On the Cost of Negation for Dynamic Pruning

Joel Mackenzie[1], Craig Macdonald[2], Falk Scholer[1], and J. Shane Culpepper[1]

[1] RMIT University, Melbourne, Australia
[2] University of Glasgow, Glasgow, Scotland, UK

Abstract. Negated query terms allow documents containing such terms to be filtered out of a search results list, supporting disambiguation. In this work, the effect of negation on the efficiency of disjunctive, top-k retrieval is examined. First, we show how negation can be integrated efficiently into two popular dynamic pruning algorithms. Then, we explore the efficiency of our approach, and show that while often efficient, negation can negatively impact the dynamic pruning effectiveness for certain queries.

Keywords: Dynamic Pruning · Query Semantics · Negation · Efficiency

1 Introduction

Modern Information Retrieval (IR) systems are extremely complex, often using a multi-stage approach to support both efficient and effective retrieval. In order to improve effectiveness, user queries can be rewritten into improved representations through query expansion, query reduction, named entity recognition, and other advanced rewriting strategies [1, 2]. These strategies can make use of the *term negation* operator [3], which allows the results to be filtered so that documents containing negated terms are not returned by the IR system. In particular, this operator is useful for disambiguating query terms, allowing irrelevant results to be filtered from the result list. Furthermore, while negation is not commonly used directly by users of web search systems [4], many large scale search systems still explicitly support the negation operator as an advanced search feature, including Google,[3] Bing,[4] and Twitter [5], where users can manually negate query terms if desired.

2 Dynamic Pruning Strategies

Dynamic pruning strategies are often used in large scale search systems to allow efficient *candidate generation*, where typically hundreds or thousands of potentially relevant documents are quickly identified for further consideration [6]. In this work, we focus on two popular *Document-at-a-Time* (DAAT) dynamic pruning strategies, namely WAND [7] and BMW [8].

[3] `https://support.google.com/websearch?p=adv_operators` (Accessed Oct. 2017)
[4] `https://msdn.microsoft.com/library/ff795633.aspx` (Accessed Oct. 2017)

The WAND algorithm provides efficient traversal of document-ordered postings lists by storing the *upper-bound* score that the ranking function can contribute for each given term (U_t). WAND uses the score of the lowest scoring document in the current result set as a *threshold* (θ). Then, the values of U_t are used to estimate an upper-bound score that a document may achieve, and only documents with an upper-bound score greater than θ are evaluated, allowing redundant documents to be skipped.

Improving the WAND algorithm, Ding and Suel [8] proposed the *Block-Max* WAND (BMW) algorithm. Instead of just storing the U_t score, BMW also stores the maximum score for each *block* in each postings list ($U_{b,t}$). Query processing is the same as WAND, but after a potential (*pivot*) document has been found, the block scores are used to refine the upper-bound score, to ensure the potential score of the pivot is still greater than θ. If not, the block max scores can be used to induce additional skipping, resulting in faster retrieval. We refer the reader to the work of Broder et al. [7], Ding and Suel [8], and Petri et al. [9] for more information on the workings of these algorithms.

Integrating Negation. We now explain how negation can be efficiently supported in modern dynamic pruning algorithms. Firstly, the query parser is modified to recognise the term negation operator (-). Next, the query processing framework is modified to maintain two sets of postings lists: those to be scored, and those which are negated. Next, an efficient function called `isNegated` is implemented, which, when given the set of negated postings and a document identifier (DocID), returns `true` if the document contains a negated term, and `false` otherwise. This function efficiently skips to a compressed block that may contain the given document identifier, decompresses the block, and probes for the candidate DocID. If found, the function returns `true`. Otherwise, the next negated list is considered. This process is repeated for all negated lists which have cursors before the pivot document. Note that, similar to WAND, postings are sorted from current smallest to largest document identifier, as this allows early exiting from the `isNegated` function.

For the WAND algorithm, processing proceeds as usual, except that once a pivot document has been found, the pivot is checked for negated terms using the `isNegated` function. If it does contain negated terms, we select a new pivot. Otherwise, we proceed to score the document. We denote this algorithm as N-WAND in our experiments and discussion. Figure 1 shows a pictorial example of N-WAND pivoting and the `isNegated` function.

