Efficiency Comparison of Document Matching Techniques

Patrice Lacour, Craig Macdonald, and Iadh Ounis

Department of Computing Science,
University of Glasgow, Glasgow, G12 8QQ, UK.
{lacourp,craigm,ounis}@dcs.gla.ac.uk

Abstract. Inverted indices are one of the most commonly used techniques to search very large document collections. While the typical size of web document collections is constantly increasing, users have come to expect a very quick response time, and accurate search results. Hence, to make best use of available hardware resources, efficient and effective retrieval techniques are desirable. We review several state-of-the-art approaches for matching documents to query terms, based on term-centric and document-centric scoring. We test the techniques using three modern Web Information Retrieval (IR) test collections, and conclude in terms of the trade-off between retrieval effectiveness and efficiency.

1 Introduction

Inverted indices are one of the most commonly used techniques to search very large document collections. Usually, these contain for each term a stream of *posting*s, i.e. the id of every document that the term occurs in, and the frequency of the term in that document [18]. Other information may also be stored in the posting list, such as term position information, or the HTML tags where the term occurs within the document [3]. In this paper, we only consider a simple inverted index containing the document identifier and term frequency postings.

The most common method for scoring documents retrieved in response to a query (when using a bag-of-words retrieval approach) is to score each occurrence of a query term in a document using the information contained in its corresponding posting list in the inverted file, and combining these scores for each document. However, for terms with low discriminatory power, then every document the term occurs in must be scored, leading to high retrieval time without benefit to retrieval effectiveness.

While parallelised retrieval can mitigate the cost of high retrieval time, three other matching approaches exist to reduce retrieval times, by trading off with the overall effectiveness of system: Firstly, the pruning of low value documents or terms from the inverted indices [2, 14]; secondly, ordering inverted indices postings by their impact on retrieval [12]; thirdly avoiding scoring all occurrences of the query terms. The focus of this work is to assess various techniques for efficient matching in the latter case. Broadly, these efficient matching techniques

fall into two categories - scoring by each term, or scoring by documents - known as Term-at-a-time (TAAT) and Document-at-a-time (DAAT), respectively.

We experiment using these techniques, and integrate them into an existing IR platform. The platform provides a classical TAAT approach where every term occurrence for each query term is evaluated, and we apply this as the baseline in our experiments. Moreover, several DAAT approaches utilise an improved inverted index format, which allows the decompression of documents that are unlikely to be relevant to be skipped - indeed we experiment with two skipping strategies. All our experiments are conducted in a uniform experimental setting, consisting of several standard Web IR test collections of varying size. We conclude in terms of retrieval effectiveness, retrieval efficiency, and for the inverted index skipping techniques, the additional disk space required by the skipping structures. The contributions of this paper are two-fold: firstly, an in-depth study of the efficiency versus the effectiveness of various matching techniques, and, secondly, to relate the findings to the statistics of the applied test collections. Our work differs from that of other studies such as that of Cambazoglu & Aykanat [7], in that both efficiency *and* effectiveness measures are examined, and trade-off conclusions made.

The remainder of this paper is structured as follows: Section 2 briefly reviews various matching approaches; Section 3 describes several inverted index formats supporting efficient skipping; Section 4 details our experimental setting; and results are provided in Section 5. We provide concluding remarks in Section 6.

2 Matching Techniques

We now describe the matching techniques we experiment with in this paper. As mentioned in the introduction, these fall into two categories - namely TAAT and DAAT. One of the problems with TAAT approaches is that a larger number of documents are scored, requiring more memory to accumulate their scores (known as accumulators [7]). In contrast, DAAT techniques require less memory, as less documents should be scored overall.

However, in all cases except for Full TAAT (the baseline), a trade-off is made by recognising that most Web retrieval tasks favour high-precision retrieval, and aiming to retain high precision effectiveness at the system's top-ranked documents, while degrading retrieval effectiveness at lower ranks. This is typically achieved by predicting which terms have the least impact on the retrieved documents and ignoring portions of their posting lists not likely to affect the top-ranked documents.

Full TAAT. This is our baseline, a TAAT approach, where we compute the score of all postings for every query term. This is in contrast to the other approaches, which aim to avoid some scoring computations, this technique exhibits no degradation in the effectiveness of the ranking.

