 Combining both approaches for web search

Both new indexes can be combined in a single score-fitted constant length index. In this case, the constant length approach approximates the document lengths, and the score-fitting approach approximates the document priors. Such a combined index, which does not store the document lengths nor the priors, is needed for

[1] We removed document lengths to simplify query processing, not to minimize index size (the lengths take about 1% of the size). Surprisingly, sorting the documents by their length reduced the size of the WT10g index to 75% due to better compression; smaller than random sorting and even smaller than URL-based sorting [31].

search tasks where document priors are skewed, but unrelated to the document lengths. Web home page finding [2] is an example of such a task: Longer documents are *not* more likely to be (relevant) home pages. While a document length prior is a good prior for *informational* queries (the queries tested in Sections 3 and 4), it is a bad prior for *navigational* queries, such as the web home page finding queries [6]. For such queries the web graph and URL provide a much better prior probability of relevance.

To test the combined index, we therefore reproduce three home page finding experiments of Kraaij et al. [16]: *1)* LM42 with a document length prior (the original LM42 above), *2)* LM42 with an inlink prior, which assumes that documents with more incoming hyperlinks are more likely to be relevant, and *3)* LM42 with a URL prior, which assumes that documents with shorter URLs are more likely to be relevant, distinguishing four types of URLs: root, subroot, path and file. The URL prior for each URL type is determined from the query-relevance judgments of a set of training queries by counting the number of relevant documents (for any training query) and divide that by the number of documents of that type [16]. Those estimates are used on test queries (so the only real training is done for the URL prior runs). We report experimental results on a set of 100 home page finding training queries provided by Craswell [12], and on the official TREC home page finding test queries (topics EP1–EP145). Following the TREC home page finding task, we report the mean reciprocal rank (MRR). In most cases when there is only a single relevant document per query (the home page), the reciprocal rank equals the average precision, so MRR equals MAP. Subsequently, we repeat each of the three experiments using: *1)* the contant length index, *2)* the score-fitted index, and *3)* the constant-length-score-fitted index.

Table 4: Results of WT10g home page task

experiment	MRR train	MRR test
BM25 original	0.236	0.277
LM42 length prior (original)	0.283	0.273
LM42 inlink prior	0.391	0.442
LM42 URL prior	0.756	0.722
LM42 constant, length prior	0.283	0.265
LM42 constant, inlink prior	0.390	0.442
LM42 constant, URL prior	0.760	0.719
LM42 fitted length prior	0.287	0.272
LM42 fitted inlink prior	0.392	0.435
LM42 fitted URL prior	0.592	0.540
LM42 fitted combined prior	0.708	0.674
LM42 constant & fitted length prior	0.286	0.272
LM42 constant & fitted inlink prior	0.390	0.429
LM42 constant & fitted URL prior	0.588	0.539
LM42 constant & fitted combined prior	0.710	0.672

Table 4 shows all experimental results in a single table. The three reproduced experiments show a substantial improvement of the MRR for the LM42 inlink prior over the LM42 length prior, and a remarkable improvement of the LM42 URL prior over both (MRR 0.722 vs. 0.273 and 0.442), confirming the results of Kraaij et al. [16]. Our results on all three experiments are slightly worse than those of Kraaij et al., possibly because our indexer did not remove HTML boiler plate. For completeness, we also report the BM25 results, which are on par with the LM42 length prior run.

For the constant length index, ("LM42 constant" with length prior, inlink prior, or URL prior in Table 4), the best results on the train queries were obtained for $constdl = 400$ and $\beta = 1.0$ for all three priors. Note the following difference with the experiments in Section 4: In Section 4, we removed the length prior (because document lengths are constant), but in this experiment we retained the document prior for all three experiments (also the length prior), because we treat document lengths and document priors as two separate data structures that we remove from the index. Interestingly, if we retain the prior, $\beta = 1.0$ gives the best performance, which seems to be in line with the theory behind LM42 (the β parameter was inspired by BM25). Experimental results in Table 4 show only minor differences between the constant length index and the original index, showing that the constant length index approximates the original index well.

