Efficient Dynamic Pruning with Proximity Support

Nicola Tonellotto
Information Science and Technologies Institute
National Research Council
Via G. Moruzzi 1, 56124 Pisa, Italy
nicola.tonellotto@isti.cnr.it

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

ABSTRACT

Modern retrieval approaches apply not just single-term weighting models when ranking documents - instead, proximity weighting models are in common use, which highly score the co-occurrence of pairs of query terms in close proximity to each other in documents. The adoption of these proximity weighting models can cause a computational overhead when documents are scored, negatively impacting the efficiency of the retrieval process. In this paper, we discuss the integration of proximity weighting models into efficient dynamic pruning strategies. In particular, we propose to modify document-at-a-time strategies to include proximity scoring without any modifications to pre-existing index structures. Our resulting two-stage dynamic pruning strategies only consider single query terms during first stage pruning, but can early terminate the proximity scoring of a document if it can be shown that it will never be retrieved. We empirically examine the efficiency benefits of our approach using a large Web test collection of 50 million documents and 10,000 queries from a real query log. Our results show that our proposed two-stage dynamic pruning strategies are considerably more efficient than the original strategies, particularly for queries of 3 or more terms.

Categories and Subject Descriptors: H.3.3 [Information Storage & Retrieval]: Information Search & Retrieval

General Terms: Algorithms, Performance, Experimentation

Keywords: Dynamic Pruning, Efficient Proximity

1. INTRODUCTION

In most information retrieval (IR) systems, the relevance score for a document given a query follows the general outline given by the best match strategy: a score is calculated for each query term occurring in the document. These scores are then aggregated by a summation to give the final document relevance score. However, there are many queries where the relevant documents contain the query terms in close proximity. Hence, modern retrieval systems apply not just single-term weighting models when ranking documents. Instead, proximity weighting models are commonly applied, which highly score the co-occurrence of pairs of query terms in close proximity to each other in documents [8].

LSDS-IR 2010, July 23, 2010, Geneva, Switzerland.

Dynamic pruning strategies reduce the scoring of documents, such that efficient retrieval can be obtained, without impacting on the retrieval effectiveness before rank K - such strategies are *safe-up-to-rank-K*. However, when additional proximity scores must be calculated for each document, the computational overhead impacts the efficiency of the retrieval process. While pruning techniques have been studied to efficiently score documents without considering term proximity [4, 20], there are very few proposals considering efficient top K retrieval where proximity is considered [19, 21, 22]. Moreover, these proposals require modifications of the index structure to implement efficient scoring strategies. Indeed, such modifications include sorting the posting lists by frequency or impact [2, 10], or using additional index structures containing the intersection of pairs of posting lists [19, 21, 22]. However, these can lead to negative effects on other aspects of the IR system, such as the compression of index structures or the impossibility to use other existing ranking strategies.

This work contributes a study into the behaviour of dynamic pruning strategies when combined with proximity weighting models. In particular, we analyse two existing document-at-a-time (DAAT) dynamic pruning strategies, namely MaxScore [20] and Wand [4], that can efficiently score documents without decreasing the retrieval effectiveness at rank K, nor requiring impact sorted indices. Moreover, we propose a runtime modification of these strategies to take into account proximity scores. We generate at runtime the posting lists of the term pairs, and transparently include the processing of these pair posting lists in the MaxScore and Wand strategies. Next, we propose a reorganisation of these strategies to increase their efficiency. Using thorough experiments on a 50 million document corpus and 10,000 queries from a real query log, we evaluate the proposed modification to determine their efficiency.

The remainder of this paper is structured as follows: In Section 2, we describe the state-of-the-art approaches to efficient ranking and the current existing solutions taking into account proximity scoring. In Section 3, we describe in detail the proposed framework to support proximity scores in DAAT strategies, and in Section 4, we evaluate the efficiency of the proposed modification. We provide concluding remarks in Section 5.

2. BACKGROUND

In the following, we outline the state-of-the-art strategies of dynamic pruning, followed by a discussion on proximity weighting models.

2.1 Dynamic Pruning

The algorithms to match and score documents for a query fall into two main categories [16]: in *term-at-a-time (TAAT)* scoring, the query term posting lists are processed and scored in sequence, so that documents containing query term t_i gain a partial score before scoring commences on term t_{i+1}. In contrast, in *document-at-a-time (DAAT)* scoring, the query term postings lists are processed in parallel, such that all *postings* of document d_j are considered before scoring commences on d_{j+1}. Compared to TAAT, DAAT has a smaller memory footprint than TAAT, due to the lack of maintaining intermediate scores for many documents, and is reportedly applied by large search engines [1]. An alternative strategy to DAAT and TAAT is called score-at-a-time [2], however this is suitable only for indices sorted or partially sorted by document importance, which must be calculated before the actual query processing. The algorithms from the family of threshold algorithms [10] work similarly.