Turtle TAAT. This technique is a TAAT approach developed by Turtle & Flood in [17]. Firstly, the query terms are ordered by their maximum possible impact on the retrieval score (called the *term upper bounds*). Then, the scoring

of occurrences is performed in two phases: During the first phase, all postings are evaluated for each query term. At the end of the scoring of each query term, if the minimum score of the current top-ranked R documents is greater than the sum of the upper bounds of the query terms remaining to be scored, then the first phase scoring is terminated, and we proceed to the second phase. In this second phase, only the documents which have already been scored are computed for the remaining query terms.

This technique ensures that the top R documents are correctly ordered compared to a Full TAAT, but without the need to fully evaluate all the query terms, leading to document relevance scores that are different from the Full TAAT, but with broadly similar rankings.

Moffat TAAT. The idea for this approach introduced by Moffat and Zobel in [10] is also to score the most important query terms first. However, when K documents have been scored, the matching enters a second phase, where no more documents are retrieved, and the K documents retrieved thus far are fully scored. K is typically equal to 0.2% of the number of documents in the collection. Note that Moffat and Zobel presented two different heuristics, called 'Quit' and 'Continue' to determine when the first phrase scoring should be terminated. In the 'Quit' approach, the condition of accumulators reaching size K is checked for every posting scored, while for the 'Continue' approach, the condition is checked only after each query term has been scored. In this paper, we only experiment using the 'Continue' approach as this has less efficiency overheads, but may result in scoring far more than K documents if a term posting list is very long.

Note also that in the second phase of Turtle TAAT and Moffat TAAT matching approaches, it is possible to omit large numbers of postings for each query term. In this scenario, it is advantageous for the inverted index implementation to support a *next(docid)* operation. In the following section, we review several inverted index skipping techniques.

Lester et al. [9] describe adaptive improvements to the Moffat TAAT approach in the context of Web-style queries. We do not experiment with this approach, as we wish to experiment with generic algorithms without specific assumptions.

Turtle DAAT. This DAAT technique was also developed by Turtle & Flood in [17]. In a DAAT technique, the postings lists for all query terms are read concurrently. Similar to Turtle TAAT, the term upper bounds are computed. Then all of the ith postings for each query term are evaluated in turn. When a document is scored, we add to the actual score of the document, the sum of the upper bound of the remaining terms to be scored. If this sum is less than the minimum score of the current Rth-ranked document, then we do not calculate the term score of the document, as the document could not be in the top R documents.

It is of note that Turtle & Flood [17] suggest that their DAAT technique could be improved by the use of inverted index posting skipping, however it is not clear from their description how this would be applied. Turtle & Flood's methods are known as MAX-SCORE (MAX-SCORE TAAT and DAAT). Moreover,

Stronham et al. [15]. revisit the MAX-SCORE approaches for DAAT, suggesting heuristics to determine the top documents for a given query term, pre-scoring these documents, and using these pre-scored documents as the threshold for the MAX-SCORE algorithm. However, in contrast to the previously described approaches, their approach requires that posting lists be sorted by decreasing tf, not ascending docid.

3 Skipping Inverted Index Postings

Typically query term scoring in an IR system is disk IO intensive, however, for larger Web-scale settings with many machines available, posting lists may be kept in the main system memory or in the system cache [16]. In both cases, compression is useful to reduce IO in favour of slight compression overheads [6]. Some matching methods described above (in particular Moffat TAAT) can potentially avoid scoring some postings of the query terms. Therefore, if this skipping could reduce the IO operations, then the retrieval time can be reduced. Skipping techniques on compressed inverted indices were first described by Moffat & Zobel in [10]. In this work, we review both Moffat & Zobel's approach as well as the later approach proposed by Boldi & Vigna in [3].

Posting compression in inverted indices can be performed in several fashions. One compression technique that can be applied is the use of Elias-Unary and Elias-Gamma encoding [8]. While other techniques such as variable-byte encoding exist (and may decode more efficiently), in this work we concentrate only on these Unary and Gamma encodings.

As a given posting can then be of variable length, to efficiently skip over postings in the posting list, additional pointers have to be embedded in the posting list describing a potential skip when decoding. Unsurprisingly, these additional pointers cause the size of the inverted index to grow.