The third part of Table 4, reports the score-fitted index that removes the document priors, while keeping the document lengths (unlike in Section 3 where the fitted lengths approximate both the length prior and the document lengths). The score-fitted index performs equally well on the length prior and the inlink prior, but worse on the URL prior (0.540 vs. 0.722 MRR). Clearly, the URL prior, which only uses four unique values for each URL type, cannot be approximated well by the log-linear model, which determines a unique prior for every document. We experimented with several ways to combine the URL-, inlink-, and length priors, including simple linear combinations and a naive Bayes approach [16, Section 3.2]. Table 4 reports an even simpler "LM42 fitted combined prior" approach: It sorts by URL type, inlink, and document length, respectively (so documents with the same URL type are sorted by inlinks, etc.), before fitting a linear model on 8 logarithmic bins using the training query-relevance judgments. This score-fitted combined prior achieves much better performance, but still not on par with the URL prior on the original index (0.674 vs. 0.722 MRR).

Finally, we combined the constant length and score-fitted approaches in one combined index that does not store document lengths nor priors. By applying the constant length approach on the score-fitted index, the quality remains approximately the same (as it did when applied to the original index).

6 Conclusion

We show that the constant length index removes the storage of document lengths from an inverted index, while retaining the quality of BM25 and LM42 ranking. We also show that the score-fitted index removes document priors from the index for the web home page task, while retaining the quality of a length prior and an inlink prior, but with a minor loss of quality for the original URL prior when compared to a score-fitted combined URL-inlink-length prior. This may be remedied in subsequent work by smartly combining multiple priors. Future work should determine if omitting document lengths and priors from the index will result in substantial efficiency gains. Our code is available from: https://gitlab.science.ru.nl/informagus/zoekeend.

Acknowledgments

Thanks to the EU project OpenWebSearch.eu (GA 101070014).

References

[1] Vo Ngoc Anh, Owen de Kretser, and Alistair Moffat. 2001. Vector-space ranking with effective early termination. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 35–42.

[2] Peter Bailey, Nick Craswell, and David Hawking. 2003. Engineering a multi-purpose test collection for web retrieval experiments. *Information Processing & Management* 39, 6 (2003), 853–871.

[3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv* arXiv:1611.09268 (2016).

[4] Dan Blandford and Guy Blelloch. 2002. Index compression through document reordering. In *Proceedings of the Data Compression Conference (DCC)*. IEEE, 342–351.

[5] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30, 1-7 (1998), 107–117.

[6] Andrei Broder. 2002. A taxonomy of web search. In *ACM Sigir forum*, Vol. 36. ACM, 3–10.

[7] Matt Crane, Andrew Trotman, and Richard O'Keefe. 2013. Maintaining discriminatory power in quantized indexes. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management (CIKM)*. 1221–1224.

[8] Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 1533–1536.

[9] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2288–2292.

[10] Michel Goldstein, Steven Morris, and Gary Yen. 2004. Problems with fitting to the power-law distribution. *European Physical Journal B* 41 (2004), 255–258.

[11] Donna Harman, Edward Fox, Ricardo Baeza-Yates, and W. Lee. 1992. Inverted files. In *Information Retrieval: Data Structures and Algorithms*, William B. Frakes and Baeza-Yates Ricardo (Eds.). Chapter 3.

[12] David Hawking and Nick Craswell. 2001. Overview of the TREC-2001 Web Track. In *Proceedings of the 10th Text Retrieval Conference (TREC)*. 61–67.

[13] Djoerd Hiemstra. 2000. A probabilistic justification for using tf.idf term weighting in information retrieval. *International Journal on Digital Libraries* 3 (2000), 131–139.

[14] Djoerd Hiemstra and Wessel Kraaij. 1999. Twenty-One at TREC-7: Ad-hoc and cross-language track. In *Proceedings of the 7th Text REtrieval Conference (TREC)*. NIST, 227–238.

[15] Chris Kamphuis, Arjen De Vries, Leonid Boytsov, and Jimmy Lin. 2020. Which BM25 do you mean? A large-scale reproducibility study of scoring variants. In *Advances in Information Retrieval: 42nd European Conference on Information Retrieval (ECIR)*. 28–34.