Efficient dynamic pruning strategies do not rank every document in the collection for each user query; they manage to rank only the documents that will have a chance to enter in the top-K results returned to the users. These strategies are safe-up-to-rank-K [20], meaning that the ranking of documents up to rank K will have full possible effectiveness, but with increased efficiency. Dynamic pruning strategies rely on maintaining, at query scoring time, a threshold score that documents must overcome to be considered in the top-K documents. To guarantee that the dynamic pruning strategy will provide the correct top-K documents, an *upper bound* for each term on its maximal contribution to the score of any document in its posting list is used. In this paper, we focus on two state-of-the-art safe-up-to-rank-K DAAT dynamic pruning strategies, namely MaxScore and Wand.

The MaxScore strategy maintains, at query scoring time, a sorted list containing the current top-K documents scored so far. The list is sorted in decreasing order of score. The score of the last top-K document is a *threshold* score that documents must overcome to be considered in the top-K documents. A new document is given a *partial* score while the posting lists with that document are processed. A document scoring can terminate early when it is possible to guarantee that the document will never obtain a score greater than that of the current threshold. This happens when the current document score plus the upper bounds of terms yet to be scored is not greater than the threshold.

The Wand strategy maintains the same top-K documents list and the threshold score, but, for any new document, it calculates an *approximate* score, summing up some upper bounds for the terms associated with the document. If this approximate score is greater than the current threshold, then the document is fully scored. It is then inserted in the top-K candidate document set if this score is greater than the current threshold, and the current threshold is updated. If the approximate score check fails, the next document is processed. The selection of the next document to score is optimised [4] – however, for our purposes, it is of note that the set of postings lists are sorted by the document identifier (docid) they currently represent. More details on the Wand document selection strategy, which uses the skipping [16] of postings in the posting lists to reduce disk IO and increase efficiency, is presented in Appendix A.

The MaxScore and Wand dynamic pruning strategies can both enhance retrieval efficiency, whilst ensuring that the top K documents are fully scored – i.e. that the retrieval effectiveness at rank K is not at all negatively impacted. Generally, speaking, Wand is more efficient [14], due to its ability to skip postings for unimportant query terms. Note that both strategies examine at least one term from each document, and hence cannot benefit efficiency for single term queries.

2.2 Proximity

There are many queries where the relevant documents contain the query terms in close proximity. Hence, modern retrieval systems apply not just single-term weighting models when ranking documents. Instead, proximity weighting models are commonly applied, which highly score the co-occurrence of pairs of query terms in close proximity to each other in documents [8]. Hence, some scoring proximity (or term dependence) models have recently been proposed that integrate single term and proximity scores for ranking documents [5, 15, 18]. In this manner, the basic ranking model of an IR system for a query Q can be expressed as:

$$\text{score}_Q(d, Q) = \omega\, S(d) + \kappa \sum_{t \in Q} \text{score}(tf_d, *_d, t) + \phi\, \text{prox}(d, Q)$$

where $S(d)$ is the combination of some query independent features of document d (e.g. PageRank, URL length), and $\text{score}(tf_d, *_d, t)$ is the application of a weighting model to score tf_d occurrences of term t in document d. $*_d$ denotes any other document statistics required by a particular weighting model (e.g. document length). $\text{prox}(d, Q)$ represents some proximity scoring function. The influence of the various features is controlled using weights ω, κ and ϕ.

