Author/s:
Clarke, CLA;Culpepper, JS;Moffat, A

Title:
Assessing efficiency–effectiveness tradeoffs in multi-stage retrieval systems without using relevance judgments

Date:
2016-08-01

Citation:
Clarke, C. L. A., Culpepper, J. S. & Moffat, A. (2016). Assessing efficiency–effectiveness tradeoffs in multi-stage retrieval systems without using relevance judgments. Information Retrieval Journal, 19 (4), pp.351-377. https://doi.org/10.1007/s10791-016-9279-1.

Persistent Link:
https://hdl.handle.net/11343/283250

Noname manuscript No.
(will be inserted by the editor)

Assessing Efficiency-Effectiveness Tradeoffs in Multi-Stage Retrieval Systems Without Using Relevance Judgments

Charles L. A. Clarke · J. Shane Culpepper · Alistair Moffat

Received: date / Accepted: date

Abstract Large-scale retrieval systems are often implemented as a cascading sequence of phases – a first filtering step, in which a large set of candidate documents are extracted using a simple technique such as Boolean matching and/or static document scores; and then one or more ranking steps, in which the pool of documents retrieved by the filter is scored more precisely using dozens or perhaps hundreds of different features. The documents returned to the user are then taken from the head of the final ranked list. Here we examine methods for measuring the quality of filtering and preliminary ranking stages, and show how to use these measurements to tune the overall performance of the system. Standard top-weighted metrics used for overall system evaluation are not appropriate for assessing filtering stages, since the output is a set of documents, rather than an ordered sequence of documents. Instead, we use an approach in which a quality score is computed based on the discrepancy between filtered and full evaluation. Unlike previous approaches, our methods do not require relevance judgments, and thus can be used with virtually any query set. We show that this quality score directly correlates with actual differences in measured effectiveness when relevance judgments are available. Since the quality score does not require relevance judgments, it can be used to identify queries that perform particularly poorly for a given filter. Using these methods, we explore a wide range of filtering options using thousands of queries, categorize the relative merits of the different approaches, and identify useful parameter combinations.

Charles L. A. Clarke
University of Waterloo, Canada
E-mail: claclarke@cs.uwaterloo.ca

J. Shane Culpepper
RMIT University, Australia
E-mail: shane.culpepper@rmit.edu.au

Alistair Moffat
The University of Melbourne, Australia
E-mail: ammoffat@unimelb.edu.au

1 Introduction

The purpose of an information retrieval system is well-defined: given a query q, and a large collection $\mathcal{D}$ of documents, identify and present a small subset of the collection by identifying documents that are deemed responsive to q. Typical collections contain billions of documents and occupy terabytes of storage. Queries are a often a few words long, and answer lists contain between ten and a few hundred items. Zobel and Moffat [32] and Büttcher et al. [6] provide surveys of these processes.

The systems that have been developed in response to these goals are assembled from a suite of possible components, with many different ways of generating query-document similarity scores so that documents can be ranked. But in broad terms, most retrieval systems can be thought of as being composed from a small number of underlying building blocks that can be categorized into three groups: *pre-ordering stages*, that make use of static index-time data such as page-rank or document quality scores (including spam scores); *filtering stages*, in which a subset of the collection is extracted; and then one or more *detailed ranking phases*, in which a comprehensive evaluation of those documents is undertaken, so that the final top-k ranked list can be generated. Each of these stages can in turn be composed of one or more steps. In this work we differentiate only between the filtering stage that identifies potential candidate documents, and the final ranking stage that works with a smaller subset of documents drawn from the entire collection.

The end-to-end effectiveness of an IR system is measured using any one of a large range of metrics such as precision (P), average precision (AP), discounted cumulative gain (DCG), normalized discounted cumulative gain (NDCG), rank-biased precision (RBP), the Q-Measure (QM), reciprocal rank (RR), expected reciprocal rank (ERR), time-biased gain (TBG), and so on [7, 13, 18, 22, 23]. Two broad families can be identified – metrics that are *recall dependent*, for which the calculated score is relative to the best that could be attained by any ranking, and hence can be regarded as an absolute assessment (AP, NDCG, QM); and metrics that are *recall independent*, for which the calculated score reflects the user's experience of the supplied ranking, and is not affected by the density or otherwise of relevant documents in the unseen part of the collection (P, DCG, RBP, RR, ERR, TBG).

Conventional metrics are not directly applicable to the task of measuring the quality (that is, contribution to overall search performance) of the pre-ordering stage, or of the filtering stage. For example, Boolean conjunction has been suggested as a useful filtering stage, so that every presented answer document has every query term in it somewhere [14]. But applying AP or NDCG to the output of a Boolean conjunction is unhelpful, since before it is seen by the user, that set of documents might be (and quite probably will be) permuted in to an entirely different ordering by the ranking stages.

Contributions

We make two key contributions. First, we demonstrate the applicability of a filter-stage evaluation approach that is based on bounding the loss of end-to-end effectiveness that can occur as a result of the filtering process. This quantification of the possible degradation is straightforward to compute for all recall-independent metrics;

and, in the experiments described below, is validated using data and topics for which relevance judgments are available. Second, we then use that approach to examine thousands of queries for which relevance judgments are not available, and compare a variety of possible early-stage filters.

Those results then provide the basis for estimating the combined multi-stage retrieval time versus effectiveness trade-off options for large-scale retrieval. Extending prior results [2], our findings show that conjunctive Boolean mechanisms are unlikely to provide useful trade-offs, despite their high retrieval speed. On the other hand, aggressive WAND evaluation strategies [5], in which answer list "safeness" is sacrificed for evaluation speed, do offer appealing options.

2 Background

A retrieval system can be regarded as being a composition of multiple phases, with any of the phases potentially being omitted, or broken in to sub-phases. We now group possible retrieval tasks into the three general phases; then describe each phase and the systems that result when they are composed in various ways; and finally in this section discuss ways in which system effectiveness can be evaluated.

Static Ordering

The first phase is a method for *static ranking*, or *pre-ordering* the documents in the collection. A function $S(\mathcal{D}, k)$ is provided that processes $\mathcal{D}$ according to some query-independent scoring regime, and then returns as an ordered set the k documents with the highest static scores. The abbreviated notation $S(\mathcal{D})$ indicates $S(\mathcal{D}, |\mathcal{D}|)$, a complete ordering of $\mathcal{D}$ according to static score, rather than a top-k truncated ranking. Examples of this first phase include orderings based on page rank, spam score, document length, perceived document quality, plus combinations of these and other query-agnostic features.

Filtering

The second phase of a retrieval system is a *filtering* stage $F(\mathcal{D}, q, k)$ that extracts a k-subset of the collection $\mathcal{D}$ in response to a query q, usually (but not always) selecting those that match q in some manner. At most k documents are selected by the filter, which retains the same ordering of documents in the output as was supplied in the input. Again, as a notational convenience, we use $F(\mathcal{D}, q)$ for $F(\mathcal{D}, q, |\mathcal{D}|)$ to cover situations in which all of the documents in $\mathcal{D}$ might potentially be allowed through by the filter. A trivial filter could simply transmit the first k documents that are presented, regardless of their merits, and withhold the remainder of the collection. A more complex filter might implement a conjunctive `AND` operation that selects the set of documents that contain all of the query terms. Or third, a simple-but-fast ranking system might be employed as a filter, passing through the top (say) $k = 10^3$ documents according to a scoring mechanism such as term overlap count.

Ranking
The third phase of a retrieval system is a *ranking* or *reordering* stage $R(\mathcal{D}, q, k)$ that selects at most k documents from $\mathcal{D}$, according to their perceived relevance to query q, and returns as an ordered sequence the top k documents in decreasing score order. For example, a BM25 or Language Model similarity computation based on collection, term, and document statistics might be used to score each document in the collection (see Zobel and Moffat [32] or Büttcher et al. [6, Chapters 8 and 9]), and then the documents with the top k scores extracted and returned. As before, the shorthand $R(\mathcal{D}, q)$ is used to represent $R(\mathcal{D}, q, |\mathcal{D}|)$, with (potentially) every document in the collection assigned a score and then returned by the ranker.

System Options
These different processes might be implemented using different underlying structures and mechanisms. For example, the pre-scoring process might happen at index construction time, and involve explicit reordering of the documents in the collection so that high-quality documents have low document numbers. The filtering stage might then be supported by a document-level (non-positional) inverted index, so that documents matching the filter's specification can be quickly identified. The ranking stage might be supported by the same index augmented by term positions; or, especially if complex features based on phrases or term proximities are being employed, might be based on document surrogates computed at the time the collection was indexed (a "direct" file).

Combining the various retrieval options in different ways gives a range of possible retrieval systems.

— *Constant*: A simple ordered list, such as a newspaper home page. In this configuration, $\mathcal{D}$ is the set of newspaper stories ordered temporally; $S()$ is a query-free scoring regime based on story recency and story importance as determined by non-query features, such as user profile or past reading behavior; and the retrieval system is of the form $S(\mathcal{D}, k)$, where $k \approx 10$, for example, is the number of stories to be presented.

— *Boolean*: General Boolean querying, such as 1980s-style abstract search systems. In this configuration, $\mathcal{D}$ is the set of abstracts; q is expressed as a conjunction of concepts, each described as a disjunction of terms or negated terms; $F()$ is a Boolean matching process; and the retrieval system is of the form $F(\mathcal{D}, q)$; in this configuration there is no limit on the number of answers provided, and no particular ordering within the documents that match the query.

— *Conjunctive-Ranked*: Ordered conjunctive querying, of a style used by some early search systems. In this configuration, $\mathcal{D}$ is the set of web pages; q is a list of query terms; $F()$ is a Boolean conjunction over those terms; and the retrieval system is of the form $F(S(\mathcal{D}), q, k)$, where $k \approx 10$ and $S()$ provides a static collection-wide ordering of some sort.

— *Normal-Ranked*: Standard document ranking, such as the systems developed by academic groups participating in TREC Ad-Hoc and Web Tracks. In this configuration, $\mathcal{D}$ is the set of web pages; q is a list of query terms; $R()$ is a ranking computation; and the retrieval system has the form $R(\mathcal{D}, q, k)$, where k is of the order of $1{,}000$.

— *Multi-Level-Ranked*: Multi-phase systems, including the ones presumed to be deployed by commercial web search providers. The retrieval system has the form $R(F(\mathcal{D}, q, k_1), q, k_2)$, where $k_1 \approx 10^3$, and $k_1 \gg k_2 \approx 10$ or 20. The filter $F()$ might be a Boolean conjunction [14], or might be a simple-but-fast ranker [30] that returns not more than k_1 (and at least k_2 documents) from the collection. The ranker $R()$ then selects and returns the top k_2 of those k_1 documents, employing a large set of features, possibly including term proximities and adjacencies, and possibly including the application of learning-to-rank techniques. There is no requirement that the top k_2 documents are a proper subset of the k_1 documents returned from any single filter. It is entirely possible for the candidate document set k_1 to be a composition of documents drawn from several different filtering approaches.