[16] Wessel Kraaij, Thijs Westerveld, and Djoerd Hiemstra. 2002. The importance of prior probabilities for entry page search. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in Information Retrieval*. ACM, 27–34.

[17] John Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 111–119.

[18] Jimmy Lin and Xueguang Ma. 2021. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. *arXiv preprint arXiv:2106.14807* (2021).

[19] Jimmy Lin, Joel Mackenzie, Chris Kamphuis, Craig Macdonald, Antonio Mallia, Michał Siedlaczek, Andrew Trotman, and Arjen de Vries. 2020. Supporting interoperability between open-source search engines with the common index file format. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2149–2152.

[20] Xiaohui Long and Torsten Suel. 2003. Optimized query execution in large search engines with global page ordering. In *Proceedings of the Very Large Databases Conference (VLDB)*. 129–140.

[21] Lucene. 2024. Apache Lucene Index File Formats. https://lucene.apache.org/core/10_0_0/core/org/apache/lucene/codecs/lucene100/package-summary.html.

[22] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1723–1727.

[23] Alistair Moffat, Justin Zobel, and Ron Sacks-Davis. 1994. Memory efficient ranking. *Information Processing & Management* 30, 6 (1994), 733–744.

[24] Hannes Mühleisen, Thaer Samar, Jimmy Lin, and Arjen De Vries. 2014. Old dogs are great at new tricks: Column stores for IR prototyping. In *Proceedings of the 37th international ACM SIGIR conference on research & development in Information Retrieval*. ACM, 863–866.

[25] Thong Nguyen, Sean MacAvaney, and Andrew Yates. 2023. A unified framework for learned sparse retrieval. In *European Conference on Information Retrieval (ECIR)*. 101–116.

[26] Giulio Ermanno Pibiri and Rossano Venturini. 2020. Techniques for inverted index compression. *ACM Computing Surveys (CSUR)* 53, 6 (2020), 1–36.

[27] Mark Raasveldt and Hannes Mühleisen. 2019. Duckdb: an embeddable analytical database. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*. ACM, 1981–1984.

[28] Stephen Robertson, Stephen Walker, and Micheline Hancock-Beaulieu. 1994. Okapi at TREC-3. In *Proceedings of the 3rd Text Retrieval Conference (TREC)*. 109–126.

[29] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[30] Falk Scholer, Hugh E Williams, John Yiannis, and Justin Zobel. 2002. Compression of inverted indexes for fast query evaluation. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in Information Retrieval*. ACM, 222–229.

[31] Fabrizio Silvestri. 2007. Sorting out the document identifier assignment problem. In *European conference on information retrieval (ECIR)*. Springer, 101–112.

[32] Terrier. 2024. Accessing Terrier's Index API. https://pyterrier.readthedocs.io/en/latest/terrier-index-api.html.

[33] Andrew Trotman, Xiangfei Jia, and Matt Crane. 2012. Towards an Efficient and Effective Search Enginex. In *Proceedings of the Open Source IR workshop (OSIR)*. 40–47.

[34] Ellen Voorhees. 2005. The TREC robust retrieval track. *ACM SIGIR Forum* 39, 1 (2005), 11–20.

[35] Qi Wang and Torsten Suel. 2019. Document reordering for faster intersection. *Proceedings of the VLDB Endowment* 12, 5 (2019), 475–487.

[36] Erman Yafay and Ismail Sengor Altingovde. 2023. Faster Dynamic Pruning via Reordering of Documents in Inverted Indexes. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2001–2005.

[37] Mingjie Zhu, Shuming Shi, Mingjing Li, and Ji-Rong Wen. 2007. Effective top-k computation in retrieving structured documents with term-proximity support. In *Proceedings of the sixteenth ACM Conference on Information and Knowledge Management (CIKM)*. 771–780.

[38] Shengyao Zhuang and Guido Zuccon. 2021. TILDE: Term independent likelihood model for passage re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1483–1492.

[39] Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM computing surveys (CSUR)* 38, 2 (2006).