For the BMW algorithm, we propose two versions to address negated query terms. The first version will select a pivot document, and perform the refined block-max check. If the block-max check passes, the document is then examined for negated terms. We denote this version N-BMW$_{v1}$. In the second version, we switch the order of the negation test and the block-max check. That is, we select a pivot, and then ensure that it does not contain any negated terms. If the pivot contains no negated terms, we then continue to the block-max check. Otherwise, we select a new pivot. This algorithm is denoted as N-BMW$_{v2}$.

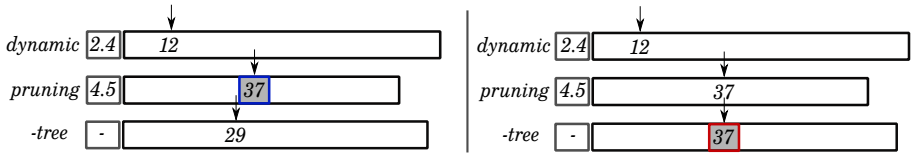

Fig. 1: An example of N-WAND processing for the query "*-tree dynamic pruning*" with a heap threshold of $\theta = 5.1$. Left shows the pivot selection: document 37 is the next document that can potentially make the heap since $2.4 + 4.5 > \theta$. Right shows the check negation function: the negated list is searched for the next DocID ≥ 37. Since 37 was found in the negated list, a new pivot will be selected.

3 Experiments

Experimental Setup. Experiments are conducted on the standard TREC ClueWeb09B collection, which contains 50 million web documents. We use a custom implementation of the WAND and BMW algorithms [10], modified as discussed above, and the code is made available for reproducibility.[5] All timings are performed in-memory on an otherwise idle Red Hat Enterprise Linux Server (v7.2) with two Intel Xeon E5-2690 v3 CPUs, and 256GB of RAM. All reported timings use a single core only, and are the average of 3 runs. The query log consists of 317 queries extracted from the Excite query logs from 1997, 1999 and 2001 [4]. We extracted all queries containing negated terms, removed illegal characters, and then s-stemmed and stopped the queries. We also created a copy of this query set that has the negated terms removed, which allows us to compare the negated algorithms with the plain WAND and BMW algorithms. For example, consider the query "*fish net -stocking*"; the corresponding plain query would be "*fish net*". For each query, the top-k documents are ranked and retrieved using a BM25 ranking model.

Comparing Negation and Plain Disjunction. For each algorithm, we retrieve the top $k = \{10, 100, 1{,}000, 10{,}000\}$ documents. The resulting response times are shown in Fig. 2. Firstly we note that, similar to the disjunctive algorithms, the efficiency of processing queries with negated terms decreases as the value of k increases, and the N-BMW variants outperform N-WAND for all values of k. In addition, adding negated terms makes processing slower than plain disjunctive processing, although the gap is quite small. This is not surprising, as negation involves an additional check for every pivot document that is considered by WAND or BMW. To further explore the overhead caused by negated terms, we profile each of the algorithms (Table 1). Interestingly, we find that on average, the negated algorithms score fewer postings than when processing the corresponding plain query. Therefore, the cost of negation is not in the scoring of postings, but rather, in the negation check and pivot selection aspects of the algorithms.