Note that all of these descriptions are declarative rather than procedural, and define the result of the computation, not a literal prescription as to how the computation is to be implemented. In an actual implementation the phases' computations might be tightly integrated into one process; or they might be separated. What is of interest is the logical structure of the computation, not the implementation details.

Effectiveness Evaluation

An *effectiveness metric* is a function $\mathsf{M}(\mathcal{D}_q, \mathcal{J}_q) \rightarrow v$, where $\mathcal{D}_q$ is an ordered or unordered sub-collection generated by some ranking system for query q; $\mathcal{J}_q \subseteq \mathcal{D}$ is a set of corresponding positive judgments for query q; and $v \in \mathcal{R}$ is a real-valued number, usually in the range zero to one. To compute a numeric effectiveness score, each element in $\mathcal{D}_q$ is checked against $\mathcal{J}_q$. Suppose that d_i is the i th element of $\mathcal{D}_q$, whether $\mathcal{D}_q$ is ordered or unordered. Then the (binary) relevance at the i th position of query q's ranking is given by

$$r_i = \begin{cases} 1 & \text{if } d_i \in \mathcal{J}_q \\ 0 & \text{otherwise}\,. \end{cases}$$

Different metrics can be defined in terms of which r_i values are considered, and how they are combined. Relevance score r_i can also be regarded as being fractional rather than integral, with $0 \le r_i \le 1$ indicating the strength of relevance of d_i, a situation known as *graded relevance*. There are also evaluation situations in which r_i is dependent not just on d_i, but on d_1 through to d_{i-1} as well, considered as a set. For example, the first time a particular document is encountered in a ranking, or a particular intent interpretation of a query is encountered, the relevance score r_i might be 1. But a subsequent listing of a duplicate document, or of a document that addresses the same intent, might be considered as corresponding to $r_i = 0$. One way of interpreting the value r_i is that it is the *utility*, or *benefit* that the user accrues by encountering that document in the ranking. In this work we assume that the r_i values can be computed in some appropriate manner, taking into account the supplied ranking, and that what is required is that the set of r_i values be converted to a numeric effectiveness score.

Metrics

A large number of effectiveness metrics have been proposed. Precision is a standard

metric applied to *Boolean* querying (which has as its output a set rather than a sequence), defined as the fraction of $\mathcal{D}_q$ that appears in $\mathcal{J}_q$. Similarly, Recall is the fraction of $\mathcal{J}_q$ that appears in $\mathcal{D}_q$; and from these two, F_1 is calculated as the harmonic mean of Precision and Recall. Variants of precision and recall can also be used to provide effectiveness scores for the ranked sequences generated by the *Constant*, *Conjunctive-Ranked* and *Normal-Ranked* retrieval systems defined earlier in this section, with scoring typically computed "to depth k", for some chosen value of k. Evaluating precision in this way results in a *top-weighted* effectiveness metric, since it is biased in favor of the top-k documents. Within the k documents, however, all are treated equally.

A range of other top-weighted metrics have been defined, with behaviors that provide a smoother transition from top to bottom of the ranking than does Precision@k. These include average precision, AP; normalized discounted cumulative gain, NDCG [13]; rank-biased precision, RBP [18]; the Q-measure, QM [22]; and time-biased gain, TBG [23]. Each of these top-weighted metrics can be applied to whole rankings, or, with varying degrees of imprecision (and varying degrees of knowledge about the magnitude of that imprecision), to fixed-length depth-k prefixes.

Provided that the required relevance judgments $\mathcal{J}_q$ are available to depth k for each member q of a set of test topics $\mathcal{Q}$, two rankers $R_a()$ and $R_b()$ can be compared by measuring the relative performance, and calculating a paired significance test over the set of corresponding values of an effectiveness metric M:

$$\langle \mathsf{M}(R_a(\mathcal{D}, q, k), \mathcal{J}_q), \mathsf{M}(R_b(\mathcal{D}, q, k), \mathcal{J}_q) \mid q \in \mathcal{Q} \rangle \, .$$

Judgments

A critical question that arises is how the judgments $\mathcal{J}_q$ are formed. In most evaluations human judges examine documents, deciding for each whether or not it is relevant or irrelevant. Construction of exhaustive judgments is impossibly expensive, and so only subsets of the collection are normally judged, with *pooling to depth* ℓ one strategy that can be used to control the cost of forming judgments so that a set of systems can be evaluated. In this approach, a document is judged for a query if and only if its minimum rank in any of the system runs being pooled is ℓ or less. Hence, the set $\mathcal{J}_q$ of judged relevant documents for a query q is likely to be a subset of the true set of relevant documents [31]. As a result, the documents in $\mathcal{D}_q$ will fall in to one of three categories: those judged relevant; those judged non-relevant; and those that are unjudged. One way of handling unjudged documents is to remove them from the ranking and form a *condensed* sequence [21]; the more common mechanism is to assume that unjudged documents are non-relevant. Weighted-precision metrics, including RBP, provide the capacity to retain a "residual" that reflects the range of scores that could arise, with the size of that range depending on where in the ranking the unjudged documents appear, and reflecting the maximum and minimum score that could be achieved if those documents were respectively found (were they to be judged) to be relevant or not relevant [18].

If the evaluation depth k is less than the pooling depth ℓ, all documents required during scoring of the contributing runs will have been judged, and the comparison is unlikely to be biased. On the other hand, if the evaluation depth k exceeds the pooling

depth ℓ, or if runs are to be scored that were not included when the pool was formed, then comparing effectiveness scores may not be appropriate without also allowing for the unjudged documents [31]. All three options – assuming that unjudged implies not relevant; condensing the sequence; or maintaining error ranges on scores – have drawbacks.

Safe Filtering
In some special cases a filter might be matched to a corresponding ranker and be said to be *set safe to depth* k, in that it selects exactly the final k documents as a set, but not necessarily in the same final order as assigned by the ranker. That is, $F()$ and $R()$ are a "set safe to depth k" combination if, when considered as sequences,

$$R(\mathcal{D}, q, k) = R(F(\mathcal{D}, q, k), q)\,.$$

If this is the case, then the imposition of the filtering stage will not affect overall retrieval effectiveness [27].

The more usual situation is when both the filter $F()$ and the ranker $R()$ contribute to the quality of the overall results listings presented to the users of the system, the ranker in a positive sense, and the filter in a possibly negative sense. Given the large number of ways that filters (which select an unordered subset of the collection, possibly of bounded size) and rankers (which select an ordered subset of the collection, again possibly of bounded size) can be implemented, mechanisms are required that allow rankers and filters to be compared, and/or their behavior numerically quantified.

3 Multi-Phase Effectiveness

Now consider effectiveness evaluation for a system composed of distinct filtering and reranking phases. One possibility is to treat retrieval as an end-to-end process that creates a ranked sequence; and then interpret the output list $R(F(\mathcal{D}, q, k_1), q, k_2)$ exactly the same way as the result of a *Normal-Ranked* system would be. That is, if overall performance is all that matters, then assessment can be via a standard metric M, and the system can be regarded as being a single entity, despite the fact that it is assembled from distinct components.

But suppose that the measurement must focus on the usefulness of a filtering stage $F()$, so as to establish (for example) a separate effectiveness-efficiency tradeoff curve for it; or to understand what effect k_1 has on overall effectiveness. Or, as a second scenario, suppose that alternative filters $F_a()$ and $F_b()$ are provided, and that their usefulness is to be compared in the context of a specified third-phase ranker. How should the quality of a pre-ordering, or of a filter, be measured?

Recall
One possible approach is to use available relevance judgments $\mathcal{J}$ to determine the coverage of the filter. Ideally, the filter would identify every document relevant to the

query and return it as part of the (unordered) answer set; which suggests that it makes sense to measure the quality of the filter by computing

$$\mathsf{Recall}(\mathcal{D}') = \frac{|\mathcal{D}' \cap \mathcal{J}_q|}{|\mathcal{J}_q|} \tag{1}$$

where $\mathcal{D}' = F(\mathcal{D}, \mathcal{Q}, k)$ is the set of k documents the filter extracted from the collection. If a filter has a high recall according to Equation 1, then the set of documents passed to the ranking stage contains all or most of those that the ranker could possibly preference, and so retrieval effectiveness should not be degraded by the filter's presence. However, the converse is not true: a low recall score does not necessarily give rise to a low score from the final effectiveness metric, since the final metric might be strongly top-weighted, and there might be many relevant documents for the filter to select amongst. Unfortunately, recall requires relevance judgments, limiting to applicability to queries which have judgments available.

Overlap

If two filters are to be compared, the outputs $\mathcal{D}^a = F_a(\mathcal{D}, q, k)$ and $\mathcal{D}^b = F_b(\mathcal{D}, q, k)$ can be tested against each other, and an *overlap coefficient* computed, an approach that avoids the need for relevance judgments. If it is assumed that the eventual evaluation metric M is insensitive to small changes in the sets, and $\mathcal{D}_a$ and $\mathcal{D}_b$ are close to being the same, then substituting $F_a()$ by $F_b()$ in a *Multi-Level-Ranked* system should be plausible. That is, the difference between the sets generated by two filters might in some situations be a valid surrogate for the evaluation of the eventual metric. One possible overlap coefficient is given by the Jaccard similarity coefficient:

$$\mathsf{Overlap}(\mathcal{D}^a, \mathcal{D}^b) = \frac{|\mathcal{D}^a \cap \mathcal{D}^b|}{|\mathcal{D}^a \cup \mathcal{D}^b|}\,, \tag{2}$$

which is zero when the two sets are disjoint, and is one if and only they are identical. A variation on the overlap computation replaces the denominator of Equation 2 by $\min\{|\mathcal{D}^a|, |\mathcal{D}^b|\}$. Other coefficients are possible, including ones that are asymmetric and compute the fraction of one set that is present in the other as a *coverage ratio*.

Overlap-based approaches are effective if the two sets are of comparable size, and if the differences between them are small. They also have the considerable benefit of not requiring that relevance judgments be available. But in the case of ranking systems it is also desirable to be able to compare the outcomes obtained when the two sets might be of quite different sizes. For example, one filter might generate a subset of the collection that is a tenth the size of another; what matters in this case is what change arises in the downstream effectiveness score generated by the final metric. Given that most metrics are top-weighted, the fact that $\mathcal{D}^a$ and $\mathcal{D}^b$ differ in size may be less important than Equation 2 might suggest.