Assume that the inverted index postings are grouped into blocks (of $p \geq 3$ postings). A pointer allows one or more blocks to be skipped, ensuring that the next posting to be read has a document id greater than or equal to a required document. Skipping is best illustrated following Moffat & Zobel [10]:

In a normal inverted index for a given term, a posting list of ⟨docid,tf⟩ tuples is stored as follows:

$$\langle 5,1\rangle\ \langle 8,1\rangle\ \langle 12,2\rangle\ \langle 13,3\rangle\ \langle 15,1\rangle\langle 18,1\rangle\langle 23,2\rangle\langle 28,1\rangle\langle 29,1\rangle\langle 32,3\rangle$$

We insert the pointers ⟨⟨docid, bit address⟩⟩ every p pointers (say 3 in our example), where the docid in the pointer is the first docid in the first posting of the next block, and a_i is the pointer to the next block. Note that the first pointer is an exception, as a_0 is not required.

$$\langle\langle 5,a_0\rangle\rangle\ \langle\langle 13,a_1\rangle\rangle\ \langle 5,1\rangle\langle 8,1\rangle\langle 12,2\rangle\ \langle\langle 23,a_2\rangle\rangle\ \langle 13,3\rangle\langle 15,1\rangle\langle 18,1\rangle\ \langle\langle 32,a_3\rangle\rangle\ \langle 23,2\rangle\langle 28,1\rangle\langle 29,1\rangle$$

We can compress further the data by removing redundant document ids and by applying delta gaps between pointers.

$$\langle\langle 5,a_0\rangle\rangle\ \langle\langle 8,a_1\rangle\rangle\ \langle 1\rangle\langle 3,1\rangle\langle 1,2\rangle\ \langle\langle 9,a_2 - a1\rangle\rangle\ \langle 3\rangle\langle 2,1\rangle\langle 3,1\rangle\ \langle\langle 9,a_3 - a_2\rangle\rangle\ \langle 2\rangle\langle 5,1\rangle\langle 1,1\rangle$$

This is a common approach of all of the skipping approaches. It allows us skipping forward by an entire block, without decoding the information encoded in the block.

From [3] & [10], we note three approaches. These differ in the way that they determine the size of a block of postings p. Moreover, two approaches contain multiple levels of pointers, allowing effective skipping of several blocks in one disk seek.

Single Moffat: In this approach, the size of a posting block p for term t is set as follows:

$$\frac{p = \sqrt{LN_t}}{2} \tag{1}$$

where N_t is the number of documents that term t occurs in, and L is a free parameter. Moffat & Zobel suggest three values of L, namely 10, 100 and 1000, depending on statistics of the collection and of the query set [10].

Multi Moffat: In this method, the size of the skip p_i of each level i of term t is determined as a function of the skipping parameter L as follows:

$$p_i = \frac{1}{2} L^{\frac{i}{h+1}} p^{\frac{h-i+1}{h+1}} \tag{2}$$

where h is the number of levels to be computed for this term, calculated as:

$$h = (log_e \frac{N_t}{L}) - 1 \tag{3}$$

Multi Boldi: The lowest level has a block of L (32 or 64) postings, and each higher level is composed of 2 lower ones [3]. Moreover, when a posting list does not have exactly an exact multiple of L postings, extra unconnected pointers can exist. In this approach they are removed, to prevent posting lists being unnecessary large.

In the following section, we experiment with the Single Moffat and Multi Boldi skipping techniques, as these are two representatives of the single and multiple skipping level classes of techniques. Experiments using Multi Moffat remain as future work.

4 Experimental Setting

In the following, we experiment with the reviewed matching techniques and inverted index skipping techniques. To test the matching methods, we use three different Web IR test collections, related to different domains and timescales. The collections we experiment with are: two TREC Web test collections WT2G and WT10G, which are medium-scale general Web crawls from early 1997; .GOV2 is a more recent and much larger TREC Web test collection - a crawl of the .gov domain of the Web from 2003. Statistics from the collections are given in Table 1, as well as the TREC topic sets applied. Note that, as expected, the average length of the term postings increases as the collection size grows. This infers that for larger collections, there is more potential improvements for

Collection	#Docs	Avg Doc Len	#Terms	Avg Posting Len	Topics
WT2G	247,491	645.3	1,002,586	62.7	401-450
WT10G	1,692,096	399	3,140,838	89.1	451-500
GOV2	25,205,179	652.4	15,466,363	304.4	801-850

Table 1. Statistics of the Web IR test collections applied.

increasing the retrieval speed when skipping can be used. Finally, we experiment with short (title-only) queries and long (title, description and narrative) queries from the used test collections, to assess whether the topic length has an impact on the efficiency or effectiveness retrieval performances[1].