However, none of the proximity weighting models proposed have been designed for efficient document scoring. The main approaches to integrate proximity weighting models into pruning strategies require modifications to the index structure to include information on the proximity scores upper bounds. In [19, 21, 22], the authors detail several approaches to leverage early termination when proximity scores are included in the ranking model. While these strategies alter the index structure (e.g. by adding term-pair inverted indices), we aim to exploit the proximity scores without modifying the index structure (other than keeping position occurrence information in the standard inverted index posting list). In particular, we use the sequential term dependence model of Markov Random Fields (MRF) [15], which has been shown to be effective at modelling the proximity of query term occurrences in documents. In MRF, the proximity score is calculated as follows:

$$\text{prox(d,Q)} = \sum_{p=(t_i,t_{i+1}) \in Q} \Big(\text{score}\big(pf(t_i, t_{i+1}, d, k_1), l_d, p\big) + \text{score}\big(pf(t_i, t_{i+1}, d, k_2), l_d, p\big) \Big)$$

where $pf(t_i, t_{i+1}, d, k)$ represents the number of occurrences of the pair of sequential query terms (t_i, t_{i+1}) occurring in document d in windows of size k (abbreviated as pair frequency pf_d). Following [15], we set $\kappa = 1, \phi = 0.1$, and $k_1 = 2$ and $k_2 = 8$ to account for the proximity of two terms as an exact phrase, and proximity at distance 8, respectively. $\text{score}(pf_d, l_d, p)$ is implemented using Dirichlet language modelling [11], but where pair frequency takes the role of term frequency. However, for the background statistics of the language model, in contrast to term weighting, when using proximity weighting, it is common to assume a constant frequency for the pair in the collection [13][1].

[1]As implemented by the authors of MRF in the Ivory retrieval system, see `www.umiacs.umd.edu/~jimmylin/ivory`

3. FRAMEWORK

The integration of proximity weighting models within efficient dynamic pruning strategies requires the materialisation of term pair posting lists and their integration into the existing dynamic pruning decision mechanism. In the following we discuss how we proposed to address both aspects.

3.1 Term pair posting lists

Most dynamic pruning algorithms use posting list iterators – object-oriented interfaces to a posting list, allowing a posting to be read, or to be moved on to the next posting. With a standard inverted index, one posting list's iterator represents the documents in which a single query term occurs, ordered by docid.

Proximity weighting models require knowledge of the occurrence of pairs of query terms in a document. The posting list of pairs of terms can be constructed either statically (i.e., at indexing time, calculating the intersections of all pairs of term posting lists) or dynamically (i.e., at retrieval time, generating term pair postings on the fly). Previous approaches [19, 21, 22] investigated different methodologies to statically calculate these intersections. However, the static approach has two drawbacks. Firstly, storing new posting lists requires additional space on disk, and secondly, the pairs of terms whose posting lists must be intersected must be known in advance (e.g. by identifying popular phrases in the corpus [22]), to avoid generating a large number of new, potentially useless posting lists. While these drawbacks may be lightened by caching solutions to store paired posting lists [12], even in this case, there is always a relative consumption of disk or memory resources.

Instead, the pair posting lists can be built dynamically. Given two single term iterators on postings lists, there is a valid term *pair posting* each time they point to the same docid. In order to transparently include these pair postings in existing DAAT strategies, we must be sure that they are ordered by docid. A pair posting list is illustrated in Figure 1, based on the postings for terms t_1 and t_2. In our proposed approach, to avoid additional I/O operations at runtime, only the single term posting lists are responsible for reading from disk and decompressing the single postings, while the pair posting docid is updated each time a new single posting is read with the minimum of the current single term docids. The pair posting is valid only when the docids of the underlying single term posting lists are equal (i.e., in Figure 1, only two valid postings exist, namely docid 1 and docid 8.). When a term posting list ends, all the associated pair posting lists end as well. Overall, the pair posting list is docid-sorted and cannot skip over potentially useful term pair postings, however, a number of invalid pair postings will occur (e.g. (8,2) and (9,14) in Figure 1).

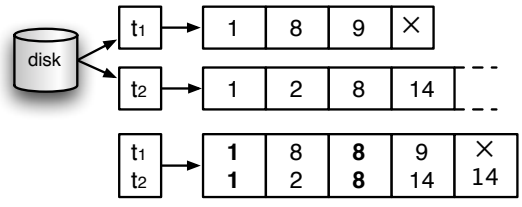

Figure 1: The dynamic creation of a pair posting list for terms t_1 and t_2. Bold entries are valid pair postings, while $\times$ indicates the end of a posting list.

3.2 Dynamic pruning with proximity

The dynamic pair posting lists can be directly put into work in existing DAAT strategies without modification. When a term pair posting is selected for scoring, it is necessary to calculate the exact value for the pair frequency at window size k, by comparing the lists of positions stored in both term postings. With dynamic pruning strategies (MaxScore and Wand), this computation can be avoided if the posting is not considered for scoring. Moreover, both the MaxScore and Wand pruning strategies require upper bounds on the score contributions of single terms. Hence, when using proximity, we need also to provide upper bounds on the score contributions of pairs as well. In [4], the authors proposed using a dynamic estimation of the inverse document frequency of pairs to determine the upper bound (the particular proximity weighting model is not defined, but assumed to be similar to [5]). In [14], we proposed a new approximation for upper bounds of the Markov Random Fields, requiring only the knowledge of the maximum term frequency of the postings in the two term posting lists.