Rank-Biased Overlap

Top-weighted set overlap computations applicable to ordered sequences have also been presented. Webber et al. [29] describe a method they call Rank-Biased Overlap, or RBO, which computes a top-weighted overlap score between two sequences. A

user model akin to that of the rank-biased precision effectiveness metric is presumed, in which the reader scans from the top of the two lists, determining the depth that they will view the two sequences to according to a geometric probability distribution governed by a parameter p. The value of p is the conditional probability of the user stepping from depth i to depth $i+1$ in the pair of sequences. When p is close to one, the user is "patient", and expects to examine a relatively long prefix of the two sequences; when p is smaller, the user is modeled as only examining a relatively short prefix. The RBO score between the two sequences is then computed as a weighted average of overlap ratios:

$$\mathsf{RBO}(\mathcal{D}^a, \mathcal{D}^b) = (1-p)\sum_{i=1}^{\infty} p^{i-1} \cdot \frac{|\mathcal{D}^a_{1..i} \cap \mathcal{D}^b_{1..i}|}{i}\,, \tag{3}$$

where $\mathcal{D}_{1..i}$ refers to the first i elements in ordered sequence $\mathcal{D}$.

Like Overlap, RBO has the advantage of not requiring that relevance judgments be available, and hence experiments can be carried out automatically over large sets of sample queries, allowing high levels of confidence in the measured outcomes. But note that RBO can only be used if the filter $F()$ generates an ordered sequence of documents – it cannot be applied if the filter produces a set as its output. Nor does it exactly match any particular eventual effectiveness metric, although there is strong relationship between RBP and RBO. Being metric-agnostic can be thought of as being a strength of several of these approaches, but also as a weakness.

Another top-heavy measure is Dice Top (DT), introduced by Macdonald et al. [17]. Macdonald et al. primarily use DT to characterize features which improve effectiveness in a learning to rank model, but it could also be used to compare the similarity between any two result sets.

End-to-End Effectiveness

Another way of measuring the effectiveness of a filter is to determine the extent to which the insertion of the filter alters the metric score compared to the pure ranking scheme, assuming that the filter extracts a subset $\mathcal{D}' = F(\mathcal{D}, q, k)$ of the collection for subsequent processing. That is, both the with-filter and without-filter systems are regarded as being "end to end", and a statistical test is carried out using pairs

$$\langle \mathsf{M}(R(\mathcal{D}, q), \mathcal{J}_q), \mathsf{M}(R(\mathcal{D}', q), \mathcal{J}_q) \mid q \in \mathcal{Q} \rangle$$

A comparison using this methodology might, for example, seek to demonstrate that the introduction of the filter allows rejection of the hypothesis that "the filter reduces the effectiveness scores attained by ϵ or more", for some small ϵ specified as part of the experimental design. Like the recall-based method (Equation 1), this approach requires that relevance judgments be available.

Maximized Effectiveness Difference (MED)

Recent work by Tan and Clarke [25] offers a further alternative, and provides the basis for the evaluation in our experiments. Given two documents sequences $\mathcal{D}^a$ and $\mathcal{D}^b$ and a chosen metric M, a set of judgments $\mathcal{J}'$ is identified that maximally separates

the M-scores for the two orderings. The resulting value is the *maximum effectiveness difference* with respect to M:

$$\mathsf{MED}_{\mathsf{M}}(\mathcal{D}^a, \mathcal{D}^b) = \max_{\mathcal{J} \subseteq \mathcal{D}^a \cup \mathcal{D}^b} |\mathsf{M}(\mathcal{D}^a, \mathcal{J}) - \mathsf{M}(\mathcal{D}^b, \mathcal{J})| \,. \tag{4}$$

The methodology presented by Tan and Clarke allows a set of partial judgments $\mathcal{J}' \subseteq \mathcal{J}$ to be specified as an additional constraint, plus a list of documents that may not be members of $\mathcal{J}$. Computation of $\mathsf{MED}_{\mathsf{M}}$ when M is recall-independent is straightforward [25], even when these additional constraints are added. To see the usefulness of this concept for evaluating two-level retrieval systems, suppose that $\mathcal{D}^a = R(F(\mathcal{D}, q, k), q)$ and that $\mathcal{D}^b = R(\mathcal{D}, q)$ for some query q and complete collection $\mathcal{D}$. That is, suppose that $\mathcal{D}^b$ is the sequence generated by ranker $R()$ without a filtering step, and $\mathcal{D}^a$ is the sequence generated by the ranking when k documents are selected by the filter $F()$. Then $\mathsf{MED}_{\mathsf{M}}(\mathcal{D}^a, \mathcal{D}^b)$, as defined by Equation 4, represents the maximum possible degradation in measured effectiveness (according to metric M) that results from the insertion of $F()$ in to the processing pipeline.

For example, suppose that some query gives rise to the ranking

$$\mathcal{D}^b = \langle 20, 45, 17, 11, 33, 29, 18, 56, 72, 91, 54, 83, 22, \ldots \rangle$$

and that the documents allowed by the filter give rise to the subsequence

$$\mathcal{D}^a = \langle 20, 45, 17, 33, 29, 56, 72, 91, 54, 22, \ldots \rangle \,.$$

Suppose further that the metric being used to measure system effectiveness is rank-biased precision RBP with the parameter $p = 0.8$. The definition of MED, and the definition of RBP, mean that $\mathsf{MED}_{\mathsf{RBP}}$ arises when the documents that are common to the two sequences are deemed to be non-relevant, making $\mathsf{RBP}(\mathcal{D}^a) = 0$; and the documents that appear in $\mathcal{D}^b$ only are all deemed to be fully relevant, that is, with $r_i = 1$. In the case of the example rankings, that gives rise to the two computations

$$\begin{aligned}
\mathsf{RBP}0.8(\mathcal{D}^b) &= \mathsf{RBP}0.8(\langle 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, \ldots \rangle) \\
&= 0.2(0.8^3 + 0.8^6 + 0.8^{10}) \\
&= 0.176 \\
\mathsf{RBP}0.8(\mathcal{D}^a) &= \mathsf{RBP}0.8(\langle 0, 0, 0, 0, 0, 0, 0, 0, \ldots \rangle) \\
&= 0.0 \,.
\end{aligned}$$

So, $\mathsf{MED}_{\mathsf{RBP}0.8}(\mathcal{D}^a, \mathcal{D}^b) = 0.176$, and is an upper bound on the effectiveness loss when the filtered ranking $\mathcal{D}^a$ is used to approximate the full ranking $\mathcal{D}^b$ when the runs are scored using RBP with $p = 0.8$. Note that no judgments are required in this computation; it is derived solely from the full and filtered sequences of document numbers. Moreover, it can be based on any recall-independent effectiveness metric, meaning that in many situations it will be possible to employ the metric that is used to measure end-to-end system effectiveness.

Tan and Clarke [25] indicate that the MED family of metrics was directly inspired by the wish to generalize RBO. In the same way that there is a close association between RBP and RBO, MED provides a general procedure for deriving a rank similarity measure from a (recall independent) effectiveness measure. Tan and Clarke [25]

Algorithm 1 Boolean Intersection Algorithm (Boolean)

INPUT: A list of q ordered sets $S_1 \ldots S_{|q|}$.
OUTPUT: An ordered set of documents R.

```
1: set R ← ∅
2: initialize each S_i with its least element as its candidate
3: SORT(S_1 ... S_|q|) based on candidate
4: set x ← the candidate from S_|q|
5: while x is defined do
6:     if the candidate from S_1 is equal to x then
7:         APPEND(R, x)
8:     set x ← SUCCESSOR(S_|q|, x)
9:     for i = 1 to |q| − 1 do
10:        F-SEARCH(S_i, x)
11:    SORT(S_1 ... S_|q|) based on candidate
12:    set x ← the candidate from S_|q|
   return R
```

show that $\mathsf{MED}_{\mathsf{RBP}}$ is highly correlated with RBO, supporting our choice of the MED family for measuring filter-stage effectiveness.

4 Filters

We next describe the various filters that might be employed in a staged retrieval system, and then in Section 5 present validation experiments that confirm the relationship between $\mathsf{MED}_{\mathsf{M}}$ and the effectiveness loss (relative to metric M) that arises when filtering is employed. Section 6 steps away from the use of relevance judgments, and expands the experimentation to a large query sequence. A key goal is to demonstrate that the $\mathsf{MED}_{\mathsf{M}}$ approach yields usable information about filtering-stage effectiveness, but we are also able to compare and contrast the relative efficiency of the various filtering mechanisms.

Pure Boolean Conjunction
Conjunctive Boolean queries have been extensively studied by the IR research community [9, 14]. Using hybrid postings lists that combine bitvectors and compressed postings can significantly improve the efficiency in terms of both time and space. Efficiency can be further improved using document reordering [14]. However, the fastest of these approaches cannot use the same inverted index for ranked querying and Boolean filtering.

The experiments reported below make use of two different Conjunctive Boolean Algorithms. The first variant is the `#band` operator in Indri[1] which performs a full Boolean pass over the index. We also implement a second variant based on the Adaptive Intersection Algorithm [10, 11] which is easily amenable to a block-compressed inverted index, and can be processed in a document-at-a-time manner similar to WAND [5]. The algorithm used is shown in Algorithm 1. First the smallest document

[1] `http://www.lemurproject.org/indri/`

identifier is selected as the target identifier x, and all lists are sorted in increasing order by the current document id cursor. If the first cursor and last cursor are equal, the identifier is appended to the results list. Next, the target identifier x is set as the successor from the last postings list, and the cursors in every list are forwarded to x, or its successor. When the end of any postings list is reached, the algorithm terminates.

In contrast to the term-at-a-time based intersection algorithm SvS used by Asadi and Lin [3], this intersection algorithm is directly amenable to document-at-a-time scoring algorithms such as WAND. If a full conjunctive Boolean result set is desirable, using a SvS approach is the most efficient processing scheme in practice [9]. However, SvS does not easily support early termination after k items have been identified, but a WAND-like traversal of the lists does.

Boolean Conjunction with Static Scores

The effectiveness of Boolean conjunction as a filtering stage can be enhanced by the inclusion of a static scoring process. This is commonly achieved by reordering the document collection using a static score before constructing the index. Common static scoring regimes include spam score [8], PageRank [19], or document length. If k documents are to be retrieved, the first k that are identified at step 7 of Algorithm 1 are the ones returned – they will have the highest static score amongst all documents that contain all of the query terms.

It is also possible to integrate on-the-fly static scoring into Algorithm 1 by adding a min-heap structure as part of the **APPEND** operation at step 7. The advantage of this arrangement is that the same index can be used to evaluate a range of different static scores; the drawback is that processing must be completed across all documents, and execution cannot be terminated once k matching items have been determined. Unless otherwise noted, we assume here that the collection is pre-ordered according to static score, and that the first k that contain all of the query terms according to a single static scoring method are the ones passed through to the ranking stage.