In most retrieval tasks, the user is interested in the accuracy of the first page of documents retrieved. For this reason, in all methods, we set R, the number of top-ranked documents which should be fully scored, to $R = 20$.

We use the Terrier IR platform [11] in our experiments. The collections are indexed, removing standard English stopwords, and applying Porter's English stemmer. For retrieval we use the BM11 document weighting model (as this is similar to the TF-IDF which was applied in the original papers on DAAT and TAAT matching approaches) [13], as follows:

$$score(d, Q) = \sum_{t \in Q} qtw \cdot \frac{tf \cdot k_1}{tf + k_1 \cdot (\frac{1-b+b \cdot l}{avg\_l})} \cdot log_2 \frac{N}{\frac{N}{N_t+1}} \quad (4)$$

where tf is the frequency of query term t in document d, l is the length of document d, avg_l is the average length of all documents in the collection, N is the number of documents in the collection, and N_t is the number of documents containing an occurrence of term t. k_1 and b are parameters, for which we use the default settings $k = 1.2$ and $b = 0.75$.

Note that BM11 includes a document length normalisation component, to ensure a fair retrieval between long and short documents. By applying normalisation, the frequency of a term tf is normalised by the length of the document l, with respect to the average document length avg_l. However, this normalisation can possibly hinder the correct calculation of the maximum score a term in a document can achieve (the term upper bound), depending on whether the posting with largest tf also has smallest document length. To simplify, we approximate document length with average document length in the collection when calculating term upper bounds using BM11, as this avoids pre-scoring every posting for every term in the collection, in order to obtain exact upper bounds. However, since tf can exhibit strong correlations with l [1], average document length may not be a good approximation. A more accurate approximation of the term upper bounds is an appropriate future work direction.

For experiments where timing is recorded, these were carried out using a dedicated Intel PIV 2GHz single CPU machine, with 512MB of RAM, running Linux. The file-store is a RAID array attached to a Linux server by 20MB/sec Wide SCSI. Machines are inter-connected by 100MBPs network.

[1] Long queries on GOV2 are omitted.

Matching	Short			Long		
	#Term Scorings	P@20	MAP	#Term Scorings	P@20	MAP
WT2G						
Full TAAT	24,926	0.3650	0.2613	498,966	0.3620	0.2784
Turtle TAAT	7,202	0.2411	0.1788	253,991	0.1980	0.1262
Turtle DAAT	23,649	0.3230	0.1300	253,911	0.0350	0.0059
Moffat TAAT	6,855	0.3460	0.2350	16,142	0.1560	0.0857
WT10G						
Full TAAT	183,607	0.2030	0.1880	2,039,356	0.2810	0.2326
Turtle TAAT	29,553	0.1740	0.1295	740,658	0.1440	0.1043
Turtle DAAT	163,371	0.1690	0.1054	1,253,580	0.0290	0.0090
Moffat TAAT	27,845	0.1870	0.1793	49,860	0.1530	0.0803
GOV2						
Full TAAT	2,221,422	0.2840	0.2244	-	-	-
Turtle TAAT	692,921	0.2360	0.1402	-	-	-
Turtle DAAT	2,003,572	0.2190	0.1134	-	-	-
Moffat TAAT	628,538	0.2330	0.1320	-	-	-

Table 2. Efficiency and effectiveness results applying various matching techniques to the three Web test collections.

5 Experimental Results

Table 2 presents the experimental results for the tested matching approaches. Effectiveness is measured using Mean Average Precision (MAP) and Precision at rank 20 (P@20). P@20 is a useful measure as it represents the accuracy of the first screen of search results presented to a search engine user [5]. Efficiency is measured by the average number of query term-document scorings per query. The results presented for all methods do not use any additional inverted index structure for skipping postings, instead emulating skipping by simply iterating through the postings until the required document is found.

From the result in Table 2, we note firstly that all three 'efficient techniques' (i.e. Turtle TAAT, Turtle DAAT and Moffat TAAT) can reduce the number of term occurrence scorings compared to the Full TAAT (our baseline matching technique). However, such increase in efficiency has a corresponding trade-off in retrieval effectiveness - this is typically more marked for MAP than P@20. This is expected, as all the efficient techniques aim to get the top ranked documents correct while trading off overall retrieval performance.

Comparing the techniques for short queries, it is noticeable that Moffat TAAT matching usually achieves the lowest overall drop in P@20 and MAP. This is surprising, given that far less term occurrences are scored when compared to Turtle DAAT, which has slightly lower MAP and P@20 scores. Turtle TAAT is more comparable to Moffat TAAT in terms of number of scorings, but less so in terms of retrieval performance.