[5] `http://github.com/JMMackenzie/DaaT-Negation`

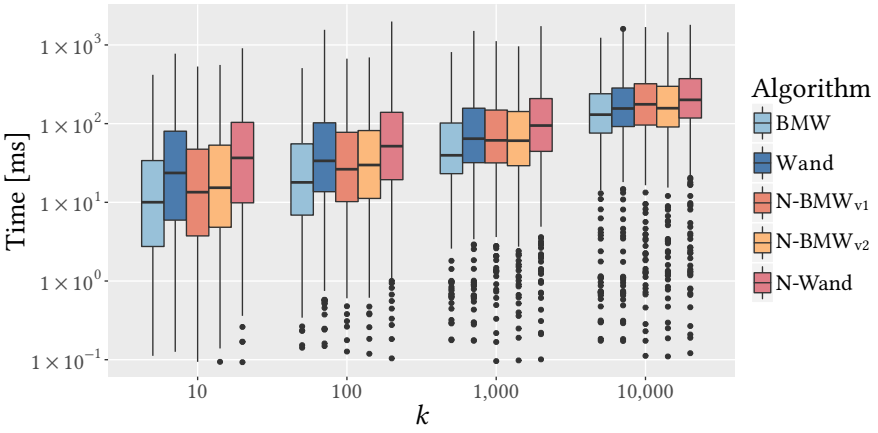

Fig. 2: Comparing all Bmw and Wand algorithms across all queries for varying values of k. The plain Bmw and Wand algorithms run each query by simply ignoring the negated terms.

Further Exploration of Dynamic Pruning Power. Examining Table 1 further, it is clear that the algorithms that process negated terms have *lower* thresholds than the corresponding plain algorithms. Since some documents will not make the heap due to containing negated terms, the heap threshold does not rise as quickly or as highly it would in a plain disjunctive setting. A lower threshold means less pruning power, which results in more pivot documents being considered, more negation checks, and less skipping across the DocID space. As an example, we plot the value of the threshold, θ, each time a document is scored for a single query, "*silver city -new mexico*", and for the corresponding plain disjunction, "*silver city mexico*" (Fig. 3). Clearly, the negation results in a lower threshold, as documents that would usually make the top-k in the disjunctive processing may contain negated terms. This explains why N-Bmw$_{v2}$ outperforms N-Bmw$_{v1}$ for larger values of k; N-Bmw$_{v2}$ performs less redundant block-max checks than N-Bmw$_{v1}$, especially as the density of negated terms in high scoring documents increases. Similar observations were made by Petri et al. [9], where the authors explored the impact of various ranking functions on the dynamic pruning power of both Wand and Bmw.

4 Conclusion and Future Work

In this work, we presented an initial investigation on the impact that negation has on modern dynamic pruning algorithms. We first show how to integrate efficient negation into modern dynamic pruning strategies, namely Wand [7] and Bmw [8], and then compare and contrast the efficiency of such extensions across a log of around 300 real queries. Our results demonstrate that although negation

Algorithm	Postings Processed	Unique Pivots	Final θ	Mean Time	Median Time
		$k = 10$			
WAND	271,533	611,013	18.32	63.63	23.62
N-WAND	226,605	628,349	17.74	82.91	36.68
BMW	18,958	222,032	18.32	28.73	10.03
N-BMW$_{v1}$	16,872	250,090	17.74	37.98	13.46
N-BMW$_{v2}$	16,873	254,634	17.74	40.03	15.33
		$k = 100$			
WAND	412,657	795,176	16.25	86.50	33.65
N-WAND	357,645	836,650	15.64	112.93	51.72
BMW	51,279	351,743	16.25	42.59	17.93
N-BMW$_{v1}$	46,046	389,226	15.64	60.65	26.34
N-BMW$_{v2}$	46,050	396,606	15.64	62.91	29.80
		$k = 1{,}000$			
WAND	607,186	1,051,990	13.02	124.24	64.39
N-WAND	535,137	1,125,580	12.27	167.54	94.58
BMW	159,800	571,003	13.02	77.51	39.60
N-BMW$_{v1}$	147,730	642,714	12.27	110.29	61.31
N-BMW$_{v2}$	147,742	650,942	12.27	106.74	60.70
		$k = 10{,}000$			
WAND	1,085,950	1,574,320	9.47	222.31	155.72
N-WAND	985,028	1,696,080	8.99	281.17	200.51
BMW	577,460	1,073,130	9.47	172.92	130.10
N-BMW$_{v1}$	535,980	1,201,890	8.99	237.29	176.04
N-BMW$_{v2}$	535,995	1,209,400	8.99	220.96	157.04

Table 1: Average statistics for all algorithms across the 317 queries. Although the negation algorithms process fewer postings on average (compared with their respective plain algorithms), they select more unique pivot documents due to reduced heap thresholds.

can be supported efficiently most of the time, occasionally queries can negatively impact the effectiveness of dynamic pruning. Future work will compare the N-WAND and N-BMW approaches with similar extensions to the MAXSCORE [11, 12] algorithm and the recently proposed Variable-BMW algorithms [13]. A more comprehensive exploration of the role of negation in query substitutions [1] for large scale search is a captivating problem that we leave for future work.