WAND and MaxScore

A third filtering option to use is an efficient bag-of-words scoring algorithm such as WAND (Weak AND, or Weighted AND) [5] or MaxScore [24, 27]. Both algorithms can be used with a variety of ranking functions, including the Okapi BM25 computation and methods based on language models. Macdonald et al. [16] document the advantages and disadvantages of using a bag-of-words filtering step for second stage learning to rank algorithms. Asadi and Lin [3] and Wang et al. [28] also investigate various trade-offs with bag-of-words filtering in multi-stage retrieval architectures.

Aggressive WAND

The WAND algorithm is defined as follows. Given a list of Boolean indicator variables $x_1, x_2, \ldots, x_q$ with $x_i \in \{0, 1\}$ indicating whether or not the ith term in the query appears in the current document, a list of positive weights $U_1, U_2, \ldots, U_q$ with U_i indicating the largest score contribution that the ith term in the query gives rise to

in any document in the collection, and a threshold t,

$$\text{WAND}(x_1, U_1, \ldots, x_q, U_q, t) \equiv \left(\sum_{1 \le i \le q} x_i U_i \ge t \right) .$$

Therefore, WAND is a Boolean indicator that is 0 when there is no possibility that the score for the current document can exceed t, and 1 if and only if it can. Let $s_{\min}$ be the value of the k th largest document score that has been generated to date in the computation. In the *aggressive WAND* strategy the indicator

$$\text{WAND}(x_1, U_1, \ldots, x_q, U_q, \theta \cdot s_{\min})$$

is used to provide guidance as to whether the current document should have its score computed.

If $\theta = 1$ then a standard evaluation takes place, and every document that *might* be able to enter the heap will be scored, with the overall top-k items guaranteed to be returned. When $\theta > 1$, the barrier is raised, and fewer documents are scored, on the presumption that any documents that have scores that are close to the heap's lower limit at the time they are first encountered are likely to either not make it in to the heap at all, once their scores are computed; or even if they do make it in to the heap, to be evicted again during the course of the remaining processing. The risk is that there are more documents with WAND estimates a little above $s_{\min}$ at the time they are processed, plus have actual similarity scores also greater than $s_{\min}$; and that the computation as a whole has stabilized, so that $s_{\min}$ does not rise very much subsequently, and hence that these documents would enter the heap and then remain there. The larger θ, the greater the risk that such errors might occur. In the limit, if $\theta = \infty$, only the first k documents evaluated get added to the heap, and no other documents will be scored. The experiments reported in the next section consider several θ values greater than one, exploring the balance between filtering effectiveness and processing efficiency. In their paper describing WAND, Broder et al. [5] also note that θ might be used as a parameter to accelerate searching while risking the score-safe nature of the computation. One of the contributions of this paper is a detailed exploration of the usefulness of the aggressive WAND approach.

Scored Boolean WAND

Another variation on WAND is to only Okapi-score documents which are a full conjunctive match, and contain all of the query terms. This is implemented as a modification to the Adaptive Intersection algorithm shown in Algorithm 1, with the **APPEND** operation at step 7 replaced by first an evaluation of the Okapi score of that document, then a test against the entry threshold for a min-heap containing the k largest scores accumulated for documents processed to this moment, and then finally, if the new score is greater, suitable heap operations to update the state of the heap. This variation on WAND has a similar effectiveness profile to the `#band` operator in Indri, which is not optimized for performance. In general, θ has less of an effect on scored Boolean WAND than it does in the standard Okapi computation, since the fact that all of the query terms must always appear means that the sum of the upper bound scores in pivot evaluation is always relatively high.

Other Approaches
Wu and Fang [30] present another compromise between Scored Boolean and Static Boolean. The key idea is to use a decision tree with per term IDF values to prioritize the matches. The efficiency and effectiveness horizon of this approach lies between the two methods explored in this work. Asadi and Lin [3] also investigate the impact of disjunctive and conjunctive WAND when used as a filtering step for a learning-to-rank second stage derived from Lambda Mart. Asadi and Lin conclude that conjunctive WAND was the best compromise as the final effectiveness results were statistically indistinguishable from the disjunctive WAND filter. But they do not consider aggressive WAND variations, or the impact of alternative second stage runs. Finally, Wang et al. [28] present a cascaded learning-to-rank approach to iteratively shrink the candidates evaluated. The approach is efficient and effective in practice, but relatively difficult to separate into distinct stages for independent evaluation.

5 Validation Experiments

Experimental Setup
In order to validate the use of MED to establish a correlation between filtering effectiveness and end-to-end effectiveness, a detailed evaluation using judged topics has been undertaken, reinforcing the similar experiments carried out by Tan and Clarke [25]. All of the experiments described in this section and the next section use Part B of the ClueWeb 2009 collection (CW09B); in this section (only), the 2010 and 2012 Ad-Hoc queries from the TREC Web Track are also used (50 queries per year), together with the relevance judgments associated with them. We refer to these collections of topics as AH2010 and AH2012 respectively.

To ensure our effectiveness results are consistent with commonly used search engines, we use version 5.6 of Indri. For Indri Boolean runs, we use the `#band` operator; for Language model runs we use the default parameters; and for Okapi BM25 runs we use $k_1 = 0.9, b = 0.4, k_3 = 0$, since these values consistently give better results than the default parameters [26]. All runs employ stemming using the Krovetz stemmer, and the default stopwords list is used when stopping is enabled. Runs without stopping were also carried out.

Results
Table 1 summarizes the filter-stage combinations used for the validation and efficiency experiments. For all of the filter stage runs, we investigated four different types of indexes: Unstopped and Unpruned; Stopped and Unpruned; Unstopped and Pruned; and, fourthly, Stopped and Pruned. The pruned indexes were restricted to only include documents with spam scores greater than 70, using the *Fusion* scores described by Cormack et al. [8]. Each of the listed filter-stage options was evaluated to depths k of 20, 50, 100, 200, 500, 1,000, 2,000, 5,000, and 10,000.

Two different orderings by PageRank were also used. The first version used unnormalized raw PageRank scoring; the second version used the binned log probability values. Both versions used are freely available on the ClueWeb09 Wiki[2].

[2] `http://boston.lti.cs.cmu.edu/clueweb09/wiki/tiki-index.php?page=PageRank`

Stage One Run Methods	Runs
Bag-of-words, language model, Dirichlet smoothing (Indri), × stopped or not, × spam-pruned or not	4
Bag-of-words, BM25, $k_1 = 0.9, b = 0.4$ (Indri), × stopped or not, × spam-pruned or not	4
Bag-of-words, aggressive WAND, BM25, $k_1 = 0.9, b = 0.4$, stopped, not spam-pruned, $\theta \in \{1.0, 1.02, 1.05, 1.1, 1.2, 1.5, 2.0\}$	7
Scored Boolean, WAND, BM25, $k_1 = 0.9, b = 0.4$, only used in Section 6	0
Scored Boolean, Indri `#band`, × stopped or not, × spam-pruned or not	4
Boolean, static pagerank as raw scores, × stopped or not, × spam-pruned or not	4
Boolean, static pagerank as bucketed log probability prior, × stopped or not, × spam-pruned or not	4
Boolean, static fusion spam score, × stopped or not, × spam-pruned or not	4

Table 1: Summary of filter stage mechanisms explored. A total of four different Indri indexes were constructed from CW09B for each filter type: an unstopped index, a stopped index, an unstopped index containing documents with spam scores greater than 70 only, and a stopped index containing documents with spam scores greater than 70 only. In the three static Boolean approaches, document length was used as a secondary key. A total of $6 \times 4 + 7 = 31$ filtering mechanisms were tested in the experiments reported in this section.

Collection	TREC Run Identifier	Depth
MQ2009	`uogTRMQdph40`	1,000
AH2010	`IvoryL2Rb`	10,000
AH2012	`DFalah121A`	Variable depth ($d = 11$ – 2,009)

Table 2: Second stage runs used as reference points.

For gold standard final ranking stage runs, we employed the top runs submitted to TREC for the corresponding query sets. These are listed in Table 2. For AH2010 and AH2012, the selected runs achieved the best performance over Part B of the ClueWeb 2009 collection under the primary measure used for reporting track results (ERR@20).

The top plot of Figure 1 shows the correlation between measured $\mathsf{MED}_{\mathsf{RBP0.95}}$ and measured AP. The bottom plot of Figure 1 shows the correlation between measured $\mathsf{MED}_{\mathsf{DCG20}}$ and measured AP. These plots cover a suite of 31 different filter stages, and nine different depths k for each first stage. For this experiment, the measures are computed over the 50 queries of the AH2010 collection, with `IvoryL2Rb` forming the gold standard final-stage ranked run, assumed to reorder the documents provided by the filter phase according to the similarity score computed by the original TREC run. The performance of the final stage with no early-stage filtering is shown by the line at 0.1358. The diagonal lines demonstrate the clear inverse relationship between end-to-end AP and, respectively, $\mathsf{MED}_{\mathsf{RBP0.95}}$ and $\mathsf{MED}_{\mathsf{DCG20}}$ In this figure the smooth transition from good overall performance to bad overall performance suggests that the good filter-phase options are well-matched to the final stage computation that is being used.