Examining the long queries, it is noticeable that all of the efficient matching techniques perform less well here than on short queries. This contrasts from the baseline matching technique, which usually increases in effectiveness (MAP and

P@20). In particular, Turtle DAAT exhibits very low effectiveness. We believe that the low performance of the efficient techniques for long queries is because each approach tries to ascertain the impact of each query term in the retrieval process. Indeed, for Turtle DAAT, the order of term scoring may be predicted incorrectly, and hence the more important query terms occurring in the title of the topics may not be scored. It is of note that the collection originally tested by Turtle & Flood was much smaller than WT2G and WT10G (11K docs vs. 200K docs and 1.6 million docs respectively), and only medium length queries (on average, 9.1 terms each) were used [17].

Comparing techniques across collections, it is noticeable that the efficiency benefit of applying efficient techniques grows for larger collections, as more term occurrences can be omitted from the scoring. Indeed, for GOV2 with short queries, Moffat TAAT requires only 72% of term occurrences scorings compared to the Full TAAT baseline for a 17% drop in P@20.

Overall, as expected, the techniques reviewed here can be applied to speed up retrieval, if some degradations in retrieval effectiveness can be tolerated. Of all the techniques, Moffat appears the most promising in reducing term occurrence scorings while minimising the corresponding reduction in retrieval performance.

Table 3 presents a comparison of the time taken to perform retrieval for short and long queries on the the WT2G and WT10G collections using the Turtle TAAT and Moffat TAAT approaches. Using these efficient matching approaches, we compare the efficiency of two inverted index skipping approaches reviewed in Section 3. The baseline inverted index does not skip any postings.

From Table 3, we firstly note that (unsurprisingly) index sizes are increased by the inclusion of skipping pointers to the inverted index. For some collections, this can be as high as a 46% increase (see WT2G, Single Moffat $L = 1000$). However, as collection size increases, the overhead in index size associated with the skipping pointers decreases (down to 23% increase for $L = 1000$ on WT10G).

In terms of retrieval time, we found that most inverted index skipping settings improved the average query time, particularly on WT2G. On this collection, the retrieval efficiency is benefited most by Moffat TAAT, particularly with Single Moffat skipping. For WT10G, there is no large benefits in applying skipping techniques, and the most benefit is shown when using Multi Boldi (with minimum skip size $L = 32$) for short queries. Comparing short and long queries in general, the differences in retrieval times are largely correlated with the number of term occurrence scorings reported in Table 2 above. Overall, we conclude that the application of the skipping methods only benefited retrieval time on WT2G. For WT10G, it appears that the overheads in decoding the more complicated inverted index format outweighs the benefit in reducing the disk IO by skipping.

While the overall timing statistics may change if other compression techniques (e.g. variable-byte encoding) had been applied, the conclusions would not change, as the compression methods are not varied and the comparisons in Table 3 are fair.

	No Skipping	Single Moffat			Multi Boldi	
$L =$	-	10	100	1000	32	64
WT2G						
SQ Turtle TAAT	0.0899	0.0804	0.0803	0.0806	0.0804	0.0803
LQ Turtle TAAT	0.549	0.438	0.440	0.438	0.612	0.438
SQ Moffat TAAT	0.0962	0.0829	0.0831	0.0840	0.0841	0.0841
LQ Moffat TAAT	0.202	0.154	0.159	0.187	0.222	0.176
Index Size	100%	122%	136%	146%	110%	105.8%
WT10G						
SQ Turtle TAAT	0.304	0.312	0.315	0.307	0.277	0.409
LQ Turtle TAAT	2.271	2.227	2.244	2.314	2.275	2.240
SQ Moffat TAAT	0.304	0.335	0.327	0.342	0.293	0.319
LQ Moffat TAAT	0.691	0.819	0.839	0.899	0.879	0.800
Index Size	100%	111%	115%	123%	107.8%	104.3%

Table 3. Mean query time (seconds) using the Moffat TAAT and Turtle TAAT matching techniques for the various collections and inverted index skipping techniques. SQ and LQ denote short and long queries, respectively. Index sizes are also reported; L is the inverted index skipping parameter. Experiments on GOV2 are omitted.