Acknowledgements

This work was supported by the Australian Research Council's *Discovery Projects* Scheme (DP170102231), an Australian Government Research Training Program Scholarship, and a grant from the Mozilla Foundation.

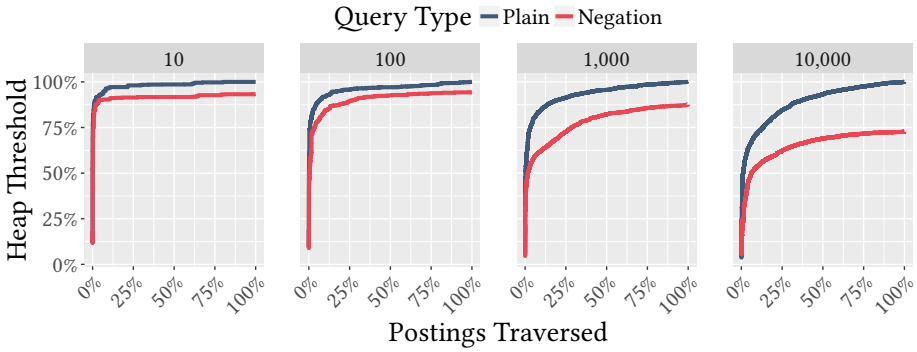

Fig. 3: Comparing the heap threshold of the negated query to the heap threshold of the plain query for the query pair "*silver city -new mexico*" and "*silver city mexico*". Clearly, by negating documents containing the term *new*, fewer high scoring documents make it into the heap, leading to a reduced threshold, and less ability for dynamic pruning to occur. As k increases, so too does the gap between the plain and negated threshold.

References

1. R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proc. WWW*, pages 387–396, 2006.
2. C. Macdonald, N. Tonellotto, and I. Ounis. Efficient & effective selective query rewriting with efficiency predictions. In *Proc. SIGIR*, pages 495–504, 2017.
3. Y. Kim, J. Seo, and W. B. Croft. Automatic Boolean query suggestion for professional search. In *Proc. SIGIR*, pages 825–834, 2011.
4. Amanda Spink, Dietmar Wolfram, Major B. J. Jansen, and Tefko Saracevic. Searching the web: The public and their queries. *JASIST*, 52(3):226–234, 2001.
5. M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin. Earlybird: Real-time search at twitter. In *Proc. ICDE*, pages 1360–1369, 2012.
6. C. Macdonald, R. L. T. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Information Retrieval*, 16(5):584–628, 2013.
7. A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Y. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc. CIKM*, pages 426–434, 2003.
8. S. Ding and T. Suel. Faster top-k document retrieval using block-max indexes. In *Proc. SIGIR*, pages 993–1002, 2016.
9. M. Petri, J. S. Culpepper, and A. Moffat. Exploring the magic of WAND. In *Proc. ADCS*, pages 58–65, 2013.
10. M. Crane, J. S. Culpepper, J. Lin, J. Mackenzie, and A. Trotman. A comparison of Document-at-a-Time and Score-at-a-Time query evaluation. In *Proc. WSDM*, pages 201–210, 2017.
11. H. Turtle and J. Flood. Query evaluation: strategies and optimizations. *Inf. Proc. & Man.*, 31(6):831–850, 1995.
12. T. Strohman, H. Turtle, and W. B. Croft. Optimization strategies for complex queries. In *Proc. SIGIR*, pages 219–225, 2005.
13. A. Mallia, G. Ottaviano, E. Porciani, N. Tonellotto, and R. Venturini. Faster blockmax wand with variable-sized blocks. In *Proc. SIGIR*, pages 625–634, 2017.