We now describe how proximity weighting is achieved using the dynamic pruning strategies. In particular, the MaxScore strategy must always know the minimum docid in the currently processed posting lists set (which can be obtained by maintaining a heap), while the Wand strategy must have access to the posting lists sorted by docid (i.e., in the worst case, every posting in each posting list must be removed and inserted in a sorted set). However, when proximity is considered, many extra pair postings must be considered (i.e., $|Q|$ single term postings, plus an additional $2(|Q|-1)$ pair postings) – causing the efficiency of Wand to be hindered. Moreover, both strategies must make additional checks to ensure that only 'valid' pair postings are considered, which can cause a performance bottleneck.

To deal with these limitations, we propose a modification that can be applied to both MaxScore and Wand pruning strategies, whereby the processing of single terms is separated from that of term pairs during each document scoring. We refer to these *two-stage* strategies as MaxScoreP and WandP. In particular, if a pair posting is updated after each term posting update, we will generate two potentially invalid pair postings. With the proposed modification, we update the pair postings only after all single terms have been moved to their respective next posting. This implies that we can generate only one pair posting instead of two each time both of the single term posting iterators advance. Hence, MaxScoreP and WandP process the single term posting lists according to their respective algorithms, however the term pairs are subsequently processed in a second stage using early termination, according to the MaxScore strategy. The use of early termination of proximity scoring is motivated by the fact that the pair frequency of a pair posting is expensive to compute (in comparison to term frequency, which is directly recorded in the posting) – hence early termination can reduce the unnecessary pair frequency and proximity score calculations.

In summary, we propose to implement proximity scoring using only normal index structures at retrieval time, and in such a way to integrate directly with existing DAAT dynamic pruning strategies. Moreover, we propose a modification to these dynamic pruning strategies, where the single terms are processed first according to the original dynamic pruning strategy, while the terms pairs are processed according to the MaxScore strategy. This can significantly

reduce the performance impact of additional data structures required at runtime, and, as will be shown in Section 4, leads to improved retrieval efficiency. Moreover, both of the MaxScoreP and WandP modified strategies remain safe-up-to-rank-K.

4. EVALUATION

In the following experiments, we want to evaluate the benefits of the proposed modification of the dynamic pruning strategies. We are mainly interested in efficiency, because all strategies are safe-up-to-rank-K – hence have no impact on effectiveness. We tackle the following research questions:

1. How do MaxScore and Wand compare when applying the Markov Random Fields proximity weighting model?
2. Do the proposed modifications benefit the efficiency when applying proximity?
3. What impact does the length of the query have?

Experiments are performed using a 230GB 50 million English document subset of the TREC ClueWeb 09 (CW09B) corpus [6]. Documents are ranked using the Dirichlet LM weighting model [11] (with parameter setting $\mu = 4000$) and the Markov Random Fields proximity weighting (see Section 2.2). No query-independent features are used (i.e., $\omega = 0$). CW09B is indexed using the Terrier IR platform [17][2], applying Porter's English stemmer and removing standard stopwords. In the posting list, docids are encoded using Elias Gamma-encoded deltas [9] and term frequencies using Elias Unary [9]. The positions of occurrences of the term within the document are also recorded in each posting, using Elias Gamma-encoded deltas. Each posting list also includes skip points [16], one every 10,000 postings. The resulting size of the inverted index is 72GB.

For testing retrieval efficiency, we extract a stream of user queries from a real search engine log. In particular, we select the first 10,000 queries of the MSN 2006 query log [7], applying Porter's English stemmer and removing standard stopwords (empty queries are removed). The experiments measure the average query response time for each dynamic pruning strategy, broken down by the number of query terms (1, 2, 3, 4 and more than 4). The number of documents retrieved for each query is $K = 1,000$. All experiments are made using a dual quad-core Intel Xeon 2.6GHz, with 8GB RAM and a 2TB SATA2 disk containing the index.