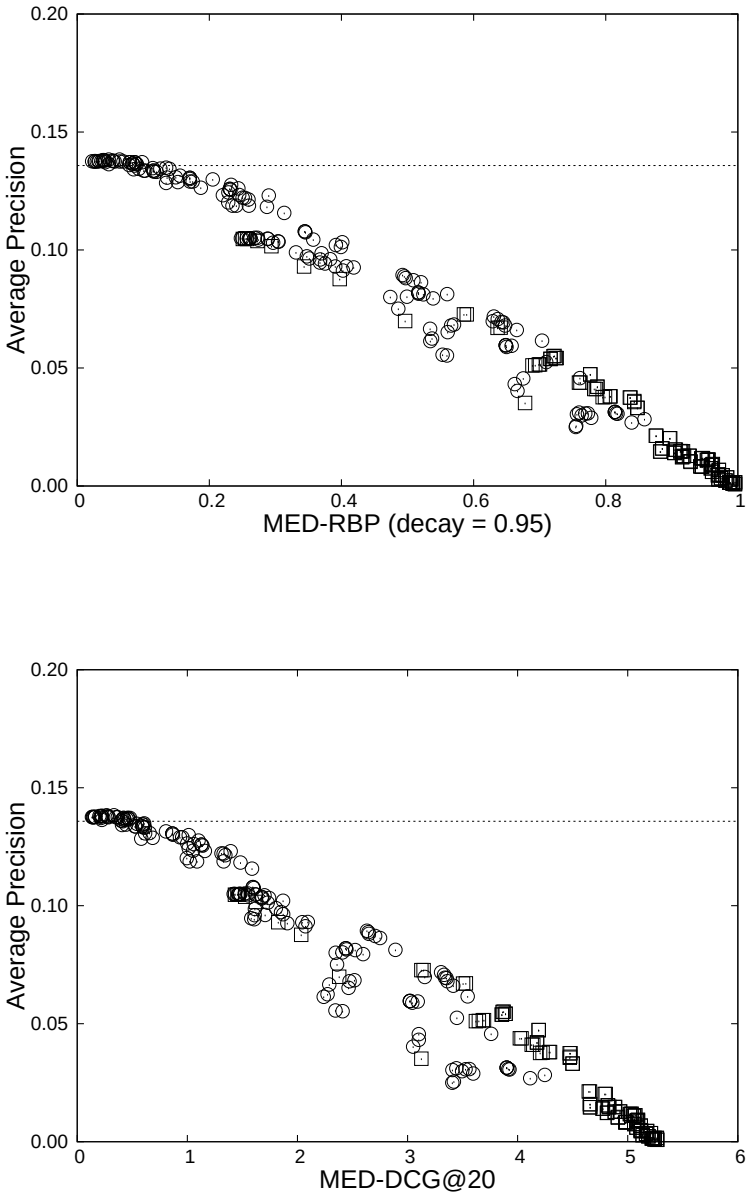

Fig. 1: Correlations between $\mathsf{MED}_{\mathsf{RBP0.95}}$ and $\mathsf{MED}_{\mathsf{DCG20}}$ and measured AP using the AH2010 collection, 50 queries, and `IvoryL2Rb` as the final ranking stage. Measured AP values are computed using standard methods and software (`trec_eval` and TREC 2010 adhoc relevance judgements). Kendall's τ correlation coefficients are -0.931 and -0.910 respectively. Each of the 279 points represents one of 31 distinct filter stages employed at one of nine filter-stage depths, ranging from $k = 20$ to $k = 10{,}000$. The dashed line indicates the performance of the final stage with no early-stage filter. Circles indicate early-stage filters that use query-dependent ranked retrieval; squares indicate early-stage filters that use Boolean retrieval with static ranks.

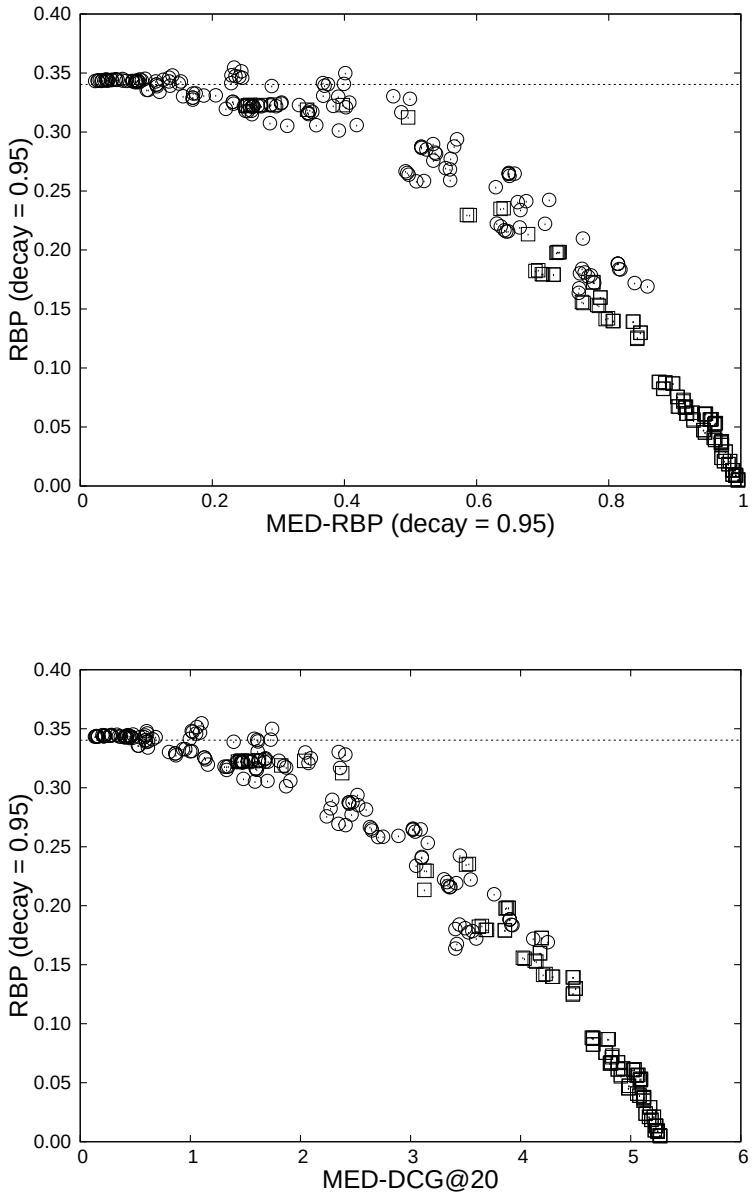

Fig. 2: Correlations between $\mathsf{MED}_{\mathsf{RBP0.95}}$ and $\mathsf{MED}_{\mathsf{DCG20}}$ and measured RBP0.95 using the AH2010 collection, 50 queries, and `IvoryL2Rb` as the final ranked stage. Measured RBP0.95 values are computed using standard methods and software (`rbp_eval-0.5` and TREC 2010 adhoc relevance judgements). Kendall's τ correlation coefficients are -0.875 and -0.882 respectively. Each of the 279 points represents one of 31 distinct filter stages employed at one of nine filter-stage depths, ranging from $k = 20$ to $k = 10{,}000$. The dashed line indicates the performance of the final stage with no early-stage filter. Circles indicate early-stage filters that use query-dependent ranked retrieval; squares indicate early-stage filters that use Boolean retrieval with static ranks.

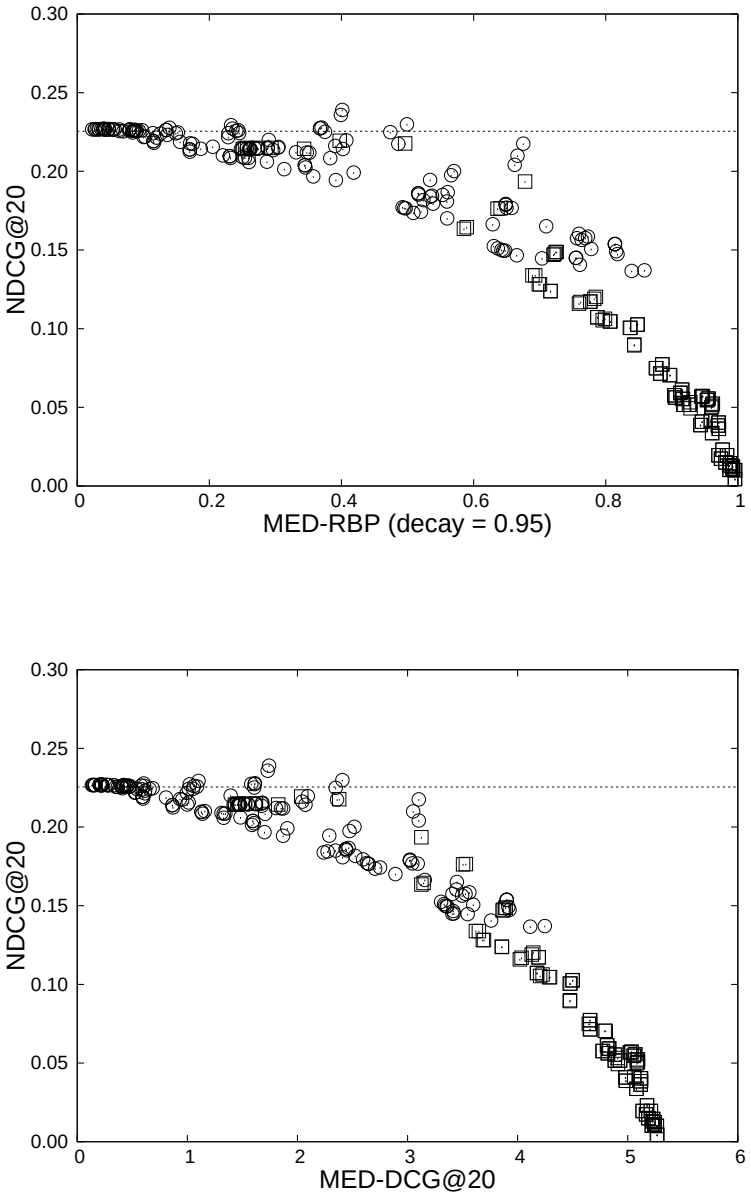

Fig. 3: Correlations between $\mathsf{MED}_{\mathsf{RBP0.95}}$ and $\mathsf{MED}_{\mathsf{DCG20}}$ and measured NDCG@20 using the AH2010 collection, 50 queries, and `IvoryL2Rb` as the final ranked stage. Kendall's τ correlation coefficients are -0.837 and -0.854 respectively. Measured NDCG@20 values are computed using standard methods and software (`gdeval.pl` and TREC 2010 adhoc relevance judgements, with five relevance grades). Kendall's τ correlation coefficients are -0.837 and -0.854 respectively. Each of the 279 points represents one of 31 distinct filter stages employed at one of nine filter-stage depths, ranging from $k = 20$ to $k = 10{,}000$. The dashed line indicates the performance of the final stage with no early-stage filter. Circles indicate early-stage filters that use query-dependent ranked retrieval; squares indicate early-stage filters that use Boolean retrieval with static ranks.