6 Conclusions

In this paper, we reviewed several approaches for efficiently matching and scoring documents in response to queries, and evaluated their efficiency and effectiveness on three standard Web IR test collections. We found that, compared to a full TAAT scoring baseline, these approaches could markedly reduce the number of term occurrence scorings, but at the cost of reduced overall effectiveness (MAP). However, high precision accuracy was usually maintained. There is anecdotal evidence in this paper that the efficient matching techniques can give larger benefits in retrieval performance for larger collections. However, improvements in applying inverted index skipping techniques did not appear to scale to a larger collection. In the future, we intend to investigate how efficiency changes when using larger test collections such as UK-2006, and comparing to the systems submitted to the Efficiency task in the TREC 2004-2006 Terabyte tracks [5]. In particular, it is probably more useful to evaluate effectiveness using 50 assessed queries, but evaluating efficiency using larger number of queries, to gain more accurate timing measurements, as performed in the TREC Terabyte track Efficiency tasks.

Other future investigations will include study into other techniques for reducing the number of term occurrences scored, such as the newer versions of Turtle TAAT [9], MAX-SCORE DAAT [15], and finally the WAND iterator proposed by Broder et al. in [4]. We would also like to understand more fully the relationship between retrieval performance and the R parameter (which controls the number of documents that are scored and retrieved), especially with respect to the application of document priors commonly required for Web search tasks such as home-page and named-page finding tasks.

References

1. Amati G.: *Probabilistic Models for Information Retrieval based on Divergence from Randomness*. PhD thesis, Department of Computing Science, University of Glasgow, 2003.
2. Blanco R., Barreiro A.: Static Pruning of Terms in Inverted Files. In *Proceedings of ECIR 2007*: (2007) 64–75
3. Boldi, P., Vigna, S.: Compressed Perfect Embedded Skip Lists for Quick Inverted-Index Lookups. In *Proceedings of SPIRE 2005* LNCS 3772 (2005) 25–28
4. Broder A., Carmel D., Herscovici M., Soffer A., Zien J.: Efficient Query Evaluation using a Two-Level Retrieval Process. In *Proceedings of CIKM 2003* (2003) 426–434
5. Buttcher S., Clarke C.L.A., Soboroff I.: The TREC 2006 Terabyte Track. In *Proceedings of TREC 2006*, (2007)
6. Buttcher S., Clarke C.L.A.: Index Compression is Good, Especially for Random Access. In *Proceedings CIKM 2007*, (2007)
7. Cambazoglu B. B., Aykanat C.: Performance of query processing implementations in ranking-based text retrieval systems using inverted indices. *Inf. Process. Manage.* **42**(4) (2006) 875–898
8. Elias, P.: Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory* **21**(2) (1975) 194–203
9. Lester N., Moffat A., Webber W., Zobel J.: Space-limited ranked query evaluation using adaptive pruning. In *Proceedings of the WISE Workshop on Information Systems Engineering* (2005) 470–482
10. Moffat A., Zobel J.: Self Indexing Files for Fast Text Retrieval *ACM Transactions on Information Systems* **14**(4) (1996) 349–379
11. Ounis I., Amati G., Plachouras V., He B., Macdonald C., and Lioma C.: Terrier: A high performance and scalable information retrieval platform. In *Proceedings of SIGIR OSIR Workshop 2006* (2006)
12. Persin M., Zobel J., Sacks-Davis R.: Filtered document retrieval with frequency-sorted indexes. *J. American Society of Information Science*, **47** (1996)
13. Robertson S.E., Walker S., Jones S., Hancock-Beaulieu M.M., Gatford M.: Okapi at TREC-3 In *Proceedings of TREC 3* (1994)
14. Soffer A., Carmel D., Cohen D., Fagin R., Farchi E., Herscovici M., Maarek Y.S.: Static Index Pruning for Information Retrieval Systems. *Proceedings of ACM SIGIR 2001* (2001) 43–50
15. Strohman T., Turtle H., Croft W.B.: Optimization strategies for complex queries In *Proceedings of ACM SIGIR 2005* (2005) 219-225
16. Strohman T., Croft W.B.: Efficient document retrieval in main memory In Proceedings of ACM SIGIR 2007 (2007) 175–182
17. Turtle H., Flood J.: Query Evaluation : Strategies and Optimisations. *Information Processing & Management*: **31**(6) (1995) 831-850
18. Witten I.H., Moffat A., Bell T.C.: Managing Gigabytes: Compressing and Indexing Documents and Images Morgan Kaufmann (1999).