In the following, we compare five strategies, namely: an exhaustive "Full" DAAT strategy, which fully scores every posting for all query terms and pairs; the original MaxScore and Wand dynamic pruning strategies without any modification; and the proposed two-stage MaxScoreP and WandP dynamic pruning strategies which integrate the modification proposed in Section 3.2. Every strategy uses dynamic pair posting lists, as discussed in Section 3.1.

Table 1 details the average query response time for both the original and two-stage strategies per number of query terms. From this table, we note that the average response time are reduced by approximately 22%-30% by applying the original MaxScore, but for the original Wand the average response times are worse than for Full DAAT scoring. This counters the normal efficiency of Wand, and supports

[2]http://terrier.org

	# of query terms				
	1	2	3	4	> 4
# queries	3456	3149	1754	853	550
Full	**0.53**	2.76	5.57	9.95	16.42
Original strategies					
MaxScore	**0.53**	2.07	3.92	6.45	12.68
Wand	**0.53**	3.01	5.78	10.67	18.42
Two-stage strategies					
MaxScoreP	**0.53**	1.90	3.32	5.27	8.51
WandP	**0.53**	**1.67**	**2.73**	**4.46**	**7.77**

Table 1: Average response times (in seconds), original strategies and two-stage strategies.

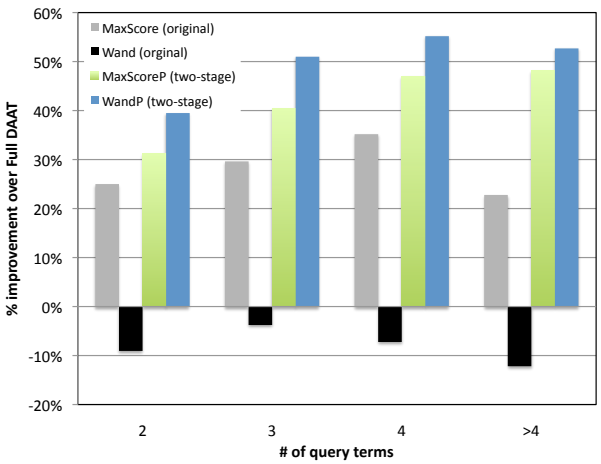

Figure 2: The relative impact on the average response time of the proposed strategies w.r.t. the Full DAAT strategy.

our assertion that Wand is not suitable for proximity scoring in its normal form. Indeed, when using the pair posting lists, there is a higher number of posting lists to maintain in the docid sorted set of posting lists used by Wand, in addition to the check that each pair posting is valid.

In contrast, the two-stage pruning strategies perform better in comparison to their original versions. MaxScoreP exhibits improvements of average response times varying from 31% for two terms, up to 48% for more than four terms, while WandP benefits vary from 39% to 58%. Moreover, we note that WandP exhibits better efficiency than MaxScoreP, in common with MaxScore versus Wand for non-proximity queries [14]. For single term queries, no proximity is applied, and, as expected, all dynamic pruning strategies are equally efficient to Full DAAT scoring.

Figure 2 summarises the percentage differences of the dynamic pruning strategies with respect to the Full DAAT scoring, for varying lengths of query. As already reported, the two-stage MaxScoreP and WandP strategies outperform their original equivalents. Moreover, their benefits increase as the length of the queries (and hence pairs of terms) increase, up to 40-55% improvements for queries with 3 or more terms.

5. CONCLUSIONS

In this work, we examined how to efficiently score documents using dynamic pruning strategies when using the Markov Random Field proximity weighting model [15]. In particular, we discussed how pair posting lists could be used to allow proximity scoring without changes to the

underlying index. Moreover, the most efficient way to score documents using DAAT dynamic pruning strategies was discussed. We proposed that dynamic pruning should be performed in a two-stage process (as exemplified by the MaxScoreP and WandP strategies), whereby only single query terms are processed and pruned in the first stage. The pair posting lists are only considered during a second stage, since they are omitted from consideration in the first stage.

We performed large-scale experiments comparing the orginal and proposed two-stage dynamic pruning strategies, using a corpus of 50 million documents, and 10,000 user queries from a real query log. Our results demonstrated the benefit of the two-stage versions of the dynamic pruning strategies, particularly for queries of 3 or more terms, and are a promising start for the future efficient examination of other proximity weighting models.

Dynamic pruning techniques have previously been shown to be readily applicable in distributed retrieval settings [3]. Similarly, we infer that our strategies can also be applied in distributed retrieval without requiring adaptation.

6. REFERENCES

[1] A. Anagnostopoulos, A. Z. Broder and D. Carmel. Sampling search-engine results. In *Proc. of WWW 2005*, 245–256.