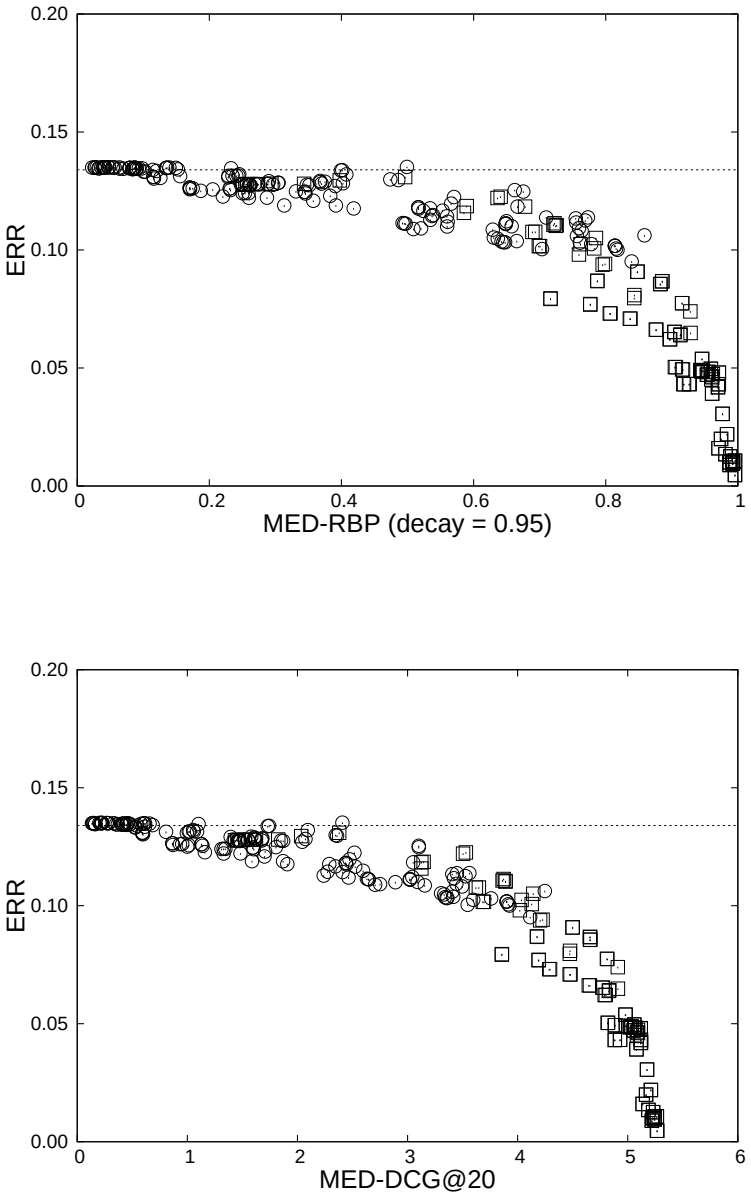

Fig. 4: Correlations between $\mathsf{MED}_{\mathsf{RBP0.95}}$ and $\mathsf{MED}_{\mathsf{DCG20}}$ and measured ERR using the AH2010 collection, 50 queries, and `IvoryL2Rb` as the final ranked stage. Measured ERR values are computed using standard methods and software (`gdeval.pl` and TREC 2010 adhoc relevance judgements, with five relevance grades). Kendall's τ correlation coefficients are -0.835 and -0.836 respectively. Each of the 279 points represents one of 31 distinct filter stages employed at one of nine filter-stage depths, ranging from $k = 20$ to $k = 10{,}000$. The dashed line indicates the performance of the final stage with no early-stage filter. Circles indicate early-stage filters that use query-dependent ranked retrieval; squares indicate early-stage filters that use Boolean retrieval with static ranks.

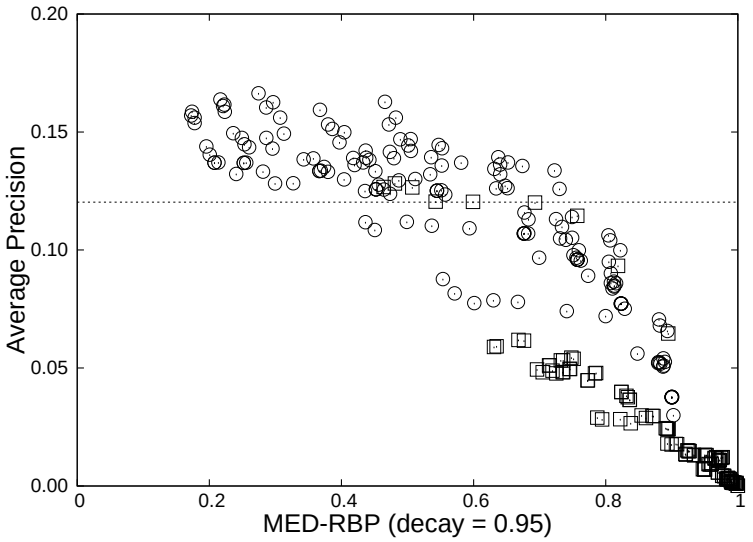

Fig. 5: Correlation between $\mathsf{MED}_{\mathsf{RBP0.95}}$ and measured AP using the AH2012 collection, 50 queries, and `DFalah121A` as the final ranked stage. Measured AP values are computed using standard methods and software (`trec_eval` and TREC 2010 adhoc relevance judgements). Kendall's τ correlation coefficient is -0.804. Each of the 279 points represents one of 31 distinct filter stages employed at one of nine filter-stage depths, ranging from $k = 20$ to $k = 10{,}000$. The dashed line indicates the performance of the final stage with no early-stage filter. Circles indicate early-stage filters that use query-dependent ranked retrieval; squares indicate early-stage filters that use Boolean retrieval with static ranks.

Early-stage filters with $\mathsf{MED}_{\mathsf{RBP0.95}}$ values under 0.15 (or $\mathsf{MED}_{\mathsf{DCG20}}$ values under 0.75) have little impact on end-to-end effectiveness as measured by AP. These early-stage filters are based on bag-of-words ranked retrieval methods (for example, BM25) and large retrieval depths (for example, $k = 10{,}000$). Their low MED values suggest that they are providing the final stage with an appropriate set of documents for re-ranking, even though the reranking stage incorporates ranking methods and features not found in these filter stages, including a learned ranker and term proximity [12]. With smaller values for k, the MED values of these filters increase as their end-to-end performance decreases. Filter stages with $\mathsf{MED}_{\mathsf{RBP0.95}}$ values over 0.90 (or $\mathsf{MED}_{\mathsf{DCG20}}$ values over 4.5) use Boolean retrieval and smaller values of k.

While filter stages with low MED values provide essentially the same end-to-end effectiveness, their low MED values serve to differentiate them, with lower values suggesting a better fit with the final stage ranker. The AP values are not able to provide this differentiation, possibly due to the presence of unjudged documents, which are assumed to be non-relevant. By treating the output of the final stage as a gold standard and directly measuring the impact of filtering in an early stage, we avoid the complexities introduced by these unjudged documents.

Figure 2 shows equivalent plots for measured RBP, a shallower effectiveness metric than AP; Figure 3 shows equivalent plots for measured NDCG@20, using graded relevance assessments; Figure 4 shows equivalent plots for measured ERR, also using graded relevance assessments. The flattening towards the left of these plots suggests that these metrics are more sensitive than AP to distinctions between filtering stages, although the same general trends are observed, particularly for higher values of MED.

While we do not show plots, similar correlations also arise when other collections and effectiveness measures. Effectiveness measures focused on precision at low ranks, particularly ERR, are less sensitive to the choice of early-stage rankers, with wide ranges of MED values corresponding to similar measured end-to-end effectiveness. This relative tolerance is as expected – shallow end-to-end metrics can be satisfied by rankings with more divergence than can deep end-to-end metrics, since the second ranking stage can still find the documents it needs, even if further down the first-phase's output, and get them to the top of a ranking for the shallow metric to benefit from. That is, if a deep-weighted MED evaluation is carried out as a predictor of a shallow end-to-end metric, higher MED scores can be regarded as being acceptable.

Note that in each of Figures 2, 3, and 4 there are filtering combinations evident that have higher effectiveness scores according to the target metric than the reference run being used as a benchmark for measured performance. The MED score computed for a particular pairing of runs provides a guarantee that one run cannot score more than a certain amount lower than the other according to the given metric, but cannot provide any guidance as to whether one of the runs outperforms the other. The latter relationship can only be established through the use of comprehensive relevance judgments, in conjunction with a statistical test for significance. In any case, if filtering improves the performance of the second-stage ranker, this highlights a protential problem with that stage, which should be investigated from that perspective.

Figure 5 shows a more complex situation. For this experiment, the measures are computed over the 50 queries of the AH2012 collection. The run `DFalah121A` forms the final ranking stage, chosen for its excellent performance on the 50 queries of AH2012 when measured by ERR [1]. This run took unusual approaches to indexing and query processing, making substantial use of external resources.

There are a number of points to be noted. First, none of the early-stage filters generates the documents that the final stage is wishing to see in the top-ranked positions, and the result is that $\mathsf{MED}_{\mathsf{RBP0.95}}$ is never less than about 0.2. Second, this discrepancy does not greatly harm the measured AP score for the combined run. Indeed, the best early-stage filters actually increase AP substantially above the score of the ranking stage without filtering, as shown by the line at 0.1203. These filtering stages are essentially forcing the ranking stage to improve under the recall-based AP effectiveness measure, by restricting the set of the documents it can rank. Third, the correlation between $\mathsf{MED}_{\mathsf{RBP0.95}}$ and measured AP is weaker than in Figure 1, again suggesting that these early stage filters may be a poor fit with this final stage.

While the improvements to AP and the weaker correlation would not be visible without relevance judgments, the poor fit would be noticeable from the lack of low $\mathsf{MED}_{\mathsf{RBP0.95}}$ values. This example emphasizes an important limitation of our approach. We assume that the final ranking stage provides an acceptable gold standard

for comparing early-stage filters. Poor end-to-end performance cannot be detected by the application of MED.

6 Judgment-Free Measurement of Effectiveness Tradeoffs

We now explore the relationship between filter-stage retrieval effectiveness and efficiency using a large query log, and $\mathsf{MED}_{\mathsf{RBP}}$ as a measure for filtering effectiveness.

Experimental Resources
For the efficiency experiments, $40{,}000$ queries from the 2009 Million Query Track were used with a stopped and unpruned CW09B index. We refer to this combination as MQ2009. The `uogTRMQdph40` system is used as the gold standard, as it represents the top-scoring system (when measured over the small subset of the queries that were evaluated) that returned runs for all of the large set of $40{,}000$ MQ2009 queries. Algorithms were implemented in C++ and compiled with gcc 4.8.1 using –O3 optimizations; and experiments were run on a 24 core Intel Xeon E5-2630 running at 2.3 GHz using 256 GB of RAM. All efficiency runs are reported as the median of the per query execution times of a single execution of a complete stream containing one instance of each query in MQ2009, executed entirely in-memory. Postings lists are stored compressed using the FastPFOR library [15], with skipping enabled. As discussed in the previous section, each of the listed filter-stage options was evaluated to depths k of 20, 50, 100, 200, 500, $1{,}000$, $2{,}000$, $5{,}000$, and $10{,}000$.

Filter Stage Effectiveness
Figures 6 and 7 demonstrate typical outcomes. In each of the two plots, $\mathsf{MED}_{\mathsf{RBP0.95}}$ is used as the effectiveness assessment, and the depth k of the filter-phase output is plotted on the horizontal axis. The reference run in both cases came from the University of Glasgow (run `uogTRMQdph40`).