[2] V. N. Anh and A. Moffat. Pruned query evaluation using pre-computed impact scores. In *Proc. of SIGIR 2006*, 372–379.

[3] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Plachouras and L. Telloli. On the feasibility of multi-site web search engines. In *Proc. of CIKM 2009*, 425–434.

[4] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer and J. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc. of CIKM 2006*, 426–434.

[5] S. Büttcher, C. L. A. Clarke and B. Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proc. of SIGIR 2006*, 621–622.

[6] C. L. A. Clarke, N. Craswell and I. Soboroff. Overview of the TREC 2009 Web track. In *Proc. of TREC 2009*.

[7] N. Craswell, R. Jones, G. Dupret and E. Viegas. *Proc. of the Web Search Click Data Workshop at WSDM 2009*.

[8] W. B. Croft, D. Metzler and T. Strohman. Search Engines: Information Retrieval in Practice Addison Wesley, 2009.

[9] P. Elias. Universal codeword sets and representations of the integers. *Information Theory, IEEE Transactions on*, 21(2):194–203, 1975.

[10] R. Fagin, A. Lotem and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.* 66(4):614–656, 2003.

[11] J. Lafferty and C. Zhai. A study of smoothing methods for language models applied to information retrieval. In *Proc. of SIGIR 2001*, 334–342.

[12] X. Long and T. Suel. Three-level caching for efficient query processing in large web search engines. In *Proc. of WWW 2005*, 257–266.

[13] C. Macdonald and I. Ounis. Global Statistics in Proximity Weighting Models. In *Proc. of Web N-gram Workshop at SIGIR 2010*.

[14] C. Macdonald, N. Tonellotto and I. Ounis. Upper Bound Approximations for Dynamic Pruning. *Manuscript submitted for publication*, 2010.

[15] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. In *Proc. of SIGIR 2005*, 472–479.

[16] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *Transactions on Information Systems*, 14(4):349–379, 1996.

[17] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald and C. Lioma. Terrier: a high performance and scalable information retrieval platform. In *Proc. of OSIR Workshop at SIGIR 2006*.

[18] J. Peng, C. Macdonald, B. He, V. Plachouras and I. Ounis. Incorporating term dependency in the DFR framework. In *Proc. of SIGIR 2007*, 843-844.

[19] R. Schenkel, A. Broschart, S. Hwang, M. Theobald and M. Gatford. Efficient text proximity search. In *Proc. of SPIRE 2007*, 287–299.

[20] H. Turtle and J. Flood. Query evaluation: strategies and optimizations. *Information Processing and Management*, 31(6):831–850, 1995.

[21] M. Zhu, S. Shi, M. Li and J.-R. Wen. Effective top-k computation in retrieving structured documents with term-proximity support. In *Proc. of CIKM 2007*, 771–780.

[22] M. Zhu, S. Shi, N. Yu and J.-R. Wen. Can phrase indexing help to process non-phrase queries? In *Proc. of CIKM 2008*, 679–688.

APPENDIX

A. WAND DOCUMENT SELECTION

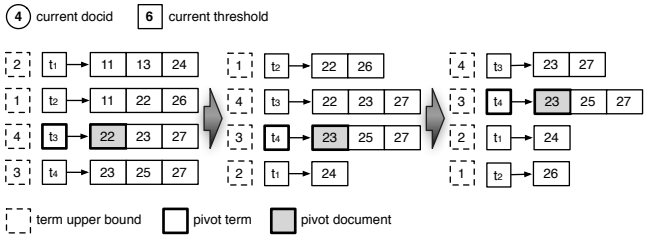

Figure 3: How the WAND strategy selects the next document to score

The selection of the next document performed in the Wand strategy is explained with the help of Figure 3. The posting lists are maintained in increasing order of docid. Then a *pivot term* is computed, i.e. the first term for which the accumulated sum of upper bounds of preceding terms and itself exceeds the current threshold (e.g., term t_3 with accumulated score of 7). The corresponding docid identifies the *pivot document*, i.e. the smallest docid having a chance to overcome the current threshold. If the current docids of the previous terms are equal to the pivot document docid, the document is fully scored. Otherwise, one of the preceding terms posting list is moved to the pivot document docid, and the procedure is repeated. In the example, at the third step a good candidate document is found (23) and it is fully processed. Wand can benefit from skipping every posting list to a particular document, however the selection of the right document requires some additional CPU time.