In Figure 6, which makes use of an Okapi similarity computation for the filtering phase, there is a clear inverse relationship between k and the upper bound on quality. Passing as few as $1{,}000$ documents to the final ranking phase is sufficient to obtain a $\mathsf{MED}_{\mathsf{RBP0.95}}$ score of 0.2, which is suggestive of good quality final outcomes (Figure 1). Nonetheless, Figure 6 shows a wide variance of $\mathsf{MED}_{\mathsf{RBP0.95}}$ scores across the set of queries. While the median (marked by the overlaid line) shows a clear trend, there are upper outlier queries for which even deep filtering to depth $10{,}000$ is still inadequate. For example, "`buying first home`" (query #50066) has a $\mathsf{MED}_{\mathsf{RBP0.95}}$ value of one. Okapi scores for the top $10{,}000$ documents fall into the narrow range $[3.75338, 3.86595]$, suggesting that the collection includes many documents containing these terms in similar proportions. None of these $10{,}000$ documents are ranked above $1{,}000$ by the final ranking stage.

On the other hand, Figure 7 shows that filtering based on Boolean conjunction and a static PageRank score is unlikely to give good effectiveness in a multi-stage retrieval system, regardless of the number of documents identified by in the filtering stage. That is, the presence of all of the query terms alone is of limited usefulness towards identifying potentially relevant documents, even when coupled with PageRank as an

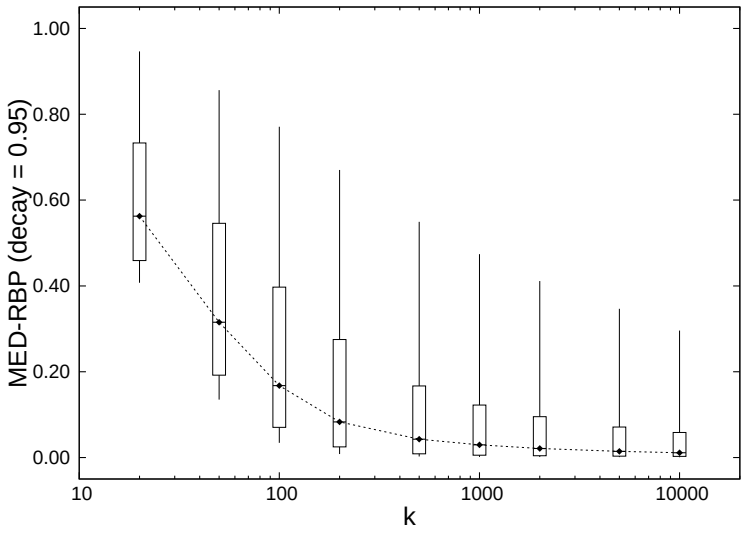

Fig. 6: Okapi ranking as early stage filter to retrieve k documents, then using `uogTRMQdph40` as the final ranking stage to order them using the stopped unpruned CW09B collection. Boxplots show $\mathsf{MED}_{\mathsf{RBP0.95}}$ values over the 40,000 MQ2009 queries, with boxes extending from the first to the third quartile and with whiskers extending to the 10 th and 90 th percentiles. Median values are connected by lines.

ordering criteria. Again, however, there are outlier queries that are at odds with this overall trend. The query "`basscat boats`" (#36318) provides one example. Only 102 documents contain both terms, so that even at depth 100 the $\mathsf{MED}_{\mathsf{RBP0.95}}$ score falls close to zero.

Across the set of filter-stage methods explored, there was a consistent separation – they either gave plots that corresponded to Figure 6, or they gave plots similar to Figure 7. The differentiating criteria was very simple. Methods based on Boolean conjunction, even when combined with a static pre-ordering criteria such as PageRank, gave uniformly poor results, whereas methods that involved an element of ranking performed as shown in Figure 6, with a gradated trade-off between k and measured effectiveness. Within the latter group there were small difference in $\mathsf{MED}_{\mathsf{RBP0.95}}$. For example, aggressive WAND processing with $\theta = 2.0$ was less effective (or rather, gave higher $\mathsf{MED}_{\mathsf{RBP0.95}}$ values) than when smaller values of θ were used. This outcome is consistent with our validation experiments in the previous section (Figures 1, 2, and 5) where these Boolean filters (marked as squares in the plots) exhibited poor performance.

Filtering Stage Evaluation Cost

Figure 8 shows the flip side of those effectiveness results, plotting filter-stage evaluation time as a function of k. Strict conjunctive Boolean evaluation is extremely fast,

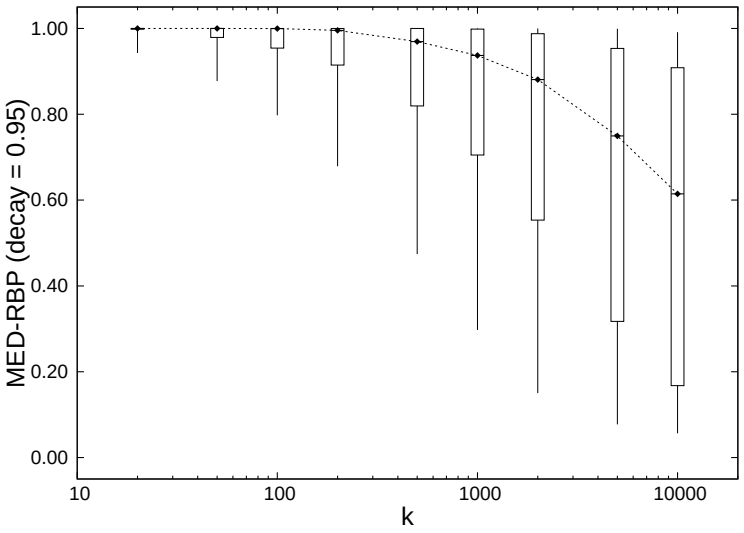

Fig. 7: Boolean conjunction with the collection pre-ordered by PageRank as the filter stage to retrieve k documents, then using `uogTRMQdph40` as the final ranked stage to order them. Boxplots show $\mathsf{MED}_{\mathsf{RBP0.95}}$ values over the 40,000 MQ2009 queries, with boxes extending from the first to the third quartile and with whiskers extending to the 10 th and 90 th percentiles. Median values are connected by lines.

partly because (using Algorithm 1) the processing required per answer document is modest, and partly because filter-stage evaluation can be abandoned just as soon as k matching documents have been identified (an option not shown in Algorithm 1, but trivial to implement). The execution time trends shown for the WAND variants are reflected in operation counts for document scoring, heap insertions, and so on, confirming the basis of the time savings. The fastest of the aggressive WAND strategies, using $\theta = 2.0$, is around an order of magnitude slower than pure Boolean filtering – at least one of the postings lists for the terms must be fully scanned before any documents can be returned at all, and the per document processing cost is also higher. That aggressive WAND variant is in turn is another one to two orders of magnitude faster than the set-safe $\theta = 1.0$ WAND version.

The scored Boolean WAND method provides a different computational profile to the aggressive WAND implementations. Regardless of k, it performs roughly the same amount of work in terms of scoring; and the reduced number of heap operations when k is small is not enough to have a major influence on execution cost.

Filter-Stage Tradeoffs

Figure 9 depicts filter stage execution cost, now as a function of $\mathsf{MED}_{\mathsf{RBP0.95}}$, as k is varied. The suite of alternative methods describes a trade-off frontier that defines

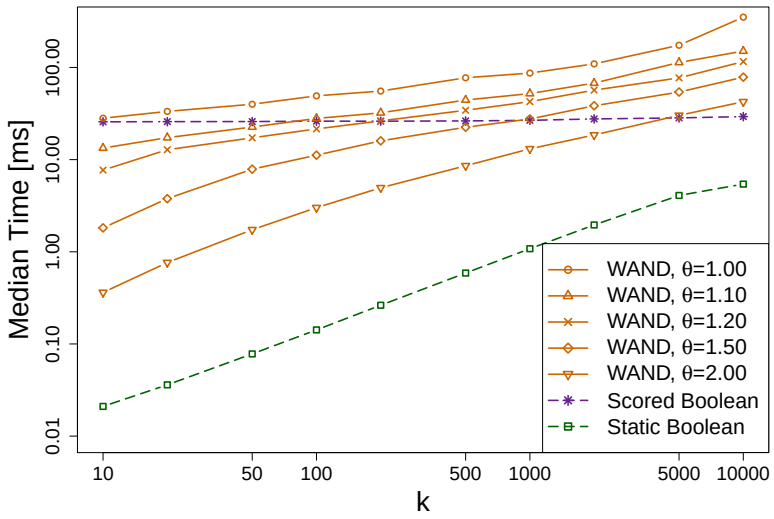

Fig. 8: Median query execution time in milliseconds for fully in-memory execution, plotted as a function of k, the number of documents required, using 40,000 queries and the stopped unpruned CW09B collection.

a subset of techniques that are of possible interest. That subset is dominated by aggressive WAND approaches, except when MED is required to be very small. The low fidelity of the static-score conjunctive Boolean methods means that they do not contribute to the frontier except when query processing must be very fast, in which case high MED scores must be tolerated.

Combining the Two Stages

In a retrieval system the overall cost of generating a results page and returning it to the user includes components other than the time spent computing the filtered shortlist of possible answers. Most notably, the cost of performing the final ranking stage on the k_1 documents supplied by the filtering stage must be allowed for, and the cost of creating answer snippets for the top k_2 documents identified by the final ranking stage. If the existence of a "direct" file is assumed, from which a pre-generated set of features for any specified document can be quickly retrieved, these costs are linear in k_1 and k_2 respectively; and with $k_1 \gg k_2$, the cost of the final ranking stage is the critical one. Asadi and Lin [2] experimented with a variety of efficient index representations and showed the average time per document to perform a feature extraction in the ClueWeb09B collection was around 14–20 μs. The number of features required and the respective cost to retrieve or calculate each largely depends on the final ranking stage algorithms used, but a 20 μs per document penalty is a reasonable lower bound for an efficient and effective final stage approach.

Figure 10 is derived from Figure 9, and is generated by adding an allowance for the final stage computation on k documents to each query's execution time, computed at a (conservative) rate of 0.02 milliseconds per document. The Boolean conjunctive

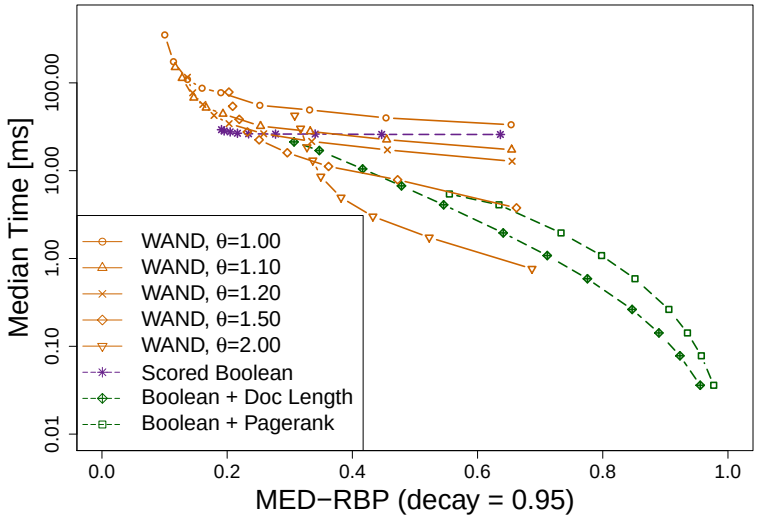

Fig. 9: Filter-stage effectiveness-efficiency trade-off curves, showing the median query execution time as a function of mean $\mathsf{MED}_{\mathsf{RBP}0.95}$ measured for a set of depths k, taken across $40{,}000$ queries on the stopped unpruned CW09B collection.

mechanisms are unattractive when viewed in this light, and the aggressive WAND methods define the frontier for best combinations of efficiency and effectiveness. A very similar pattern of results was obtained with a final stage scoring cost of 0.04 milliseconds per document, indicating that the overall relative costs are not particularly sensitive to the actual value used.

Our work represents an extension and refinement of previous measurements by Asadi and Lin [2, 3]. They also compared a range of filtering methods, including Boolean conjunction and WAND-based disjunction. Building on their results, we have used a faster Boolean computation that allows simple early exit once k documents have been identified, and have also considered aggressive WAND techniques. Moreover, we base our fidelity estimates on MED computations over large numbers of queries and do *not* require relevance judgments; theirs are based on NDCG scores that, as they demonstrate, may be vulnerable to the uncertainties in measurement associated with unjudged documents, and require relevance judgments for all queries used in the final evaluation.

Limitations

We reiterate that we are in no way asserting that the filters used in our experiments represent the state of the art. For example, previous work by Brin and Page [4] and Macdonald et al. [16] has shown that in web search applications structural elements in documents such as anchortext, title, and headings are valuable in early-stage filters in order to identify likely candidate documents. These, and other elements, can be combined with ranked conjunctive queries to build filters that are more selective than the range we have explored here. Richardson et al. [20] show that machine learning

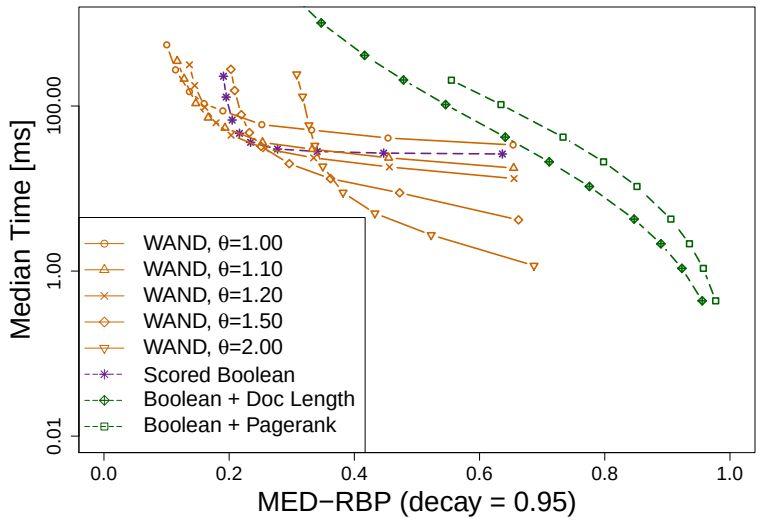

Fig. 10: Combined effectiveness-efficiency trade-off curves for multi-stage retrieval, showing the median query execution time as a function of mean $\mathsf{MED}_{\mathsf{RBP}0.95}$ measured for a set of depths k, taken across 40,000 queries on the stopped unpruned CW09B collection, and with the final ranking stage computation assumed to require 0.02 milliseconds per document.

is also a valuable early-stage tool for static ranking. Rather than define new filters, or even measure the performance of previous ones, our goal has been to describe a *methodology* whereby which any proposed change in a retrieval system – including the introduction of a new filter – can be automatically checked for plausibility using very large query pools, without incurring the expense of relevance judgments across all of those queries.

7 Summary

We have described an approach that can be used for effectiveness measurement for early stage static and query-based filtering, when the purpose is to select a useful subset of a large collection to be ranked using a final-stage reordering mechanism. Our approach does not require the final stage output to be a proper subset of the filtering stage output being evaluated, allowing filters to be evaluated independently; and has the significant benefit of not requiring relevance judgments. Using this technique, and a large collection of queries and documents, we have measured efficiency-effectiveness trade-offs in multi-stage query systems.

In contrast to previous studies of efficiency-effectiveness trade-offs, which were limited to smaller query sets, our methods are applicable to sets containing thousands of queries. As illustrated by the examples and figures of Section 6, our approach allows us to consider the variance in query performance across these thousands of queries. Identifying queries that perform particularly poorly provides insights into the

behavior of first-stage filters, potentially leading to further improvements. In particular, by training over large query sets, we may be able to select a early-stage filtering strategy, or even combinations of filtering stages, for each query on an individual basis by considering features derived from the index (for example, the size of postings lists) and from the query itself (for example, the number of query terms). Note also that MED values can be monitored as queries are processed, meaning that it might also be possible to develop a feedback loop that reacts to mismatches as they are detected, and switches to other (or deeper) filters.

Our experiments have made it clear that Boolean conjunction over all query terms is not suitable as a filtering stage, even when coupled with collection pre-ordering based on a static score such as PageRank or document length. That is, while k-prefixes of Boolean conjunctions can be computed very quickly, that alone is insufficient to provide an interesting multi-stage combination. We also explored a range of WAND-based computations using Okapi document scoring, including an aggressive WAND strategy and a scored Boolean WAND approach, both of which do provide useful trade-offs. By including the cost of a detailed final stage ranker, we were also able to catalog the end-to-end cost of various combinations of filtering stage and final ranking stage. The results of this work demonstrate that the aggressive WAND mechanism offers clear benefits for early stage filtering, since it can be implemented to execute quickly, and provides a range of combinations of θ and k that can be balanced as required, even on a query-by-query basis. Combining those observations with other filtering techniques of the sort mentioned earlier – document structural elements, and learned factors – may well lead to techniques with better cost-effectiveness tradeoffs. We leave that investigation for future work.

Acknowledgments
We thank the referees for their helpful feedback. This work was supported by the National Research Council of Canada, by the Australian Research Council's *Discovery Projects* Scheme (DP140101587 and DP140103256), and by Google. Shane Culpepper is the recipient of an Australian Research Council DECRA Research Fellowship (DE140100275).

References

1. F. H. Al-akashi and D. Inkpen. Query-structure based web page indexing. In *Proc. Text REtrieval Conf.*, 2011.
2. N. Asadi and J. Lin. Document vector representations for feature extraction in multi-stage document ranking. *Information Retrieval J.*, 16(6):747–768, 2013.
3. N. Asadi and J. Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 997–1000, 2013.
4. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. Int. Conf. on the World Wide Web*, pages 107–117, 1998.

5. A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Y. Zien. Efficient query evaluation using a two-level retrieval process. In *Proc. Conf. Information and Knowledge Management*, pages 426–434, 2003.
6. S. Büttcher, C. L. A. Clarke, and G. V. Cormack. *Information Retrieval: Implementing and evaluating search engines.* MIT Press, Cambridge, Massachusetts, 2010.
7. O. Chapelle, D. Metzler, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proc. Conf. Information and Knowledge Management*, pages 621–630, 2009.
8. G. V. Cormack, M. D. Smucker, and C. L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval J.*, 14(5): 441–465, 2011.
9. J. S. Culpepper and A. Moffat. Compact set intersection for inverted indexing. *ACM Trans. Information Systems*, 29(1):1:1–1:25, 2010.
10. E. D. Demaine, A. López-Ortiz, and J. I. Munro. Adaptive set intersections, unions, and differences. In *Proc. ACM-SIAM Symp. on Discrete Algorithms*, pages 743–752, 2000.
11. E. D. Demaine, A. López-Ortiz, and J. I. Munro. Experiments on adaptive set intersections for text retrieval systems. In *Proc. Wkshp. Algorithm Engineering and Experiments*, pages 91–104, 2001.
12. T. Eisayed, N. Asadi, L. Wang, J. Lin, and D. Metzler. UMD and USC/ISI: TREC 2010 Web Track experiments with Ivory. In *Proc. Text REtrieval Conf.*, 2011.
13. K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Information Systems*, 20(4):422–446, 2002.
14. A. Kane and F. W. Tompa. Skewed partial bitvectors for list intersection. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 263–272, 2014.
15. D. Lemire and L. Boytsov. Decoding billions of integers per second through vectorization. *Software Practice & Experience*, 45(1):1–29, 2015.
16. C. Macdonald, R. L. T. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Information Retrieval J.*, 16(5):584–628, 2013.
17. C. Macdonald, R. L. T. Santos, I. Ounis, and B. He. About learning models with multiple query-dependent features. *ACM Trans. Information Systems*, 31(3):11, 2013.
18. A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Information Systems*, 27(1):2.1–2.27, 2008.
19. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999. URL `http://ilpubs.stanford.edu:8090/422/`.
20. M. Richardson, A. Prakash, and E. Brill. Beyond PageRank: Machine learning for static ranking. In *Proc. Int. Conf. on the World Wide Web*, pages 707–715,

2006.

21. T. Sakai. Alternatives to bpref. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 71–78, 2007.
22. T. Sakai and N. Kando. On information retrieval metrics designed for evaluation with incomplete relevance assessments. *Information Retrieval J.*, 11(5):447–470, 2008.
23. M. D. Smucker and C. L. A. Clarke. Time-based calibration of effectiveness measures. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 95–104, 2012.
24. T. Strohman, H. Turtle, and W. B. Croft. Optimization strategies for complex queries. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 219–225, 2005.
25. L. Tan and C. L. A. Clarke. A family of rank similarity measures based on maximized effectiveness difference. *IEEE Trans. Knowledge and Data Engineering*, 27(11):2865–2877, 2015.
26. A. Trotman, X.-F. Jia, and M. Crane. Towards an efficient and effective search engine. In *Wkshp. Open Source IR*, pages 40–47, 2012.
27. H. R. Turtle and J. Flood. Query evaluation: Strategies and optimizations. *Information Processing & Management*, 31(6):831–850, 1995.
28. L. Wang, J. Lin, and D. Metzler. A cascade ranking model for efficient ranked retrieval. In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 105–114, 2011.
29. W. Webber, A. Moffat, and J. Zobel. A similarity measure for indefinite rankings. *ACM Trans. Information Systems*, 28(4):20.1–20.38, Nov. 2010.
30. H. Wu and H. Fang. Document prioritization for scalable query processing. In *Proc. Conf. Information and Knowledge Management*, pages 1609–1618, 2014.
31. J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proc. ACM-SIGIR Int. Conf. Research and Development in Information Retrieval*, pages 307–314, 1998.
32. J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2):6.1–6.56, 2006.